

FreeBSD kézikönyv

A FreeBSD Dokumentációs Projekt

FreeBSD kézikönyv

A FreeBSD Dokumentációs Projekt

Megjelent 1999. február

Copyright © 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010 A
FreeBSD Dokumentációs Projekt

Üdvözljük a FreeBSD világában! Ez a kézikönyv ismerteti a *FreeBSD 7.3-RELEASE*, ill. a *FreeBSD 8.1-RELEASE* telepítését és használatát a mindennapokban. A kézikönyv tartalmán számos független fejlesztő *folyamatosan dolgozik*. Emiatt elképzelhető, hogy bizonyos fejezetek már elavultak és aktualizálásra szorulnak. Amennyiben úgy érezzük, hogy segíteni tudnánk a projekt munkájában, értesítsük a fejlesztőket a FreeBSD Dokumentációs Projekt levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc>) címén! Ezen dokumentum legfrissebb változata mindig elérhető a FreeBSD honlapjáról (<http://www.FreeBSD.org/>) (a korábbi változatok pedig megtalálhatóak a <http://docs.FreeBSD.org/doc/> címen). Ezenkívül még rengeteg más formátumban és tömörítve is letölthető a FreeBSD FTP szerveréről (<ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/>) vagy a tüköroldalak egyikéről. Amennyiben a kézikönyv nyomtatott változatára lenne szükségünk, megvásárolhatjuk a FreeBSD Mall (<http://www.freebsdmail.com/>)-ból. Ha pedig keresni szeretnénk benne, azt a funkciót itt (<http://www.FreeBSD.org/search/index.html>) érhetjük el.

Fordította: Páli Gábor, utolsó ellenőrzés: 2010.11.28.

A dokumentum továbbadása forrás (SGML DocBook) és feldolgozott formában (SGML, HTML, PDF, PostScript, RTF, stb.) módosítással vagy anélkül a következő feltételek mellett lehetséges:

1. A forráskódnak (SGML DocBook) tartalmaznia kell a fenti copyright megjegyzést és a feltételek ezen listáját, valamint a következő jogi nyilatkozatot, bármiféle módosítás nélkül.
2. Feldolgozott dokumentum továbbadásakor (más DTD, PDF, PostScript, RTF és más formátumok) szintén meg kell tartani a fenti copyright megjegyzést, a feltételek listáját, valamint a következő jogi nyilatkozatot a dokumentumban, vagy a dokumentumot kísérő anyagokban.

Fontos: EZT A DOKUMENTUMOT A FREEBSD DOKUMENTÁCIÓS PROJEKT A JELEN FORMÁJÁBAN BIZTOSÍTJA ÉS LEMOND MINDEN KIFEJEZETT VAGY TÖRVÉNYI SZAVATOSSÁGRÓL, BELEÉRTVE AZ ELADHATÓSÁG ÉS EGY ADOTT CÉLRA VALÓ ALKALMASSÁG SZAVATOSSÁGÁT. A FREEBSD DOKUMENTÁCIÓS PROJEKT SEMMILYEN ESETBEN SEM TEHETŐ FELELŐSSÉ A DOKUMENTUM HASZNÁLATÁBÓL EREDŐ BÁRMILYEN KÖZVETLEN, KÖZVETETT JÁRULÉKOS, KÜLÖNLEGES, BÜNTETŐ VAGY KÖVETKEZMÉNYES KÁRÉRT (BELEFOGLALVA, DE NEM KORLÁTOZVA A HELYETTESÍTŐ JAVAK BESZERZÉSÉRE, HASZON, ADAT VAGY PROFIT ELVESZTÉSÉRE, ILLETVE ÜZLETI FORGALOM KIESÉSÉRE) VAGY EGYÉB MÁS ESETBEN SEM, AMIKOR ERŐS TEHER VAGY KÍN (HANYAGSÁG VAGY EGYÉB) ERED A DOKUMENTUM AKÁRMIFÉLE FELHASZNÁLÁSÁBÓL, MÉG HA ERRE KÜLÖN FEL IS HÍVTUK A FIGYELMET.

A FreeBSD a FreeBSD Foundation bejegyzett védjegye.

A 3Com és HomeConnect a 3Com Corporation bejegyzett védjegyei.

A 3ware és Escalade a 3ware Inc. bejegyzett védjegyei.

Az ARM az ARM Limited bejegyzett védjegye.

A Adaptec az Adaptec, Inc. bejegyzett védjegye.

Az Adobe, Acrobat, Acrobat Reader, és PostScript az Adobe Systems Incorporated bejegyzett védjegyei, vagy védjegyei az Egyesült Államokban és/vagy más országokban.

Az Apple, AirPort, FireWire, Mac, Macintosh, Mac OS, Quicktime, és TrueType az Apple Computer, Inc., bejegyzett védjegyei az Egyesült Államokban és más országokban.

A Corel és WordPerfect a Corel Corporation és/vagy leányvállalatainak bejegyzett védjegye Kanadában, az Egyesült Államokban és/vagy más országokban.

A Sound Blaster a Creative Technology Ltd. védjegye az Egyesült Államokban és/vagy más országokban.

A CVSup John D. Polstra bejegyzett védjegye.

A Heidelberg, Helvetica, Palatino, és Times Roman a Heidelberger Druckmaschinen AG bejegyzett védjegyei, vagy védjegyei az Egyesült Államokban és más országokban.

Az IBM, AIX, EtherJet, Netfinity, OS/2, PowerPC, PS/2, S/390 és ThinkPad az International Business Machines Corporation védjegyei az Egyesült Államokban, más országokban, vagy mindkettőben.

Az IEEE, POSIX és 802 az Institute of Electrical and Electronics Engineers, Inc. bejegyzett védjegyei az Egyesült Államokban.

Az Intel, Celeron, EtherExpress, i386, i486, Itanium, Pentium és Xeon az Intel Corporation vagy leányvállalatainak védjegyei vagy bejegyzett védjegyei az Egyesült Államokban és más országokban.

Az Intuit és Quicken az Intuit Inc., vagy valamely leányvállalatának bejegyzett védjegyei és/vagy bejegyzett szervizmárkái az Egyesült Államokban és más országokban.

A Linux Linus Torvalds bejegyzett védjegye.

Az LSI Logic, AcceleRAID, eXtremeRAID, MegaRAID és Mylex az LSI Logic Corp. védjegyei vagy bejegyzett védjegyei.

Az M-Systems és DiskOnChip az M-Systems Flash Disk Pioneers, Ltd. védjegyei vagy bejegyzett védjegyei.

A Macromedia, Flash és Shockwave a Macromedia, Inc. védjegyei és/vagy bejegyzett védjegyei az Egyesült Államokban és/vagy más országokban.

A Microsoft, IntelliMouse, MS-DOS, Outlook, Windows, Windows Media és Windows NT a Microsoft Corporation bejegyzett védjegyei, vagy védjegyei az Egyesült Államokban és/vagy más országokban.

A Netscape és a Netscape Navigator a Netscape Communications Corporation bejegyzett védjegyei az Egyesült Államokban és más országokban.

A GateD és NextHop a NextHop bejegyzett és nem bejegyzett védjegyei az Egyesült Államokban és más országokban.

A Motif, OSF/1 és UNIX a The Open Group bejegyzett védjegyei, az IT DialTone és a The Open Group pedig védjegyei az Egyesült Államokban és/vagy más országokban.

Az Oracle az Oracle Corporation bejegyzett védjegye.

A PowerQuest és PartitionMagic a PowerQuest Corporation bejegyzett védjegyei az Egyesült Államokban és/vagy más országokban.

A RealNetworks, RealPlayer és RealAudio a RealNetworks, Inc. bejegyzett védjegyei.

A Red Hat és RPM a Red Hat, Inc. védjegyei vagy bejegyzett védjegyei az Egyesült Államokban és más országokban.

A SAP, R/3 és mySAP a SAP AG védjegyei vagy bejegyzett védjegyei Németországban és sok más országban is világszerte.

A Sun, Sun Microsystems, Java, Java Virtual Machine, JavaServer Pages, JDK, JRE, JSP, JVM, Netra, OpenJDK, Solaris, StarOffice, Sun Blade, Sun Enterprise, Sun Fire, SunOS, Ultra és VirtualBox a Sun Microsystems, Inc. védjegyei vagy bejegyzett védjegyei az Egyesült Államokban és más országokban.

A Symantec és Ghost a Symantec Corporation bejegyzett védjegyei az Egyesült Államokban és más országokban.

A MATLAB a The MathWorks, Inc. bejegyzett védjegye.

A SpeedTouch a Thomson védjegye.

A U.S. Robotics és Sportster a U.S. Robotics Corporation. bejegyzett védjegyei.

A VMware a VMware, Inc. védjegye.

A Waterloo Maple és Maple a Waterloo Maple Inc. védjegyei vagy bejegyzett védjegyei.

A Mathematica a Wolfram Research, Inc. védjegye.

Az XFree86 az XFree86 Project, Inc. védjegye.

Az Ogg Vorbis és Xiph.Org a Xiph.Org védjegyei.

A gyártók és terjesztők által használt megnevezések közül sok védjegy jogot követel. Ahol ilyen megnevezés tűnik fel ebben a dokumentumban, és a FreeBSD Projektnek tudomása volt a védjegyről, a megnevezést a “™” vagy a “®” szimbólum követi.

Tartalom

Előszó	xiv
I. Bevezetés.....	xxii
1. Bemutakozás	1
1.1. Áttekintés.....	1
1.2. Üdvözljük a FreeBSD-ben!	1
1.3. A FreeBSD Projektről.....	5
2. A FreeBSD telepítése	11
2.1. Áttekintés.....	11
2.2. Hardverkövetelmények.....	11
2.3. A telepítés előtt elvégzendő feladatok.....	12
2.4. A telepítés megkezdése.....	19
2.5. A sysinstall bemutatása.....	26
2.6. Lemezterület lefoglalása.....	30
2.7. A telepítendő összetevők kiválasztása.....	43
2.8. A telepítés eszközének kiválasztása	46
2.9. A telepítés véglegesítése.....	48
2.10. A telepítés után	49
2.11. Hibakeresés.....	83
2.12. Telepítési útmutató haladóknak	86
2.13. Saját telepítőeszköz elkészítése	88
3. A UNIX alapjai	94
3.1. Áttekintés.....	94
3.2. Virtuális konzolok és terminálok	94
3.3. Engedélyek	97
3.4. A könyvtárak elrendezése.....	102
3.5. A lemezek szervezése	104
3.6. Állományrendszerek csatlakoztatása és leválasztása.....	109
3.7. Folyamatok	112
3.8. Démonok, jelzések és a futó programok leállítása	114
3.9. Parancsértelmezők	116
3.10. Szövegszerkesztők.....	118
3.11. Eszközök és eszközléírók	118
3.12. Bináris formátumok.....	119
3.13. Bővebben olvashatunk.....	120
4. Alkalmazások telepítése: csomagok és portok.....	123
4.1. Áttekintés.....	123
4.2. Az alkalmazások telepítésének összefoglalása	123
4.3. A számunkra szükséges alkalmazások felkutatása	125
4.4. A csomagrendszer használata	126
4.5. A Portgyűjtemény használata	129
4.6. Telepítés utáni teendők	138
4.7. Teendő a sérült portokkal.....	139
5. Az X Window System	141
5.1. Áttekintés.....	141
5.2. Az X áttekintése.....	141

5.3. Az X11 telepítése.....	144
5.4. Az X11 beállítása.....	144
5.5. Betűtípusok használata az X11-ben.....	150
5.6. Az X bejelentkeztető képernyője.....	154
5.7. Munkakörnyezetek	156
II. Gyakori feladatok	162
6. Asztali alkalmazások.....	163
6.1. Áttekintés.....	163
6.2. Böngészők.....	163
6.3. Irodai eszközök.....	167
6.4. Dokumentum-megjelenítők.....	170
6.5. Pénzügyi szoftverek.....	171
6.6. Összefoglalás	173
7. Multimédia	175
7.1. Áttekintés.....	175
7.2. A hangkártya beállítása.....	175
7.3. MP3.....	179
7.4. Videók lejátszása	182
7.5. TV kártyák beállítása.....	190
7.6. Lapolvasók.....	192
8. A FreeBSD rendszermag testreszabása.....	197
8.1. Áttekintés.....	197
8.2. Miért készítsünk saját rendszermagot?.....	197
8.3. A rendszerünkben levő hardverek összeszedése	198
8.4. Meghajtók, alrendszerek és modulok	199
8.5. Saját rendszermag készítése és telepítése.....	199
8.6. A konfigurációs állomány.....	202
8.7. Ha valamilyen hiba történne	216
9. Nyomtatás.....	218
9.1. Áttekintés.....	218
9.2. Bevezetés	218
9.3. Kezdeti beállítások.....	219
9.4. Magasszintű nyomtatóbeállítás.....	232
9.5. A nyomtatók használata.....	262
9.6. Más nyomtatási rendszerek.....	270
9.7. Hibakeresés.....	271
10. Bináris Linux kompatibilitás	275
10.1. Áttekintés.....	275
10.2. Telepítés.....	275
10.3. A Mathematica® telepítése	279
10.4. A Maple™ telepítése	281
10.5. A MATLAB® telepítése.....	283
10.6. Az Oracle® telepítése.....	286
10.7. Az SAP® R/3® telepítése	289
10.8. Témák haladóknak.....	310

III. Rendszeradminisztráció.....	312
11. Beállítás és finomhangolás.....	313
11.1. Áttekintés.....	313
11.2. Kezdeti beállítások.....	313
11.3. A mag beállítása	315
11.4. Az alkalmazások beállítása.....	315
11.5. Szolgáltatások indítása.....	316
11.6. A cron segédprogram beállítása	317
11.7. Az rc használata FreeBSD alatt	319
11.8. A hálózati kártyák beállítása.....	321
11.9. Virtuális címek.....	327
11.10. Konfigurációs állományok.....	328
11.11. Finomhangolás a sysctl használatával	332
11.12. A lemezek finomhangolása.....	333
11.13. A rendszermag korlátainak finomhangolása.....	337
11.14. A lapozóterület bővítése	341
11.15. Energia- és erőforrásgazdálkodás.....	342
11.16. A FreeBSD ACPI támogatásának használata és nyomonkövetése.....	343
12. A FreeBSD rendszerindítási folyamata.....	351
12.1. Áttekintés.....	351
12.2. A rendszerindítás problémája	351
12.3. A boot manager és az indulás fokozatai	352
12.4. Kapcsolat a rendszermaggal a rendszerindítás folyamán	358
12.5. Eszköz útmutatók (device.hints).....	359
12.6. Init: A folyamatirányítás elindítása	360
12.7. A leállítási folyamat.....	361
13. Felhasználók és hozzáférések alapvető kezelése.....	362
13.1. Áttekintés.....	362
13.2. Bevezetés	362
13.3. Az adminisztrátori hozzáférés	364
13.4. Rendszerhozzáférések.....	364
13.5. Felhasználói hozzáférések	364
13.6. A hozzáférések módosítása.....	365
13.7. A felhasználók korlátozása	369
13.8. Csoportok.....	372
14. Biztonság.....	374
14.1. Áttekintés.....	374
14.2. Bevezetés	374
14.3. A FreeBSD védelme	376
14.4. DES, Blowfish, MD5 és a Crypt.....	384
14.5. Egyszeri jelszavak.....	384
14.6. A TCP kapcsolatok burkolása.....	388
14.7. KerberosIV	391
14.8. Kerberos5	398
14.9. OpenSSL.....	407
14.10. VPN IPsec felett	410
14.11. OpenSSH	416
14.12. Az állományrendszerek hozzáféréseit vezérlő listák	421

14.13. A külső programok biztonsági problémáinak figyelése.....	423
14.14. A FreeBSD biztonsági figyelmeztetései	424
14.15. A futó programok nyilvántartása.....	426
15. A jail alrendszer	428
15.1. Áttekintés	428
15.2. A jail alrendszerhez kapcsolódó fogalmak	428
15.3. Bevezetés	429
15.4. A jailek létrehozása és vezérlése	430
15.5. Finomhangolás és karbantartás.....	432
15.6. A jailek alkalmazása	433
16. Kötelező hozzáférés-vezérlés (MAC)	439
16.1. Áttekintés	439
16.2. A fejezet fontosabb fogalmai.....	440
16.3. A MAC ismertetése	441
16.4. Bővebben a MAC címkéiről	442
16.5. A védelem megtervezése	447
16.6. A modulok beállítása	447
16.7. A seeotheruids MAC-modul.....	448
16.8. A bsdextended MAC-modul	448
16.9. Az ifoff MAC-modul	449
16.10. A portacl MAC-modul.....	450
16.11. A partition MAC-modul	451
16.12. A többszintű biztonsági MAC-modul.....	452
16.13. A Biba MAC-modul	454
16.14. A LOMAC MAC-modul.....	456
16.15. A Nagios elzárása a MAC rendszerrel.....	456
16.16. A felhasználók korlátozása	460
16.17. A hibák elhárítása a MAC rendszerben	461
17. Biztonsági események vizsgálata	463
17.1. Áttekintés	463
17.2. A fejezet fontosabb fogalmai.....	463
17.3. A vizsgálat támogatásának telepítése	464
17.4. A vizsgálat beállítása	465
17.5. A vizsgálati alrendszer használata.....	468
18. Háttértárak	471
18.1. Áttekintés	471
18.2. Az eszközök elnevezései	471
18.3. Lemezek hozzáadása	472
18.4. RAID.....	474
18.5. USB tárolóeszközök	479
18.6. Lézeres tárolóeszközök (CD-k) létrehozása és használata	481
18.7. Lézeres tárolóeszközök (DVD-k) létrehozása és használata	487
18.8. Hajlékonylemezek létrehozása és használata	492
18.9. Szalagok létrehozása és használata.....	494
18.10. Biztonsági mentés hajlékonylemezekre.....	496
18.11. Mentési stratégiák.....	498
18.12. Alapvető tudnivalók a biztonsági mentésről.....	499
18.13. Hálózat, memória és állomány alapú állományrendszerek.....	503

18.14. Az állományrendszerek pillanatképei.....	505
18.15. Az állományrendszerek kvótái	507
18.16. A lemezzartíciók titkosítása	510
18.17. A lapozóterület titkosítása	516
19. GEOM: A moduláris lemezzervező rendszer	519
19.1. Áttekintés	519
19.2. A GEOM bemutatása.....	519
19.3. RAID0 - Csíkozás	519
19.4. RAID1 - Tükörozés	521
19.5. Eszközök hálózati illesztése a GEOM-ban.....	524
19.6. A lemezes eszközök címkézése	524
19.7. Naplózó UFS GEOM-on keresztül	527
20. Támogatott állományrendszerek	529
20.1. Áttekintés	529
20.2. A Z állományrendszer (ZFS)	529
21. A Vinum kötetkezelő.....	537
21.1. Áttekintés	537
21.2. Kicsik a lemezeink.....	537
21.3. A hozzáférési idők szűk keresztmetszetei	537
21.4. Adatintegritás.....	539
21.5. A Vinum objektumai.....	540
21.6. Példák	542
21.7. Az objektumok elnevezése	547
21.8. A Vinum beállítása	549
21.9. Rendszerindítás Vinum-kötetről	550
22. Virtualizáció	556
22.1. Áttekintés	556
22.2. A FreeBSD mint vendég.....	556
22.3. A FreeBSD mint gazda.....	575
23. Honosítás: Az I18N/L10N használata és beállítása	578
23.1. Áttekintés	578
23.2. Az alapok	578
23.3. A honosítás használata.....	579
23.4. I18N programok fordítása.....	585
23.5. A FreeBSD honosítása adott nyelvekre	585
24. A FreeBSD frissítése és frissen tartása	589
24.1. Áttekintés	589
24.2. A FreeBSD frissítése	589
24.3. A Portgyűjtemény frissítése a Portsnap használatával	596
24.4. A dokumentáció frissítése.....	597
24.5. A fejlesztői ág követése	602
24.6. A forrás szinkronizálása	606
24.7. Az alaprendszer újrafordítása	607
24.8. A források követése több géppel	622
25. DTrace	625
25.1. Áttekintés	625
25.2. Eltérések az implementációban	625
25.3. A DTrace támogatásának engedélyezése.....	626

25.4. A DTrace használata	627
25.5. A D nyelv	629
IV. Hálózati kommunikáció	630
26. Soros vonali kommunikáció	631
26.1. Áttekintés	631
26.2. Bevezetés	631
26.3. Terminálok	636
26.4. Betárcsázós szolgáltatások	641
26.5. A betárcsázós szolgáltatások használata	649
26.6. A soros vonali konzol beállítása	652
27. A PPP és a SLIP	661
27.1. Áttekintés	661
27.2. A felhasználói PPP alkalmazása	661
27.3. A rendszerszintű PPP alkalmazása	673
27.4. PPP kapcsolatok hibaelhárítása	681
27.5. A PPP használata Ethernet felett (PPPoE)	684
27.6. PPP ATM felett (PPPoA)	686
27.7. A SLIP használata	689
28. Elektronikus levelezés	698
28.1. Áttekintés	698
28.2. Az elektronikus levelezés használata	698
28.3. A sendmail beállítása	701
28.4. A levéltovábbító ügynök megváltoztatása	703
28.5. A hibák elhárítása	705
28.6. Komolyabb témák	709
28.7. SMTP és az UUCP	711
28.8. Csak küldés beállítása	713
28.9. Levelezés betárcsázós kapcsolattal	714
28.10. Az SMTP hitelesítése	715
28.11. Levelező kliensek	716
28.12. A fetchmail használata	723
28.13. A procmail használata	724
29. Hálózati szerverek	726
29.1. Áttekintés	726
29.2. Az inetd "szuperszerver"	726
29.3. A hálózati állományrendszer (NFS)	730
29.4. Hálózati információs rendszer (NIS/YP)	736
29.5. A hálózat automatikus beállítása (DHCP)	753
29.6. Névfeloldás (DNS)	758
29.7. Az Apache webservizet	770
29.8. Állományok átvitele (FTP)	775
29.9. Állomány- és nyomtatási szolgáltatások Microsoft Windows kliensek számára (Samba)	777
29.10. Az órák egyeztetése az NTP használatával	780
29.11. Távoli gépek naplózása <code>syslogd</code> használatával	783
30. Tűzfalak	787
30.1. Bevezetés	787
30.2. Röviden a tűzfalokról	787

30.3. Tűzfalak	788
30.4. Az OpenBSD csomagszűrője (PF) és az ALTQ	788
30.5. Az IPFILTER (IPF) tűzfal	792
30.6. IPFW	811
31. Egyéb haladó hálózati témák	830
31.1. Áttekintés	830
31.2. Átjárók és az útválasztás	830
31.3. Vezeték nélküli hálózatok	836
31.4. Bluetooth	856
31.5. Hálózati hidak	864
31.6. Linkek összefűzése és hibatűrése	870
31.7. Lemez nélküli működés	874
31.8. ISDN	881
31.9. Hálózati címfordítás	884
31.10. Párhuzamos vonali IP (PLIP)	888
31.11. Az IPv6	890
31.12. Az Aszinkron adatátviteli mód (ATM)	895
31.13. A Közös cím redundancia protokoll (CARP)	896
V. Függelék	899
A. A FreeBSD beszerzése	900
A.1. CD és DVD kiadók	900
A.2. FTP oldalak	902
A.3. BitTorrent	910
A.4. Anonim CVS	911
A.5. A CTM használata	913
A.6. A CVSup használata	917
A.7. CVS címkék	936
A.8. AFS oldalak	942
A.9. Rsync oldalak	943
B. Irodalomjegyzék	945
B.1. A FreeBSD-ről szóló könyvek és folyóiratok	945
B.2. Felhasználói kézikönyvek	946
B.3. Rendszeradminisztratori kézikönyvek	946
B.4. Programozói kézikönyvek	947
B.5. Az operációs rendszerek belső működéséről	948
B.6. Biztonságról szóló írások	948
B.7. Hardverrel foglalkozó írások	948
B.8. UNIX történelem	949
B.9. Magazinok és folyóiratok	949
C. Források az interneten	951
C.1. Levelezési listák	951
C.2. Usenet hírcsoportok	971
C.3. Világhálós szolgáltatások	972
C.4. E-mail címek	976
D. PGP-kulcsok	977
D.1. Tiszttségviselők	977
D.2. A Core Team tagjai	977

D.3. Fejlesztők	979
A FreeBSD-s szakkifejezések gyűjteménye	1059
Végző.....	1084

A táblázatok listája

2-1. Példa egy eszközléptárra	13
2-2. Az első lemez partícióinak kiosztása	37
2-3. Több lemez partícióinak kiosztása	38
2-4. FreeBSD 7.x és 8.x ISO image-ek nevei és jelentései	89
3-1. Lemezes eszközök kódjai	108
18-1. A fizikai lemezek elnevezésének szabályai	471
21-1. Vinum erezések	541
26-1. A DB-25 DB-25 közti null-modem kábel	632
26-2. A DB-9 DB-9 közti null-modem kábel	632
26-3. DB-9 DB-25 közti null-modem kábel	633
26-4. A jelek neve	641
31-1. A párhuzamos kábel hálózati használatra alkalmas bekötése	888
31-2. Fenntartott IPv6 címek	891

Előszó

Kiknek szánjuk ezt a könyvet

A FreeBSD-t még nem ismerők felfedezhetik, hogy a könyv első része a FreeBSD telepítésének folyamatán vezeti keresztül a felhasználót, valamint érintőlegesen bemutatja az ezt alátámasztó UNIX®-os alapfogalmakat és szabályokat. Ennek a résznek a végigjárása nem kíván többet, csupán egy kis felfedező kedvet, illetve a menet közben bemutatott új fogalmak befogadását.

Ha túljutottunk rajta, a kézikönyv második, jóval terjedelmesebb része a FreeBSD-t használó rendszergazdák számára nyújt mindenféle témában minden részletre kiterjedő referenciát. Ezek közül egyes fejezetek elvárnak némi előzetes felkészülést, amelyet minden fejezet áttekintésében említeni is fogunk.

További információkért olvassuk át a B Függelékét.

Változtatások a harmadik kiadás óta

A kézikönyv jelenleg interneten elérhető változata számtalan önkéntes által az utóbbi 10 évben végzett együttes erőfeszítéseinek eredményeit tükrözi. A 2004-ben két kötetben megjelentetett harmadik kiadás óta a következő fontosabb változások történtek:

- 25 fejezet: készült egy új fejezet a DTrace nevű teljesítmény-elemző eszközről.
- 20 fejezet: ebben a fejezetben a FreeBSD és a különböző más rendszerekhez fejlesztett állományrendszerek viszonyát mutatjuk be, többek a Sun™ ZFS megoldását.
- 17 fejezet: ez a fejezet FreeBSD új biztonsági fejlesztéseit foglalja össze és mutatja be a használatukat.
- 22 fejezet: ebben az új fejezetben a FreeBSD rendszerhez és rajta elérhető különböző virtualizációs technológiákról szólnak.

Változtatások a második kiadás (2004) óta

A harmadik kiadás a FreeBSD Dokumentációs Projekt tagjainak két évi kemény munkájának gyümölcse. A nyomtatott változat már olyan nagyra nőtt, hogy két külön kötetben kellett kiadnunk. Az alábbi fontosabb változtatások jelentek meg ebben az új kiadásban:

- 11 fejezet: a beállításra és finomhangolásra vonatkozó részeket bővítettük az ACPI energia- és erőforrás gazdálkodásról szóló részekkel, a `cron` rendszerprogrammal, illetve még több, a rendszermag finomhangolását elősegítő opció leírásával.
- 14 fejezet: a biztonságról szóló részt bővítettük a virtuális magánhálózatokról (VPN-ekről), állományrendszeri hozzáférés-vezérlési listákról (ACL-ek) szóló elemekkel, valamint biztonságtechnikai tanácsokkal.
- 16 fejezet: a kötelező hozzáférés-vezérlésről (MAC-ről) szóló fejezet teljesen új ebben a kiadásban. Bemutatja, mi is az a MAC és hogyan hasznosítható egy FreeBSD-s rendszer biztonságossá tételében.
- 18 fejezet: a háttértárakat tartalmazó részt bővítettük az USB-tákról, állományrendszeri pillanatképeiről, lemezkvó tákról, állomány- és hálózat alapú állományrendszerekről, továbbá a titkosított partíciókról szóló részekkel.

- 21 fejezet: a Vinum egy új fejezet ebben a kiadásban. Bemutatja a Vinum logikaikötet-kezelő használatát, aminek segítségével eszközfüggetlen módon hozhatunk létre logikai lemezeket, szoftveres RAID-0, RAID-1 és RAID-5 konfigurációkat.
- Bekerült egy hibaelhárításról szóló rész a 27 fejezet PPP és SLIP leírásához.
- 28 fejezet: az elektronikus levelezést ismertető részt bővítettük a különféle levéltovábbító rendszerekről, az SMTP hitelesítésről, UUCP protokollról, a **fetchmail** és **procmial** programokról szóló elemekkel, valamint egyéb, haladókat megcélzó témákkal.
- 29 fejezet: a hálózati szervereket ismertető rész egy teljesen új fejezet ebben a kiadásban. Benne megtalálható az **Apache HTTP szerver**, az **ftpd** szerver, illetve a Microsoft® Windows®-os kliensek számára megfelelő **Samba** szerver beállítása. Az érthetőség kedvéért egyes részek átkerültek ide a 31 fejezet, vagyis a haladó hálózati témákat tárgyaló fejezetből.
- 31 fejezet: a haladó hálózati témákat tartalmazó részt kiegészítettük a FreeBSD és a Bluetooth® eszközök kapcsolatáról, a vezeték nélküli hálózatokról és az aszinkron adatátvitel módról (ATM-ről) szóló ismeretekkel.
- Létrehoztunk egy szójegyzéket abból a célból, hogy a könyvben használt definíciók és szakkifejezések egyetlen központi helyen össze legyenek foglalva.
- Számos esztétikai javítást eszközöltünk a könyvben található ábrákon és táblázatokon.

Változtatások az első kiadás (2001) óta

A második kiadás a FreeBSD Dokumentációs Projekt tagjainak két évi komoly munkájának eredménye. Az alábbi fontosabb változtatások jelennek meg ebben a kiadásban:

- Bekerült egy teljes tárgy- és névmutató.
- Mindegyik ASCII-ábrát grafikusak váltották fel.
- Mindegyik fejezet elejére odakerült egy általános áttekintés, ami egy rövid összefoglalást ad a fejezet tartalmáról, valamint közli az elolvasásához szükséges ismereteket.
- A tartalmat felosztottuk logikailag három részre: “Bevezetés”, “Rendszeradminisztráció” és “Függelék”.
- A 2 fejezetet (“A FreeBSD telepítése”) teljesen újraírtuk és sok-sok illusztráció is hozzáadásra került a könnyebb megértés érdekében.
- A 3 fejezet (“A UNIX alapjai”) kiegészült a futó programokról, démonokról és jelzésekről szóló további hasznos információkkal.
- A 4 fejezetet (“Alkalmazások telepítése”) bővítettük a bináris csomagkezelésről szóló további ismeretekkel.
- A 5 fejezet (“Az X Window System”) teljes újraíráson ment át, aminek folyamán igyekeztünk nagyobb hangsúlyt helyezni a modern asztali technológiák, mint pl. a **KDE** és **GNOME** XFree86™ 4.X-en történő használatának leírására.
- A 12 fejezet (“A FreeBSD rendszerindítási folyamata”) kibővült.
- A 18 fejezet (“Háttértárak”) két, korábban külön levő fejezet, a “Lemezek” és “Biztonsági mentések” összeolvasztásából jött létre. Úgy éreztük, a bennük helyet kapott témákat sokkal könnyebb úgy megérteni, ha egyetlen fejezetben tárgyaljuk ezeket. Egy (hardveres és szoftveres) RAID-ről szóló rész is belekerült.
- A 26 fejezet (“Soros vonali kommunikáció”) teljes átszervezésre került, valamint a FreeBSD 4.X/5.X verziókhoz igazítottuk.

- A 27 fejezet (“A PPP és a SLIP”) lényegesen sokat fejlődött.
- Számos új rész került a 31 fejezetbe (“Egyéb haladó hálózati témák”).
- A 28 fejezet (“Elektronikus levelezés”) kibővült a **sendmail** beállításáról tartalmazó újabb információkkal.
- A 10 fejezet (“Bináris Linux kompatibilitás”) kiegészült az **Oracle®** és a **SAP® R/3®** telepítését bemutató részekkel.
- Az alábbi új témák kerültek tárgyalásra a második kiadásban:
 - Beállítás és finomhangolás (11 fejezet)
 - Multimédia (7 fejezet)

A könyv felépítése

A könyvet négy logikailag elkülönülő részre osztottuk fel. Az első, *Bevezetés* című részben bemutatjuk a FreeBSD telepítését és használatának alapjait. Elgondolásunk szerint az itt szereplő fejezeteket sorban érdemes elolvasni, esetenként kihagyni azokat, amelyek már az olvasó számára ismert témákat dolgoznak fel. A második, *Gyakori feladatok* című részben megismerhetjük a FreeBSD néhány gyakorta használt lehetőségét. Ez a rész, valamint az ezt követő összes többi tetszőleges sorrendben olvasható. Mindegyik fejezet egy rövidke összefoglalással kezdődik, amely ismerteti, az olvasótól milyen jellegű tapasztalatokat vár el a fejezet megértése. Célja, hogy segítsen az olvasónak megtalálni a számára érdekes témákat. A harmadik, *Rendszeradminisztráció* című részben rendszergazdai feladatokat tárgyalunk. A negyedik, *Hálózati kommunikáció* című részben hálózatok és szerverek üzemeltetésével kapcsolatos ismereteket foglaltunk össze. Végül, az ötödik rész tartalmazza a függelékét és az irodalomjegyzéket, hivatkozásokat.

1 fejezet: Bemutakozás

A FreeBSD bemutatkozik az új felhasználóknak. Szó esik a FreeBSD Projekt történetéről, célkitűzéseiről és a fejlesztési modelljéről.

2 fejezet: A FreeBSD telepítése

Végigvezetjük a felhasználót a telepítési folyamat egészen. Bizonyos rendhagyó kérdések, mint például a soros konzolon keresztül történő telepítés is terítékre kerülnek.

3 fejezet: A UNIX alapjai

Sorra vesszük a FreeBSD operációs rendszer alapvető parancsait és lehetőségeit. Amennyiben már jártasak vagyunk valamilyen szinten a Linux® vagy más UNIX-típusú rendszerek használatában, nyugodtan kihagyhatjuk ezt a fejezetet.

4 fejezet: Alkalmazások telepítése, csomagok és portok

Megismerhetjük, miként tudunk külső cégek által fejlesztett alkalmazásokat telepíteni a FreeBSD “Portgyűjteményének” (FreeBSD Ports Collection) vagy a megszokott bináris csomagok használatán keresztül.

5 fejezet: Az X Window System

Általános bemutatásra kerül az X Window System, valamint az X11 használata a FreeBSD-n. Ezenkívül olvashatunk az elterjedtebb munkakörnyezetekről, mint pl. a **KDE** és a **GNOME**.

6 fejezet: Asztali alkalmazások

Felsoroljuk az ismertebb asztali alkalmazásokat: webböngészőket és alkalmazói programcsomagokat, és bemutatjuk, hogyan telepítsük ezeket FreeBSD-re.

7 fejezet: Multimédia

Megtudhatjuk, hogyan állítsuk be a zene- és videolejátszást rendszerünkön. Emellett olvashatunk néhány multimédiás alkalmazás használatáról is.

8 fejezet: A FreeBSD rendszermag testreszabása

Kifejtjük, miért lehet szükségünk egy új rendszermag konfigurálására, és részletesen végigjárjuk egy rendszermag konfigurációjának, fordításának és telepítésének lépéseit.

9 fejezet: Nyomtatás

Ismertetjük, hogyan lehet nyomtatókat használni FreeBSD alatt, beleértve a munkalapok készítésének mikéntjét, a nyomtatóhasználat nyilvántartását és a kezdeti beállításokat.

10 fejezet: Bináris Linux kompatibilitás

Megismerhetjük a FreeBSD bináris Linux kompatibilitásához kapcsolódó lehetőségeket. Ezenfelül részletekre is kitérő telepítési útmutatót találhatunk különböző népszerű linuxos alkalmazásokhoz, mint például az **Oracle**, **SAP R/3** és a **Mathematica®**.

11 fejezet: Beállítás és finomhangolás

Megismerhetjük a FreeBSD azon paramétereit, amelyek megfelelő állításával a rendszergazdák a lehető legtöbbet képesek kihozni FreeBSD rendszerükből. Ezenkívül bemutatásra kerül a FreeBSD-ben használt számos konfigurációs állomány, valamint hogy ezeket hol találhatjuk meg.

12 fejezet: A FreeBSD rendszerindítási folyamata

Tartalmazza a FreeBSD rendszerindítási folyamatának leírását, és elmagyarázza, miként lehet ezt vezérelni a konfigurációs beállítások segítségével.

13 fejezet: Felhasználók és hozzáférések alapvető kezelése

Bemutatja a felhasználói fiókok létrehozását és kezelését. Emellett megemlíti a felhasználókra érvényesíthető erőforrás-megszorításokat, illetve egyéb fiókkezelési feladatokat.

14 fejezet: Biztonság

Bemutatásra kerül a FreeBSD rendszerünk biztonságossá tételére alkalmas számos különféle eszköz, többek közt a Kerberos, IPsec és az OpenSSH.

15 fejezet: A jail alrendszer

Megtudhatjuk, hogyan működik az alkalmazások elszigeteléséért felelős jail alrendszer, valamint miben emelkedik ki a FreeBSD-ben is megtalálható hagyományos “chroot” megoldással szemben.

16 fejezet: Kötelező hozzáférés-vezérlés

Megismerhetjük a kötelező hozzáférés-vezérlést (MAC-et), valamint azt, hogyan is tudjuk felhasználni egy FreeBSD-s rendszer biztonsága érdekében.

17 fejezet: Biztonsági események vizsgálata

Kiderül, mit jelent a FreeBSD-ben az események vizsgálata, illetve mindez hogyan telepíthető, konfigurálható és miként tudjuk a vizsgálatok adatait kielemezni vagy felügyelni.

18 fejezet: Háttértárak

Bemutatásra kerül, miként kezelhetjük a háttértárolókat és állományrendszereket a FreeBSD-ben. Ide tartoznak a fizikai lemezek, RAID-tömbök, optikai és szalagos egységek, memória alapú lemezek és a hálózati állományrendszerek.

19 fejezet: GEOM, a moduláris lemezszervező rendszer

Megismerhetjük a FreeBSD-ben jelenlevő GEOM alrendszert és az általa támogatott különböző RAID-szintek beállítását.

20 fejezet: Támogatott állományrendszerek

A FreeBSD operációs rendszer számára nem natív állományrendszerekkel foglalkozik, például a Sun Z állományrendszerével.

21 fejezet: A Vinum kötetkezelő

Megtudhatjuk, hogyan használjuk a Vinumot, a logikaikötet-kezelőt, amely eszközfüggetlen logikai lemezeket, szoftveres RAID-0, RAID-1 és RAID-5 konfigurációkat biztosít.

22 fejezet: Virtualizáció

Tartalmazza a virtualizációs rendszerek által felkínált lehetőségek bemutatását és használatát a FreeBSD-vel.

23 fejezet: Honosítás, az I18N/L10N használata és beállítása

Bemutatja, hogyan használjuk a FreeBSD-t a rendszer és az alkalmazások szintjén az angoltól eltérő nyelveken.

24 fejezet: A FreeBSD frissítése és frissen tartása

Elmagyarázza, mik az alapvető különbségek a FreeBSD-STABLE, FreeBSD-CURRENT verziók, valamint a FreeBSD kiadások között. Bemutatja, mely felhasználók lehetnek azok, akik a legtöbbet tudnak profitálni egy fejlesztői rendszer használatából, illetve körvonalazza ennek folyamatát. Továbbá röviden összefoglalja azokat az eszközöket, amelyekkel a felhasználók frissíthetik a rendszerüket a biztonsági és kritikus hibák javításakor.

25 fejezet: DTrace

A Sun DTrace eszközének beállítását és használatát mutatja be. A segítségével megvalósított dinamikus nyomkövetéssel lehetőségünk nyílik valós idejű elemzéseken keresztül felderíteni a különböző teljesítménybeli problémákat.

26 fejezet: Soros vonali kommunikáció

Kifejti, hogyan csatlakoztassunk terminált vagy modemet a FreeBSD rendszerünkhöz, ha behívó vagy betárcsázós kapcsolatot szeretnénk létrehozni.

27 fejezet: A PPP és a SLIP

Bemutatja, miként tudjuk PPP-n, SLIP-en és Etherneten keresztüli PPP-vel (PPPoE) összekapcsolni a FreeBSD-t távoli rendszerekkel.

28. fejezet: Elektronikus levelezés

Megismerhetjük egy elektronikus levelező szerver különféle komponenseit, és elmélyedhetünk az egyik leghíresebb levelezőszerver-szoftver, a **sendmail** használatában és felületesebb konfigurálásában.

29. fejezet: Hálózati szerverek

Részletekbe menően és konfigurációs példákkal mutatja be, miként tudunk hálózati állományrendszer kiszolgálónak, névszervernek, hálózati információs rendszer kiszolgálónak vagy időszinkronizációs szervernek beállítani egy FreeBSD-s számítógépet.

30. fejezet: Tűzfalak

Kifejti a szoftveres tűzfalak mögött álló filozófiát, valamint részletesen tárgyalja a különböző, FreeBSD-n elérhető tűzfalak konfigurációját.

31. fejezet: Egyéb haladó hálózati témák

Feldolgoz számos hálózati témát, beleértve az internet kapcsolat helyi hálózaton (LAN-on) keresztül történő megosztását több számítógép között, haladó forgalomirányítási kérdéseket, vezeték nélküli hálózatok beállítását, Bluetooth, ATM, IPv6 és sok minden mással kapcsolatos információkat.

A Függelék: A FreeBSD beszerzése

Felsorolja azokat a forrásokat, ahonnan a FreeBSD CD-n vagy DVD-n beszerezhető, valamint azokat a honlapokat, ahonnan letölthető vagy telepíthető a FreeBSD.

B Függelék: Irodalomjegyzék

A könyv sok tekintetben olyan témákat is érint, amelyek felkelthetik az olvasó érdeklődését és ezek kapcsán bővebb magyarázatra vágyik. Az irodalomjegyzékben ezért összeírtunk számos remek könyvet, amelyekre hivatkozunk is a fejezetekben.

C Függelék: Erőforrások az interneten

Tartalmazza a FreeBSD felhasználók számára elérhető azon fórumokat, ahová beküldhetik kérdéseiket, illetve szakmai jellegű társalgásokat folytathatnak.

D Függelék: PGP-kulcsok

Az egyes FreeBSD fejlesztők PGP-kulcsait sorolja fel.

A könyvben alkalmazott konvenciók

A könnyebb és egységesebb olvashatóság kedvéért az alábbi konvenciókat igyekeztünk követni a könyvben.

Tipográfiai konvenciók

Dőlt

A *dőlt* betűket állománynevek, URL-ek, kiemelt szövegek és a szakmai kifejezések első előfordulásakor használjuk.

Írógépszerű

Az írógépszerű betűket hibaüzenetek, parancsok, környezeti változók, portok, számítógépek, felhasználók, csoportok, eszközök nevei, változók és kódrészletek esetén használjuk.

Félkövér

A **félkövér** betűket alkalmazások, parancsok és billentyűk megnevezésénél használjuk.

Felhasználói bevétel

A billentyűket **félkövérrrel** írjuk, hogy kiemelkedjenek a szöveg többi részéből. Az egyszerre megnyomni kívánt billentyűk kombinációját a ‘+’ jelöléssel adjuk meg, mint például:

Ctrl+Alt+Del

Ez azt jelenti, hogy a felhasználónak a **Ctrl**, **Alt** és **Del** billentyűket egyszerre kell lenyomnia.

Azokat a billentyűket, amelyeket egymás után kell lenyomni, vesszővel választjuk el, például:

Ctrl+X, Ctrl+S

Ez tehát azt jelenti, hogy a felhasználónak először a **Ctrl** és **X** billentyűket, majd a **Ctrl** és **S** billentyűket kell egyszerre lenyomnia.

Példák

A `E:\>` kijelzéssel kezdődő példák egy MS-DOS® parancsot jelölnek. Ha másképpen nem említjük, ezeket a parancsokat a modern Microsoft Windows-okban található “Parancssorból” kell kiadni.

```
E:\> tools\fdimage floppies\kern.flp A:
```

A `#` kijelzéssel kezdődő példák a FreeBSD-ben rendszeradminisztrátori jogokat igénylő parancsok kiadását jelentik. Ehhez bejelentkezhetünk a `root` felhasználóval, vagy felvethetjük a rendszeradminisztrátori jogokat a saját felhasználói fiókunkból a `su(1)` használatával is.

```
# dd if=kern.flp of=/dev/fd0
```

A `%` kijelzéssel kezdődő példák olyan parancsra utalnak, amelyeket egy normál felhasználói fiókból érdemes kiadni. Hacsak másképpen nem jelezzük, a C-shell szintaxisát használjuk a környezeti változók és egyéb parancsok megadásakor.

```
% top
```

Köszönetnyilvánítás

A könyv, amit itt most olvashatunk, több száz ember együttes munkájának eredménye a világ minden tájáról. Akár csak elgépeléseket javítottak, vagy komplett fejezeteket adtak hozzá, minden hozzájárulás hasznosnak bizonyult.

Emellett sok cég anyagilag is támogatta a könyv fejlődését, lehetővé téve ezáltal, hogy a szerzők teljes munkaidőben dolgozhassanak rajta, pénzt kapjanak az írásaikért stb. Leginkább a BSDi (amelyet később felvásárolt a Wind River Systems (<http://www.windriver.com>)) adott teljes munkaidős fizetést a FreeBSD Dokumentációs Projekt tagjainak a könyv gondozásához, amely végül az első nyomtatott kiadás megjelentetéséhez vezetett 2000 márciusában (ISBN 1-57176-241-8). A Wind River Systems ezt követően további szerzőket is finanszírozott a nyomtatási-szedési

infrastruktúra továbbfejlesztéséhez és a könyv tartalmának bővítéséhez. Ennek eredménye lett a második nyomtatott kiadás, amely 2001 novemberében jelent meg (ISBN 1-57176-303-1). 2003 — 2004 folyamán a FreeBSD Mall, Inc. (<http://www.freebsdmall.com>) támogatott anyagilag számos hozzájárulót a kézikönyvet illető munkájáért, a harmadik nyomtatott kiadásra történő előkészítésben.

I. Bevezetés

A FreeBSD kézikönyv ezen része azoknak a felhasználóknak és rendszergazdáknak szól, akik még nem ismerik a FreeBSD-t. A fejezetek:

- Bemutatják a FreeBSD-t.
- Végigvezetnek a telepítés folyamatán.
- Ismertetik a UNIX alapjait.
- Megmutatják, hogyan telepítsük a FreeBSD-hez elérhető megannyi külső alkalmazást.
- Megismerhetjük az X-et, a UNIX-os ablakozórendszert, és részleteiben is láthatjuk, miként konfiguráljunk be egy munkakörnyezetet, amellyel kényelmesebbé válik a munka.

A fejezetek megírása során arra törekedtünk, hogy minél kevesebb hivatkozást tegyünk a könyv később következő részeire, így ennek köszönhetően a kézikönyv ezen része anélkül olvasható, hogy közben folyamatosan előre-hátra kellene lapozgatnunk benne.

Fejezet 1. Bemutakozás

Átszerkesztette, átszervezte és bizonyos részeit átdolgozta: Jim Mock.

1.1. Áttekintés

Köszönjük, hogy érdeklődik a FreeBSD iránt! A fejezet a FreeBSD Projektet több különböző vonatkozásban mutatja be: a történetét, a céljait, a fejlesztési modelljét és így tovább.

A fejezet elolvasása során megismerjük:

- hogyan viszonyul a FreeBSD más operációs rendszerekhez;
- a FreeBSD Projekt történetét;
- a FreeBSD Projekt célkitűzéseit;
- a FreeBSD nyílt forráskódú fejlesztési modelljének alapjait;
- és természetesen: hogyan is keletkezett a “FreeBSD” név.

1.2. Üdvözljük a FreeBSD-ben!

A FreeBSD egy 4.4BSD-Lite alapú operációs rendszer Intel® (x86 és Itanium®), AMD64, Alpha™, Sun UltraSPARC® számítógépekre. Jelenleg is portolás alatt áll további architektúrákra. Olvashatunk a FreeBSD történetéről vagy éppen az aktuális kiadásáról. Ha szeretnénk hozzájárulni a Projekt fejlődéséhez (forráskód, hardver vagy pénz), olvassuk el a Hozzájárulás a FreeBSD-hez (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributing/index.html) című cikket (angolul).

1.2.1. Mire képes a FreeBSD?

A FreeBSD számos figyelemre méltó tulajdonságot tudhat magáénak. Ezek közül néhány:

•

A preemptív ütemezés dinamikusan szabályozható prioritások segítségével biztosítja a számítógép felhasználók és alkalmazások közti finom és igazságos megosztását, akár a legnagyobb terhelés esetén is.

•

Többfelhasználós rendszerként lehetővé teszi, hogy sokan tudják a FreeBSD-t egyszerre többféle dologra is használni. Például, ez azt jelenti, hogy a rendszerhez csatlakoztatott különböző perifériák, mint például a nyomtatók és szalagos egységek, megfelelően szétoszthatóak a felhasználók között vagy éppen a hálózaton, és az egyes erőforrásokhoz a felhasználók vagy azok egy csoportja csak korlátozott módon férhetnek hozzájuk, elkerülve ezzel a rendszer számára létfontosságú erőforrások túlterhelését.

•

A TCP/IP hálózati protokoll gyors és megbízható implementációja, illetve a legfontosabb ipari szabványok, mint az SCTP, DHCP, NFS, NIS, PPP, SLIP, IPsec és IPv6 támogatása. Ezáltal egy FreeBSD-s számítógép könnyedén képes együttműködni más rendszerekkel vagy akár vállalati szerverként is üzemelni. Megbirkózik az NFS

(Network File System, távoli állományelérés) és az elektronikus levelezés megszervezésével ugyanúgy, ahogy a vállalatunk internetes elvárásaival a WWW, FTP és forgalomirányítási protokollokon keresztül és tűzfal iránti (biztonsági) igényeivel is.

•

A *memóriavédelem* megvalósítása gondoskodik róla, hogy az alkalmazások (vagy a felhasználók) ne zavarják egymást. Az egyik alkalmazás összeomlása nincs kihatással a rendszerben futó összes többire.

- A FreeBSD egy *32 bites* operációs rendszer (az Alpha, Itanium, AMD64 és UltraSPARC architektúrákon pedig *64 bites*), amelyet már a kezdetektől fogva annak terveztek.

•

A *X Window System* ipari szabványa (X11R7) alapján szolgáltatja a grafikus felhasználói felületet (GUI) bármelyik VGA-kártyán és monitoron, illetve annak teljes forráskódja is elérhető.

•

Bináris szintű kompatibilitás a Linuxra, SCO-ra, SVR4-re, BSDI-re és NetBSD-re készített programok nagy részével.

- *Futtatásra kész* alkalmazások ezrei érhetőek el a FreeBSD *port-* és *csomaggyűjteményében*. Miért bújnánk az internetet értük, ha mindent egy helyen is megtalálhatunk?
- További *könnyen portolható* alkalmazások ezrei állnak rendelkezésre az interneten. A FreeBSD forráskódja kompatibilis a legtöbb elterjedt kereskedelmi UNIX rendszerével, aminek köszönhetően az alkalmazások nagy része csak kevés módosítást igényel a fordításhoz, már amennyiben erre egyáltalán szükség van.

•

Az igény szerinti lapozással működő *virtuális memória* és “egyesített VM/puffer gyorsítótár” úgy lett kialakítva, hogy hatékonyan kiszolgálja a nagyobb étvágyú alkalmazásokat, miközben a többi felhasználó számára továbbra is reakcióképes marad.

•

Az *SMP* támogatása a több processzorral rendelkező számítógépek számára.

•

C, C++ és Fortran fejlesztői eszközök széles tárháza használható. Kutatáshoz és fejlesztéshez más egyéb programozási nyelvek is elérhetőek a portok és csomagok segítségével.

•

Az egész rendszer *forráskódjának* megléte lehetővé teszi, hogy a legnagyobb fokú irányítást élvezhessük a környezetünk felett. Miért is bízunk magunkat egy zárt rendszert fejlesztő cégre, mikor lehetne egy igazán nyílt rendszerünk?

- Nagy mennyiségű *internetes dokumentáció*.
- *Még sok minden más!*

A FreeBSD Kaliforniai Egyetem (Berkeley) Számítógépes rendszerek kutatócsoportja által fejlesztett 4.4BSD-Lite kiadásán alapszik és ápolja a BSD-rendszerek fejlesztésének jellegzetes hagyományait. Túl a kutatócsoport kivételes munkáján, a FreeBSD Projekt több ezernyi órát szentelt arra, hogy a legtöbbet hozza ki a rendszerből mind a teljesítményt, mind pedig a valós életben felbukkanó terhelési helyzetekben történő helytállást illetően. Ahogy a legnagyobb piaci óriások igyekeznek egy hasonló képességű, teljesítményű és megbízhatóságú PC-s operációs rendszert kifejleszteni, úgy a FreeBSD már *most* felajánlja ezeket!

Kizárólag csak a képzeletünk szabhat gátat annak, hogy mire is tudjuk használni a FreeBSD-t. Szoftverfejlesztéstől kezdve, a gyári automatizáláson és készletnyilvántartáson át a műholdas antennák tájolásáig szinte mindenre: ha ezt eddig egy kereskedelmi UNIX-szal is meg tudtuk tenni, akkor nagyon valószínű, hogy a FreeBSD-vel is képesek leszünk erre! A FreeBSD ezen felül nagyban profitál a világban található különböző kutatóközpontok és egyetemek által fejlesztett, kiváló minőségű alkalmazások ezreiből, melyek gyakorta olcsón vagy ingyen elérhetőek. Kereskedelmi alkalmazások is egyre nagyobb számban képviseltetik magukat minden nap.

Mivel a FreeBSD forráskódja általánosan elérhető, a rendszer szinte tetszőleges mértékben testreszabható a különleges elvárásokat támaztató alkalmazások vagy projektek számára. Ez a nagyobb kereskedelmi fejlesztők operációs rendszereivel majdnem teljesen elképzelhetetlen. Íme csupán néhány példája azon alkalmazásoknak, melyek jelenleg is FreeBSD-t használnak:

- *Internetes szolgáltatások:* A FreeBSD-be épített szilárd TCP/IP alapú hálózatkezelés különféle internetes szolgáltatások számára teszi ideális platformmá:

- FTP szerverek
- World Wide Web szerverek (hagyományos vagy biztonságos [SSL])
- IPv4 és IPv6 forgalomirányítás
- Tűzfalak és NAT ("IP maszkolás"), átjárók
- Elektronikus levelező szerverek
- USENET hírrendszer és üzenőfal
- Sok minden más...

A FreeBSD használatához kezdetben elegendő egy olcsó 386-os PC, melyet a vállalkozásunk fejlődésével szépen fel tudunk hozni egy RAID-dal ellátott négyprocesszoros Xeon rendszerig.

- *Oktatás:* Esetleg informatikával vagy műszaki informatikával foglalkozik? Nem is lehetne jobban a FreeBSD által felkínált élményeken kívül máshogy megismerkedni elsőkézből az operációs rendszerek, számítógépes architektúrák és hálózatok működésével! Rengeteg szabadon használható műszaki, matematikai és grafikai tervező programcsomag könnyíti meg azok munkáját is, akik számára a számítógép legfőképpen *más* feladatok elvégzésére hivatott!
- *Kutatás:* Miután a teljes FreeBSD rendszer forráskódja bárki számára elérhető, tökéletes kiindulási pontot ad az operációs rendszerek témakörében vagy a számítástudomány egyéb ágaiban végzendő kutatásokhoz. A FreeBSD nyílt természete ezenkívül lehetővé teszi egymástól távol levő csoportok közös együttműködését is anélkül, hogy a résztvevőknek aggódnia kellene a különleges licencszerződések vagy a nyílt fórumokon felmerülő korlátozások miatt.

Hálózatépítés: Szüksége van egy új útválasztóra? Esetleg egy névszerverre (DNS)? Egy tűzfalra, mely távoltartja a nemkívánatos egyéneket a belső hálózattól? A FreeBSD pillanatok alatt átváltoztatja a sarokban porosodó 386-os vagy 486-os PC-nket egy kifinomult csomagszűrési képességekkel bíró forgalomirányító eszközzé.

-

X Window munkállomás: A FreeBSD a szabadon használható X11 szerverrel együtt remek választás egy olcsó X terminál kiépítéséhez. Eltérően egy szokványos X termináltól, a FreeBSD azonban igény szerint sok alkalmazás helyi futtatását is képes megoldani, ezzel megszabadítva minket a központi szerver használatának kényszerétől. A FreeBSD viszont akár “lemez nélkül” is el tud indulni, aminek révén az egyes munkaállomások karbantartása még olcsóbbá és könnyebbé válik.

-

Szoftverfejlesztés: Az alap FreeBSD rendszer fejlesztőeszközök tömkelegével, többek közt a híres GNU C/C++ fordítóval és nyomkövetővel érkezik.

A FreeBSD CD-n, DVD-n és FTP-n keresztül elérhető forráskód és bináris formátumban is. A FreeBSD beszerzésével kapcsolatos bővebb információkért olvassuk el az A Függelékét.

1.2.2. Ki használja a FreeBSD-t?

A FreeBSD egyaránt remek eszköz- és termékfejlesztői platformként funkcionál a világ legnagyobb informatikai cégeinél, többek közt:

-

Apple (<http://www.apple.com>)

-

Cisco (<http://www.cisco.com>)

-

Juniper (<http://www.juniper.net>)

-

NetApp (<http://www.netapp.com>)

A FreeBSD mindezek mellett több nagyobb internetes oldal alapját képezi, mint például:

-

Yahoo! (<http://www.yahoo.com/>)

-

Yandex (<http://www.yandex.ru/>)

-

Apache (<http://www.apache.org/>)

-

Rambler (<http://www.rambler.ru/>)

-

Sina (<http://www.sina.com/>)

-

Pair Networks (<http://www.pair.com/>)

•

Sony Japan (<http://www.sony.co.jp/>)

•

Netcraft (<http://www.netcraft.com/>)

•

NetEase (<http://www.163.com/>)

•

Weathernews (<http://www.wni.com/>)

•

TELEHOUSE America (<http://www.telehouse.com/>)

• Experts Exchange (<http://www.experts-exchange.com/>)

és még sokan mások.

1.3. A FreeBSD Projektről

A most következő rész egy-két háttérinformációt tár fel a Projektről, többek között a történetét, céljait és a benne alkalmazott fejlesztési modellt.

1.3.1. A FreeBSD rövid története

Írta: Jordan Hubbard.

A FreeBSD Projekt valamikor 1993 kezdetéről eredeztethető, és részben a “Nem hivatalos 386BSD Patchkit”-ből nőtt ki, a patchkit 3 legutolsó koordinátorának, Nate Williamsnek, Rod Grimesnek és nekem köszönhetően.

Eredeti célunk a 386BSD köztes állapotainak rögzítése lett volna, amittől olyan problémák megoldását reméltük, melyeket a patchkitek gyártása önmagában egyszerűen nem tudott megoldani. Néhányan még talán emlékeznek is a Projekt kezdeti munkaneveire: “386BSD 0.5” vagy “386BSD Interim”, melyek pontosan erre a tényre hivatkoztak.

A 386BSD eredetileg Bill Jolitz operációs rendszere volt, amely ennél a pontnál már közel egy éve senki sem tartott karban. Mivel a hozzá tartozó patchkit pedig napról napra duzzadt, egyre kényelmetlenebbé vált a karbantartása. Ezért egyhangúan úgy döntöttünk, segítünk Billnek azzal, hogy időnként létrehozunk egy “letisztított” változatot. Ez a próbálkozásunk csúnyán kudarcba fulladt, amikor Bill Jolitz hirtelen meggondolta magát és visszalépett a Projekt támogatásától. Semmilyen egyértelmű útmutatást nem adott arra, hogy mit csináljunk helyette.

Nem tartott sokáig eldönteni, hogy ez a cél továbbra is megéri a fáradságot, még Bill segítsége nélkül is, ezért felvettük a “FreeBSD” nevet, melyet David Greenmannek köszönhetünk. Kezdeti feladatainkat a rendszer akkori felhasználóival tartott egyeztetések után állítottuk fel. Miután teljesen tisztán láthatóvá vált, hogy a Projekt a megvalósulás útján van, felvettem a kapcsolatot a Walnut Creek-kel, terjesztési mód után nézve azok számára, akik nem tudtak akkoriban könnyedén hozzáférni az internethez. A Walnut Creek nem csak támogatta a FreeBSD CD-n történő terjesztését, hanem még egy számítógépet és egy gyors internetkapcsolatot is a Projekt rendelkezésére bocsátott. A Walnut Creek szinte példátlan mértékű, egy akkoriban teljesen ismeretlen projektbe vetett hite nélkül nagyon nehezen lenne elképzelhető, hogy a FreeBSD olyan messzire és olyan gyorsan jutott volna el, ahol ma tart.

Az első CD-lemezen (és széles körben az interneten is megjelenő) változat a FreeBSD 1.0 volt, amely 1993 decemberében jelent meg. A Berkeley-ről származó 4.3BSD-Lite ("Net/2") szalagokon található források alapján készült, kiegészítve a 386BSD-ből és a Szabad Szoftver Alapítványtól (Free Software Foundation, FSF) származó komponensekkel. Első kiadásként igen méltányos sikert könyvelhetett el, melyet a még inkább sikeres FreeBSD 1.1-gyel folytattunk 1994 májusában.

Nagyjából ekkortájt néhány váratlan sötét felhő bukkant fel az égbolton, ahogy a Novell és a Berkeley hosszantartó pereskedése lezárult a Berkeley Net/2 szalagjainak jogi formáját illetően. Ennek eredményeképpen a Berkeley elfogadta, hogy a Net/2 nagy része "jelzáloggal terhelt" és a Novell tulajdona, aki pedig valamivel korábban az AT&T-től szerezte. Ezért cserébe a Berkeley megkapta a Novell "áldását" a 4.4BSD-Lite kiadásra, és amikor az véglegesen kijön, megszűnik a rajta levő jelző. Emiatt az összes Net/2 felhasználónak erősen javasolt volt váltani. Ez érintette magát a FreeBSD-t is, és így a Projekt 1994 júliusáig kapott határidőt, hogy leállítsa a Net/2 alapú termékeinek szállítását. A megegyezés értelmében a Projekt kiadhatott még egy utolsó kiadást a határidő előtt, amely végül a FreeBSD 1.1.5.1 lett.

A FreeBSD-nek ekkor szembesülnie kellett azzal a nehéz feladattal, hogy lényegében újra fel kellett találnia magát, a teljesen új és meglehetősen hiányos 4.4BSD-Lite bitjeitől elindulva. A "Lite" (egyszerűsített) kiadások abban az értelemben számítottak egyszerűbbnek, hogy a Berkeley kutatói (a különböző jogi követelések miatt) eltávolították a ténylegesen beindítható rendszerhez szükséges programrészek nagyobb részét, ill. a 4.4-es verzió Intel processzorokra készített portja nagyon is befejezetlen volt. A Projektnek egészen 1994 novemberéig tartott, hogy megtegye ezt a lépést, ugyanis ekkor jelent meg a FreeBSD 2.0 az interneten és (december vége felé) CD-n. Annak ellenére, hogy még némileg érdes maradt bizonyos helyeken, ez a kiadás jelentős sikereket ért el. Ezt követte 1995 júniusában a sokkalta stabilabb és könnyebben telepíthető FreeBSD 2.0.5.

A FreeBSD 2.1.5-öt 1996 augusztusában adtuk ki, mely akkora népszerűségnek örvendett az internet-szolgáltatók és kereskedelmi közösségek körében, hogy a 2.1-STABLE elágazásból egy újabb kiadást készítettünk. Ez volt a FreeBSD 2.1.7.1, amely 1997 februárjában jelent meg és ezzel együtt a 2.1-STABLE fejlesztését is zárta. Most már csak karbantartást végzünk rajta, és csak a biztonsági és egyéb kritikus hibajavítások kerülnek bele (RELENG_2_1_0).

A FreeBSD 2.2 fejlesztése 1996 novemberében ágazott le az akkori fejlesztői ("-CURRENT") ágból, mint a RELENG_2_2-es ág. Ebből az első teljes kiadás (2.2.1) 1997 áprilisában jelent meg. A 2.2-es ág mentén további kiadások 1997 nyarán és őszén készültek, melyek közül az utolsó (2.2.8) 1998 novemberében jelent meg. Az első hivatalos 3.0-ás kiadás 1998 októberében jött ki, ami egyúttal a 2.2-es ág befejezésének kezdetét jelentette.

A fejlesztési fa 1999. január 20-án került ismét elágaztatásra, melynek eredménye a 4.0-CURRENT és 3.X-STABLE ágak lettek. A 3.X-STABLE ágban a 3.1 1999. február 15-én, a 3.2 1999. május 15-én, a 3.3 1999. szeptember 16-án, a 3.4 1999. december 20-án és a 3.5 2000. június 24-én jelent meg, melyet pár nappal később egy kisebb alverzió, a 3.5.1 követett, a Kerberosra vonatkozó friss biztonsági javításokkal. Ez lett egyben a 3.X ág utolsó kiadása.

Egy másik fontos elágaztatás 2000. március 13-án történt, mellyel életre kelt a 4.X-STABLE ág. Ebből aztán számos kiadás született: a 4.0-RELEASE 2000 márciusában mutatkozott be, az utolsó 4.11-RELEASE pedig 2005 januárjában látott napvilágot.

A várva várt 5.0-RELEASE 2003. január 19-én került bejelentésre. Közel háromévnnyi munka eredményeképpen ez a kiadás indította meg a FreeBSD-t a többprocesszoros rendszerek és az alkalmazások szálkezelésének fejlettebb támogatásának útján, valamint az UltraSPARC és ia64 platformok támogatása is itt jelent meg először. Ezt a kiadást az 5.1 követte 2003 júniusában. A hozzá tartozó -CURRENT ágból az utolsó kiadás az 5.2.1-RELEASE volt, amely 2004 februárjában mutatkozott be.

A 2004 augusztusában, a RELENG_5 ág létrehozását a 5.3-RELEASE követte, és egyben a 5-STABLE ág kezdetét is jelezte. A legújabb 5.5-RELEASE 2006 májusában jött ki. A RELENG_5 ágból már nem fog készülni több kiadás.

A fejlesztési fa ezután 2005 júliusában ágazott el ismét, ezúttal a RELENG_6 ágnak adott életet. A 6.0-RELEASE az 6.X ág első kiadásaként 2005 novemberében jelent meg. A legújabb 6.4-RELEASE 2008 november hónapjában jelentkezett. A RELENG_6 ágból már nem készülnek további kiadások.

A RELENG_7 ág 2007 októberében jött létre. Ebből az első kiadás 2008 februárjában a 7.0-RELEASE volt. A legfrissebb 7.3-RELEASE kiadás 2010 március hónapban készült el. A RELENG_7 ágból további kiadások is várhatóak.

A fejlesztési fából 2009 augusztusában ismét levált egy ág, amely ezúttal a RELENG_8 volt. A 8.0-RELEASE, a 8.X ág első kiadása 2009 novemberében jelent meg. A legfrissebb 8.1-RELEASE 2010 július hónapban jött ki. A RELENG_8 ágból várhatóak további kiadások.

Jelen pillanatban a hosszabb távú fejlesztések a 9.X-CURRENT (törzs) ágban kapnak helyet, és a 9.X-ből készült időközönkénti pillanatkiadások folyamatosan elérhetőek CD-n (és természetesen interneten keresztül is) a pillanatkiadásokat tároló szerverről (<ftp://current.FreeBSD.org/pub/FreeBSD/snapshots/>).

1.3.2. A FreeBSD Projekt céljai

Írta: Jordan Hubbard.

A FreeBSD Projekt célja, hogy olyan szoftvereket kínáljon, amelyek tetszőlegesen, bármilyen célra felhasználhatóak, mindenféle megkötések nélkül. Sokunk jelentős energiát fektet a programokba (és a Projektbe) és minden bizonnyal egyikünk sem utasítana vissza semmilyen anyagi ellenszolgáltatást se most, se később, de egyáltalán nem ragaszkodunk hozzá. Hisszük, hogy elsődleges “küldetésünk” olyan programok és programrészek készítése bárki számára és bármilyen célra, melyeket a lehető legszélesebb körben alkalmaznak és a lehető legtöbb hasznot hajtják. Ez, úgy érzem, az egyik legalapvetőbb célja a szabad szoftvereknek, és ez az, amit mi is lelkesen magunkénak vallunk.

A forrásfánkban található GNU General Public License (GPL) vagy a Library General Public License (LGPL) alá eső kódok hozzáférhetőségére ezzel szemben némileg több megszorítás vonatkozik, legalább is inkább ami a hozzáférhetőséget illeti. Mivel a GPL-es szoftverek kereskedelmi használata további bonyodalmakat vethet fel, ha lehetőségünk adódik rá, inkább a sokkal enyhébb BSD licenccel rendelkező szoftvereket választjuk.

1.3.3. A FreeBSD fejlesztési modellje

Írta: Satoshi Asami.

A FreeBSD fejlesztése egy nagyon nyitott és rugalmas folyamat, szó szerint a világ minden tájáról érkező többszáznyi segítségéből építkezik, ahogy az látható is a résztvevőink listáján

(http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/article.html). A FreeBSD fejlesztési infrastruktúrája lehetővé teszi, hogy ez a többszáznyi résztvevő az interneten keresztül működjön együtt.

Folyamatosan várjuk az új fejlesztőket és ötleteket, és mindazok, akik komolyabban érdeklődnek a Projekt iránt, egyszerűen felvehetik velünk a kapcsolatot a FreeBSD technical discussions levelezési lista

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>) címén. Egy FreeBSD announcements levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce>) is elérhető azok számára, akik értesíteni kívánják a többi FreeBSD felhasználót munkájuk főbb eredményeiről.

A FreeBSD Projektről és annak fejlesztési modelljéről hasznos tudni az alábbiakat, függetlenül attól, hogy egyedül vagy másokkal szoros együttműködésben dolgozunk:

Az SVN és CVS repositoryk

Sok éven keresztül a FreeBSD központi forrásfáját CVS (<http://ximbiot.com/cvs/wiki/>)-en (Concurrent Versions System) keresztül tartották karban, amely egy, a FreeBSD-vel is érkező, szabadon elérhető verziókezelő rendszer. 2008 júniusában a Projekt az SVN (<http://subversion.tigris.org>) (Subversion) használatára váltott. Ez a váltás szükségszerű volt, mivel a CVS által okozott technikai nehézségek gyorsan előjöttek a forrásfa és a hozzá tartozó metainformációk szapora növekedésével. Noha a központi repository most már **SVN**-alapú, a kliensoldali **CVSup** és **csup** alkalmazások továbbra is a korábbi infrastruktúrával dolgoznak, ahogy eddig is — az **SVN** repositoryban végzett változtatások ehhez automatikusan átkerülnek **CVS** alá. Jelen pillanatban egyedül csak a központi forrásfa használja ezt a megoldást, a dokumentáció, a weboldalak és a Portgyűjtemény forrásai továbbra is **CVS** alól üzemelnek. Az elsődleges CVS repository (<http://www.FreeBSD.org/cgi/cvsweb.cgi>) egy Santa Clara-i (California, USA) számítógépen található, ahonnan a világban található rengeteg tükörszerverre másolódik. Az **SVN**-fa, mely tartalmazza a **-CURRENT** és **-STABLE** ágakat, könnyen lemásolható a saját számítógépünkre is. Ennek részleteiről bővebben a **A forrásfa szinkronizálása** c. szakaszban olvashatunk.

A committerek listája

A hivatalos fejlesztők (*committerek*) azok az emberek, akik a CVS-fához írási joggal rendelkeznek, tehát módosítást hajthatnak végre a FreeBSD forrásaiban (a “committer” kifejezés a `cvs(1) commit` parancsból származik, amelyet arra használunk, hogy felvigyük a módosításainkat a CVS repository-ba). Javaslatainkat legjobban a `send-pr(1)` használatával tudjuk a committerek elé tární. Ha valamiért ez mégsem működne, megpróbálhatjuk őket elérni közvetlenül a FreeBSD committer’s mailing list címére küldött e-maillal.

A FreeBSD Core Team

Ha a FreeBSD Projekt egy vállalat lenne, akkor a *FreeBSD Core Teamje* (irányító csoportja) foglalná magában a vezetőséget. Ennek a csoportnak elsődleges feladata, hogy fenntartsa a Projekt egészének kondícióját és gondoskodjon róla, hogy a megfelelő irányba haladjon. Az irányító csoportnak ugyanígy feladata a megbízható és odaadó committerek tömörítése és az új tagok beszerzése, ha a csoportból kilépne valaki. A jelenlegi Core Team tagjait 2008 júliusában választották meg. A választásokat kétfévente tartják.

Ebben a csoportban egyes tagoknak ezenfelül még bizonyos területekre felügyelniük is kell. Ez azt jelenti, hogy felelősek a rendszer valamelyik nagyobb részének az előírásoknak megfelelő működéséért. A FreeBSD fejlesztők teljes felsorolása és a hozzájuk tartozó területek megtalálhatóak A résztvevők listájában (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/article.html).

Megjegyzés: A Core Team legtöbb tagja pusztán önkéntesen vesz részt a FreeBSD fejlesztésében és nem származik a projektből semmilyen anyagi haszna. Emiatt a “részvétel” nem tévesztendő össze a “garantált támogatással”. A “vezetőségre” vonatkozó hasonlat nem teljesen pontos abban az értelemben, hogy ezek az emberek tulajdonképpen egy kívülálló szempontjából ésszerűtlen döntést hoztak azzal, hogy a FreeBSD támogatására áldozták az életüket!

Külső résztvevők

Végül, de nem utoljára, következzen a fejlesztők legnagyobb csoportja: ők maguk a felhasználók, akik rendszeres visszajelzéseket és hibajavításokat küldenek. A FreeBSD kevésbé központosított fejlesztésében elsősorban a FreeBSD technical discussions levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>) segítségével lehet felvenni a fonalat, ahol ezeket a témákat tárgyalják meg. A FreeBSD-hez kapcsolódó különféle levelezési listákról többet a C Függelékben olvashatunk.

A FreeBSD résztvevőinek listája

(http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/article.html) hosszú és még most is növekszik; miért nem próbálunk mi is visszaadni valamit a FreeBSD-nek?

Nem csak programozással lehet segíteni a Projektet: a megoldandó feladatok listáját megtalálhatjuk a FreeBSD Projekt honlapján (<http://www.FreeBSD.org/index.html>).

Röviden összefoglalva, a fejlesztési modellünk egymáshoz lazán kapcsolódó koncentrikus körökként szerveződik. Ez a központosított modell a FreeBSD-felhasználók kényelmét szolgáló lett kialakítva, akik így könnyedén tudnak követni egyetlen központi kódbázist, azonban megvan a lehetőségük a részvételre is! Minden vágyunk egy olyan megbízható operációs rendszer kialakítása, amihez nagy mennyiségű könnyen telepíthető és használható alkalmazás tartozik — ez a modell ennek elérésére nagyon is megfelelő.

A haladás ütemének fenntartása érdekében mindössze csak annyit kérünk a leendő FreeBSD fejlesztőinktől, hogy legyenek legalább annyira elszántak, mint a jelenlegi tagjaink!

1.3.4. Az aktuális FreeBSD kiadások

A FreeBSD egy szabadon elérhető, teljes forráskóddal érkező 4.4BSD-Lite alapú kiadás Intel i386™, i486™, Pentium®, Pentium Pro, Celeron®, Pentium II, Pentium III, Pentium 4 (vagy azzal kompatibilis), Xeon™, DEC Alpha és Sun UltraSPARC alapú számítógépekre. Elsősorban a Berkeley Számítógépes rendszerek kutatócsoportjának szoftverein alapszik, számos javítással a NetBSD, OpenBSD, 386BSD és a Szabad Szoftver Alapítvány munkásságának köszönhetően.

A FreeBSD 2.0 1994 végi megjelenése óta a FreeBSD teljesítménye, megbízhatósága és tudása drasztikusan megnövekedett. A legnagyobb változtatás az újjáalakított, összevont VM/állomány puffer gyorsítótárral rendelkező virtuális memória alrendszer, amely nem csak a teljesítményt növeli, hanem csökkenti a FreeBSD memóriaigényét is, jobban elfogadhatóvá téve ezzel az 5 MB-os minimumot. A további fejlesztések között találjuk a teljes NIS szerver és kliens támogatást, az átviteli TCP támogatását, az igény szerint tárcsázó PPP-t, a beépített DHCP támogatást, a továbbfejlesztett SCSI alrendszert, az ISDN támogatást, az ATM, FDDI, Fast és Gigabit Ethernet (1000 Mbit) hálózati csatlók támogatását, a legfrissebb Adaptec gyártmányú vezérlők fejlesztett támogatását és a többeszerinti hibajavítást.

Az alapeszközök mellé a FreeBSD felkínálja többeszerinti ismert és keresett program portjaiból álló gyűjteményét. Ebben a pillanatban is már több, mint 20 000 port érhető el! A portok listája a HTTP (WWW) szerverektől, a játékokon, nyelveken és sok mindenen keresztül a szövegszerkesztőkig terjed. Az egész Portgyűjtemény közelítőleg 417 MB tárhelyet kíván, minden portot az eredeti forráshoz viszonyított "különbségként" tárol. Ennek következtében a portok frissítése sokkal könnyebb és nagyban csökkenti a korábbi, 1.0-ás Portgyűjteménynél kialakult tárigényeket. Egy port lefordításához egyszerűen csak be kell lépni a telepíteni kívánt program könyvtárába és ki kell adnunk a `make install` parancsot, a többit a rendszer elvégzi. Minden egyes telepítendő port teljes forrása dinamikusan vagy CD-ről vagy pedig FTP-n keresztül töltődik le, így csak a ténylegesen telepítendő lefordításához elegendő

tárhelyre van szükség. Majdnem mindegyik port elérhető előre lefordított “csomag” formájában azok számára, akik nem kívánják lefordítani a portokat, és melyeket egy egyszerű parancs (`pkg_add`) segítségével telepíteni is tudják. A csomagokról és portokról a 4 fejezetben tudhatunk meg többet.

A FreeBSD telepítéséről és használatáról most már számos további nagyon hasznos dokumentumot találhatunk bármelyik FreeBSD-s számítógép `/usr/share/doc` könyvtárában. A helyileg telepített kézikönyveket bármilyen HTML-t megjeleníteni képes böngészővel el tudjuk olvasni az alábbi URL-eken:

A FreeBSD kézikönyv

`/usr/share/doc/handbook/index.html`

A FreeBSD GYIK

`/usr/share/doc/faq/index.html`

Az aktuális (leginkább frissített) verziók megtekinthetők a <http://www.FreeBSD.org/> címen.

Fejezet 2. A FreeBSD telepítése

Átszervezte, átrendezte és egyes részeit átdolgozta: Jim Mock. A sysinstall bemutatása, ábrái és bemásolása: Randy Pratt.

2.1. Áttekintés

A FreeBSD telepítéséhez egy könnyen használható szöveges telepítőprogram, a **sysinstall** használható. Ez a FreeBSD alapértelmezett telepítőprogramja, habár ezt a különféle gyártók kedvük szerint lecserélhetik. Ebben a fejezetben bemutatjuk a FreeBSD **sysinstall** segítségével történő telepítést.

A fejezet elolvasása során megismerjük:

- hogyan készítsünk telepítőlemezeket a FreeBSD-hez;
- a FreeBSD miként hivatkozza és osztja fel a merevlemezeinket;
- hogyan indítsuk el a **sysinstall** programot;
- milyen kérdéseket tesz fel nekünk a **sysinstall**, mire gondol, hogyan is kell azokat megválaszolni.

A fejezet elolvasásához ajánlott:

- a telepítendő FreeBSD verzióhoz tartozó támogatott hardvereket felsoroló lista átolvasása és benne a saját hardvereszközeink megkeresése.

Megjegyzés: Általánosan elmondható, hogy a most következő telepítési utasítások az i386 ("PC kompatibilis") architektúrájú számítógépekre vonatkoznak. Ahol erre szükség van, ott más platformokra vonatkozó utasítások is szerepelhetnek. Habár ezt a leírást igyekszünk a lehető legjobban naprakészen tartani, elképzelhető, hogy felfedezhetünk kisebb eltéréseket a telepítőben és az itt leírtak közt. Ezért ezt a fejezetet inkább egy általános útmutatónak javasoljuk, nem pedig egy szó szerint értelmezendő kézikönyvként.

2.2. Hardverkövetelmények

2.2.1. Minimális konfiguráció

A FreeBSD telepítéséhez szükséges minimális konfiguráció FreeBSD verzióként és architektúráként eltérő.

A minimális konfigurációt a FreeBSD honlapján a kiadásokról szóló oldalon (<http://www.FreeBSD.org/releases/index.html>), az "Installation Notes" részben találhatjuk meg. Ezt a következő szakaszokban foglaljuk össze. A FreeBSD telepítésének módszerétől függően szükségünk lehet egy hajlékonylemezre (floppy) vagy CD-ROM meghajtóra, esetleg egy hálózati kártyára. Ezt a 2.3.7 Szakaszban tárgyaljuk.

2.2.1.1. FreeBSD/i386 és FreeBSD/pc98

A FreeBSD/i386 és FreeBSD/pc98 egyaránt egy 486 vagy jobb processzort és legalább 24 MB memóriát igényel. A legkisebb telepítéshez legalább 150 MB szabad lemezterület szükséges.

Megjegyzés: Régebbi konfigurációk esetén nem egy gyorsabb processzor, hanem inkább több memória beszerzése, illetve több lemezterület felszabadítása a fontosabb.

2.2.1.2. FreeBSD/alpha

Megjegyzés: Az Alpha támogatás a FreeBSD 7.0 beindulásával eltávolításra került. A FreeBSD 6.x sorozat az utolsó, amely valamilyen támogatást ajánl ehhez az architektúrához. Ezzel kapcsolatban részletesebben a kiadásokkal (<http://www.FreeBSD.org/releases/index.html>) kapcsolatos információkat tartalmazó oldalon olvashatunk a FreeBSD honlapján.

2.2.1.3. FreeBSD/amd64

Két típusú processzor képes futtatni a FreeBSD/amd64 verzióját. Az első ezek közül az AMD64 processzorok, beleértve az AMD Athlon™64, AMD Athlon64-FX, AMD Opteron™ vagy újabb processzorokat.

A FreeBSD/amd64 verzióját kihasználni képes processzorok másik csoportja az Intel EM64T architektúrájára épülő processzorok. Ilyen processzor például az Intel Core™ 2 Duo, Quad és Extreme processzorcsaládok, valamint az Intel Xeon 3000, 5000 és 7000 sorozatszámú processzorai.

Ha nVidia nForce3 Pro-150 alapú géppel rendelkezünk, ki *kell* kapcsolnunk a BIOS-ban az IO APIC használatát. Ha nem találunk ilyen beállítást, akkor helyette magát az ACPI-t kell kikapcsolnunk. A Pro-150 chipsetnek vannak bizonyos hibái, amelyekre eddig még nem sikerült megfelelő megoldást találnunk.

2.2.1.4. FreeBSD/sparc64

A FreeBSD/sparc64 telepítéséhez egy támogatott platformra van szükségünk (lásd: 2.2.2 Szakasz).

A FreeBSD/sparc64 telepítéséhez egy egész lemezre lesz szükségünk, mivel a rendszer jelenleg nem képes megosztani azt más operációs rendszerekkel.

2.2.2. Támogatott hardverek

A FreeBSD minden kiadásához mellékelik a támogatott hardverek listáját "FreeBSD Hardware Notes" címmel. Ez a dokumentum többnyire a `HARDWARE.TXT` nevű állomány, amelyet a rendszer CD-n vagy FTP-n keresztül elérhető változatának gyökerében vagy a **sysinstall** dokumentációkat tartalmazó menüjében találhatunk meg.

2.3. A telepítés előtt elvégzendő feladatok

2.3.1. Készítsünk leltárt a számítógépünkről

A FreeBSD telepítése előtt érdemes összeszedni, pontosan mi minden is található a számítógépünkben. A FreeBSD telepítőrutinjai mutatni fogják a különböző komponensek (merevlemezek, hálózati kártyák, CD-meghajtók és a többi) modelljét és gyártóját. A FreeBSD ezenkívül megpróbálja kideríteni a megjelenő eszközök pontos konfigurációját is, beleértve a használt IRQ és IO portok kiosztását. A PC-s hardverek különféle szeszélyei miatt azonban ez az iménti folyamat nem minden esetben megbízható, ezért előfordulhat, hogy helyesbíteni kell a FreeBSD által megállapított értékeket.

Ha már van a gépünkön egy másik operációs rendszer, például Windows vagy Linux, akkor mindenképpen hasznos lehet az általa felkínált eszközökkel lekérdezni a hardvereink beállításait. Ha nem lennénk biztosak benne, hogy az adott bővítőkártyákat pontosan milyen beállításokkal is használjuk, nézzük meg ezeket magán a kártyán. A népszerű IRQ értékek általában a 3, 5 és 7, valamint az IO portok számát általában tizenhatos számrendszerben szerepeltetik, például 0x330.

Javasoljuk, hogy nyomtassuk ki vagy írjuk le ezeket a paramétereket a FreeBSD telepítése előtt. Ehhez rendezzük ezeket egy táblázatban, valahogy így:

Táblázat 2-1. Példa egy eszközelejtárra

Eszköz neve	IRQ	IO portok	Megjegyzés
Első merevlemez	-	-	Mérete 40 GB, gyártmánya Seagate, elsődleges IDE master
CD-ROM meghajtó	-	-	Elsődleges IDE slave
Második merevlemez	-	-	Mérete 20 GB, gyártmánya IBM, másodlagos IDE master
Első IDE vezérlő	14	0x1f0	
Hálózati kártya	-	-	Intel 10/100
Modem	-	-	3Com® 56K-s faxmodem, COM1
...			

Ahogy elkészítettük a számítógépünk alkatrészeit tartalmazó listát, vessük ezeket össze a telepítendő FreeBSD kiadás által megkövetelt eszközökkel.

2.3.2. Mentsük le az adatainkat

Amennyiben a FreeBSD telepítéséhez használt számítógép számunkra értékes adatokat tárol, igyekezzünk lementeni ezeket, és a FreeBSD tényleges telepítése előtt győződjünk is meg róla, hogy a mentés sikeres volt. A FreeBSD telepítőrutinjai természetesen megerősítést fognak kérni bármilyen adat lemezre írása előtt, azonban ha egyszer már elindítottuk a folyamatot, már semmit sem tudunk visszafordítani.

2.3.3. Döntsük el a FreeBSD telepítésének helyét

Ha a FreeBSD telepítéséhez az egész merevlemezünket fel akarjuk használni, akkor még nincs miért izgatnunk magunkat — nyugodtan átléphetjük ezt a szakaszt.

Amikor viszont a FreeBSD-t más operációs rendszerek mellé szeretnénk telepíteni, ismernünk kell, miként is helyezkednek el az adatok a lemezeken, és hogy ez miként is érint bennünket.

2.3.3.1. A lemezek kiosztása a FreeBSD/i386 esetén

A PC-k által használt lemezek különálló darabokra tagolhatóak. Ezeket a darabokat *partícióknak* nevezzük. Mivel azonban a FreeBSD maga is tárol partíciókat, ezért ez az elnevezés pillanatok alatt megtévesztővé válhat, ezért ezeket a lemezdarabokat a FreeBSD lemezslice-oknak vagy egyszerűen csak slice-oknak hívja. Például a PC-s lemezipartíciókkal dolgozó, `fdisk` nevű FreeBSD-s segédprogram partíciók helyett is slice-okra hivatkozik. A PC lemezenként alapvetően csak négy partíciót enged meg. Ezeket a partíciókat nevezik *elsődleges partícióknak*. Ettől a korlátozástól egy új típus, a *kiterjesztett partíció* létrehozásával szabadultak meg, amivel így négynél több partíció is készíthető. Lemezenként egyetlen ilyen kiterjesztett partíció található, de ezen belül speciális, ún. *logikai partíciók* hozhatóak létre.

Minden partíciónak van egy *partíció-azonosítója*, melyet a partíción található adatok típusának megállapítására használnak. A FreeBSD partícióinak azonosítója a 165.

Általánosságban véve minden operációs rendszer így azonosítja a partíciókat. Például a DOS és annak leszármazottai, mint például a Windows, minden elsődleges és logikai partícióhoz egy `C:-`-től induló *meghajtó-betűjelet* társít.

A FreeBSD-t egy elsődleges partícióra kell telepíteni. A FreeBSD az összes adatát, beleértve minden általunk létrehozott állományt is, ezen az egyetlen partíción fogja elhelyezni. Ha viszont több lemezünk van, többen is, vagy akár mindegyiken létrehozhatunk FreeBSD-s partíciókat. A FreeBSD telepítések során azonban legalább egy ilyen partíciónak használhatónak kell lennie. Ez lehet előre megtisztított üres partíció is, vagy akár egy olyan partíció, amelyen már nem használt adatok vannak.

Ha már mindegyik partíciónk betelt, akkor a többi operációs rendszer által felkínált eszközök (például MS-DOS-ban vagy Windows-ban az `fdisk`) valamelyikével először fel kell közülük szabadítanunk egyet a FreeBSD számára.

Amennyiben akadna egy használható partíció, akkor használjuk azt. Ekkor azonban előfordulhat, hogy ehhez először a meglévők közül össze kell majd zsugorítanunk valamelyiket.

A FreeBSD legkisebb telepíthető változata nagyjából 100 MB lemezterületet igényel. Azonban ez egy *nagyon* kicsi változat és szinte semmi helyet nem hagy a saját állományainknak. Sokkal valóságosabb, ha grafikus felület nélkül nagyjából 250 MB-ot mondunk, és legalább 350 MB-ot a grafikus felület használata esetén. Ha ezeken felül további szoftvereket is telepíteni kívánunk, még több helyre lesz szükségünk.

Amikor a FreeBSD számára akarunk helyet csinálni, vagy partíciókat akarunk átméretezni, használjuk például a **PartitionMagic®** nevű kereskedelmi szoftvert, vagy esetleg egy olyan szabad szoftvert, mint például a **GParted**. Ismereteink szerint a **PartitionMagic** és a **GParted** is használható az NTFS partíciókkal. A **GParted** számos live linuxos disztribúción megtalálható, ilyen többek között a SystemRescueCD (<http://www.sysresccd.org>).

Gondok lehetnek azonban a Microsoft Vista által használt partíciókkal. Ezért nem árt, ha az átméretezések során a kezünk ügyében van a Vista telepítő CD-je. Természetesen, mint minden lemezkarbantartási művelet esetén, ilyenkor is határozottan ajánlott biztonsági mentéseket készíteni.

Figyelem Az említett eszközök helytelen használata megsemmisítheti a lemezeinken tárolt adatokat, ezért a használatuk előtt gondoskodjunk friss, működőképes biztonsági mentésekről.

Példa 2-1. Meglevő partíció használata a méret megváltoztatása nélkül

Tegyük fel, hogy a számítógépünkben egyetlen 4 GB méretű lemez van, amelyen megtalálható a Windows valamelyik verziója, és ezt a lemezt korábban két, egyaránt 2 GB méretű meghajtóra osztottuk, a C : -re és D : -re. 1 GB adatunk van a C : meghajtón és fél GB a D : -n.

Mindez tehát azt jelenti, hogy a lemezünkön két partíció található, betűjelenként egy. Ha átmásoljuk a D : meghajtón levő adatainkat a C : meghajtóra, akkor ezzel felszabadíthatjuk a FreeBSD számára a második partíciót.

Példa 2-2. Meglevő partíció zsugorítása

Tegyük fel, hogy a számítógépünkben egyetlen 4 GB méretű lemez van, amelyet teljes egészében a Windows valamelyik példánya foglal el. A Windows telepítése során ezért minden bizonnyal egyetlen nagy partíciót hoztunk létre, amely a C : betűjelet kapta és a mérete 4 GB. Jelen pillanatban másfél GB helyet használunk a lemezen, és szeretnénk a FreeBSD számára 2 GB helyet felszabadítani.

A FreeBSD telepítéséhez a következők valamelyikét kell tennünk:

1. Mentsük le a Windows-os adatainkat, telepítsük újra a Windows-t úgy, hogy egy 2 GB méretű partíciót választunk neki a telepítése során.
2. A partíció összezsugorítására használjuk az előbb említett alkalmazásokat, például a **PartitionMagic**-et.

2.3.4. Szedjük össze a hálózati beállításainkat

Amennyiben a FreeBSD telepítésének részeként hálózatra is szándékozunk csatlakozni (például egy FTP vagy NFS szerverről akarunk telepíteni), ismernünk kell a hálózatra vonatkozó beállításainkat is. A telepítő rá fog kérdezni ezekre az információkra, amelyek megadása után a FreeBSD a telepítés befejezéséhez csatlakozni tud majd a hálózatra.

2.3.4.1. Csatlakozás Ethernet-hálózaton, kábel- vagy DSL-modemen keresztül

Ha egy Ethernet-hálózathoz, vagy magához az internethez csatlakozunk egy DSL- vagy kábelmodemen keresztül, akkor az alábbi adatokra lesz szükségünk:

1. IP-cím
2. Az alapértelmezett átjáró IP-címe
3. A gépünk neve
4. DNS (névfeloldó) szerverek IP-címei
5. Hálózati maszk

Ha nem ismerjük ezeket, érdeklődjünk a rendszergazdától vagy a szolgáltatóunktól. Elképzelhető az is, hogy mindezen információkat *DHCP* segítségével, automatikusan kapjuk meg. Ezt is mindenképpen jegyezzük fel.

2.3.4.2. Kapcsolódás modemmel

Ha az internet-szolgáltatónkhoz hagyományos modemem keresztül csatlakozunk, akkor is tudjuk telepíteni a FreeBSD-t interneten keresztül, azonban ez nagyon sokáig tarthat.

Ehhez tudnunk kell:

1. Az internet-szolgáltatónk behívószámát
2. A soros (COM) port számát, amelyen keresztül a modem kapcsolódik a gépünkhöz
3. Az internet-szolgáltatóunktól kapott felhasználói nevet és jelszót

2.3.5. Olvassuk el FreeBSD hibajegyzékét

Habár a FreeBSD Projekt igyekszik a FreeBSD minden egyes kiadását a lehető legmegbízhatóbban felkészíteni, hibák óhatatlanul is maradnak bennük. Nagyon ritka esetekben ezek a hibák magára a telepítés folyamatára is kihathatnak. Amint ezeket a problémákat sikerül felderíteni és javítani, rögvest megjelennek a FreeBSD honlapján található hibajegyzékben (<http://www.FreeBSD.org/releases/8.1R/errata.html>) (angolul). A telepítés előtt ezért mindig ajánlott átolvasni ezt a dokumentumot, így megbizonyosodunk róla, hogy semmilyen utólag felmerült probléma nem akadályozza munkánkat.

Az összes kiadáshoz tartozó információ, beleértve az egyes kiadások hibajegyzékeit is, a FreeBSD honlapjáról (<http://www.FreeBSD.org/index.html>) a kiadásokra vonatkozó információkat (<http://www.FreeBSD.org/releases/index.html>) tartalmazó részen érhető el (angolul).

2.3.6. Szerezzük be a FreeBSD telepítéséhez szükséges állományokat

A FreeBSD telepítése az alábbi helyek bármelyikén megtalálható állományok felhasználásával történik:

Lokálisan:

- CD vagy DVD
- Ugyanazon a számítógépen levő MS-DOS partíció
- Pendrive (USB-flash-tároló)
- SCSI- vagy QIC-szalag
- Floppylemezek

Hálózaton keresztül:

- FTP oldalról, tűzfalon keresztül vagy szükség szerint HTTP proxy használatával
- NFS szerverről
- Párhuzamos vagy soros vonali kapcsolaton keresztül

Ha megvásároltuk a FreeBSD telepítő CD-jét vagy DVD-jét, akkor már mindennel rendelkezünk a telepítéshez. Lépünk bátran tovább a következő szakaszra (2.3.7 Szakasz)!

Ha eddig még nem szereztük volna be a FreeBSD telepítéséhez szükséges állományokat, ugorjunk a 2.13 Szakaszhoz, ahol megtudhatjuk, hogyan készítjük elő a FreeBSD telepítését az imént felsorolt helyzetekben. A szakasz elolvasása után pedig jöjjünk vissza ide, majd folytassuk az olvasást a 2.3.7 Szakaszban.

2.3.7. Készítsünk egy rendszerindító lemezt

A FreeBSD telepítése úgy kezdődik, hogy a számítógépünkkel a FreeBSD telepítőt indítjuk el — ez viszont nem egy olyan program, amit más operációs rendszerben el tudunk indítani. A számítógépünk általában a merevlemezünkre telepített operációs rendszert indítja el, azonban beállítható úgy is, hogy az indulásához egy ún. “rendszerindító” (bootolható) floppy lemezt használjon. Napjaink számítógépei azonban a CD-meghajtóban levő CD-kről vagy USB lemezeiről is el tudnak indulni.

Tipp: Ha CD-n vagy DVD-n megvan a FreeBSD telepítője (akár megvettük, akár éppen magunk készítettük) és a számítógépünk tud CD-ről vagy DVD-ről rendszert indítani (a BIOS-ban van egy “Boot Order” vagy hozzá hasonló nevű beállítás), akkor kihagyhatjuk ezt a szakaszt. A FreeBSD CD- és DVD image-ek kiírásával egy rendszerindításra alkalmas lemezt kapunk, amiről minden további előkészület nélkül telepíthetünk.

Rendszerindításra alkalmas pendrive-ot az alábbi lépések mentén tudunk készíteni:

1. Az image állomány letöltése

A pendrive-okhoz készült image állományok a ISO-IMAGES/ könyvtárból tölthetők le,

`ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/architektúra/ISO-IMAGES/verzió/FreeBSD-8.1-RELEASE-architektúra` néven. Az *architektúra* és *verzió* helyére a telepítendő architektúrát és verziószámot helyettesítsük be.

Ennek megfelelően tehát például a FreeBSD/i386 8.1-RELEASE változata a

`ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.1/FreeBSD-8.1-RELEASE-i386-memstick.img` címről érhető el.

A pendrive image `.img` kiterjesztéssel rendelkezik. A ISO-IMAGES/ könyvtár általában több különféle állományt tartalmaz, ezek közül kell választanunk a FreeBSD telepítendő változatának, és sok esetben a telepítéshez rendelkezésre álló hardver típusának megfelelően.

Fontos: A következő lépés megkezdése előtt *készítsünk biztonsági mentést* a pendrive tartalmáról, mivel minden rajta levő adat *törlődni* fog.

2. A pendrive előkészítése

Figyelem Az itt található példában a rendszerindításhoz és így a művelet végrehajtásához a `/dev/da0` nevű eszközt fogjuk használni. Ezt ne felejtsük el helyettesíteni a rendszerünkön erre a célra használt eszköz nevével, máskülönben kárt tehetünk az adatainkban.

A `kern.geom.debugflags` változó értékének megfelelő beállításával engedélyezzük a céleszközön a Master Boot Record írását.

```
# sysctl kern.geom.debugflags=16
```

3. Az image pendrive-ra írása

Az `.img` kiterjesztésű állományt *nem* egyszerűen a pendrive-ra kell másolni, ez a lemez teljes tartalmát magában foglalja. Ennek megfelelően *nem* egyszerűen állományokat kell másolnunk az egyik lemeztől a másikra. Helyette a `dd(1)` parancs segítségével írjuk az image állomány tartalmát közvetlenül a lemezre.

```
# dd if=FreeBSD-8.1-RELEASE-i386-memstick.img of=/dev/da0 bs=64k
```

Rendszerindításra alkalmas floppy lemezt az alábbi lépések mentén tudunk készíteni:

1. A rendszerindító lemezek image-einek beszerzése

Fontos: A FreeBSD 8.0 kiadásától kezdődően megszűnik a floppy lemezek támogatása. Helyette telepítsünk pendrive-ról, amelyről fentebb olvashatunk, vagy egyszerűen használjunk CD-t vagy DVD-t.

A rendszerindító lemezek a telepítőeszköz `floppies/` könyvtárban találhatóak, illetve letölthetők az `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/architektúra/változat-RELEASE/floppies/` helyről. Az *architektúra* és *változat* helyére természetesen írjuk be a telepíteni kívánt architektúrát és verziót. Így például a FreeBSD/i386 8.1-RELEASE rendszerindító lemezei az `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/8.1-RELEASE/floppies/` címről érhetők el.

A floppyk image-ei `.flp` kiterjesztésűek. A `floppies/` könyvtár számos különféle image-et tartalmaz, ezek közül leginkább a telepítendő FreeBSD változat, valamint emellett olykor konkrétan a hardver határozza meg a használandót. Az esetek túlnyomó részében négy floppyra lesz szükségünk: `boot.flp`, `kern1.flp`, `kern2.flp` és `kern3.flp`. A lemezek image-eit illető legfrissebb információkat ugyanazon a könyvtáron belül szereplő `README.TXT` állományban olvashatjuk (angolul).

Fontos: Az FTP-hez használt programunkat az image-ek letöltése során ne felejtsük el *bináris (binary)* átviteli módban használni. Egyes böngészők hajlamosak ugyanis *szöveges (text vagy ASCII)* átviteli módot használni, ami viszont csak abból vehető észre, hogy nem tudjuk a lemezekről elindítani a rendszert.

2. A floppyk előkészítése

Mindegyik letöltendő image-hez elő kell készíteni egy-egy hajlékonylemezt. Nagyon fontos, hogy ezek a lemezek teljesen hibátlanok legyenek. Erről a legkönnyebben úgy győződhetünk meg, ha a lemezeket magunk formázzuk, és nem bízunk a különféle előreformázott (preformatted) floppykban. A Windows-ban található formázó segédprogram sem árul el nekünk semmit a lemezeken található hibás részekről, egyszerűen csak "rossznak" (bad) jelöli meg és figyelmen kívül hagyja ezeket. Határozottan ajánljuk, hogy amennyiben a telepítésnek ezt a módját választjuk, mindig használjunk teljesen új floppykat.

Fontos: Ha megpróbáljuk telepíteni a FreeBSD-t, és a telepítőprogram összeomlik, lefagy vagy bármilyen furcsaságot művel, elsőként mindenképpen a floppykra gyanakodhatunk. Ilyenkor írjuk ki az image-eket új lemezekre és próbálkozzunk újra a telepítéssel.

3. Az image állományok írása a floppykra

Az `.flp` kiterjesztésű állományok *nem* a lemezre másolható hagyományos állományok, hanem a lemezek teljes tartalmának képei, ezért ezeket egyszerűen *nem* másolhatjuk egyik lemezzől a másikra. Az image-ek közvetlen lemezreírásához ehelyett kifejezetten erre a célra alkalmas eszközöket kell használnunk.

Azok számára, akik a floppykat MS-DOS/Windows rendszerű számítógépeken kívánják elkészíteni, mellékelünk egy `fdimage` nevű segédprogramot.

Ha a CD-meghajtónk betűjele például `E:` és a telepítő CD-n található image-eket szeretnénk kiírni vele, akkor ezt a parancsot kell kiadnunk:

```
E:\> tools\fdimage floppies\boot.flp A:
```

Ezután ismételtel adjuk ki az iménti parancsot minden egyes használni kívánt `.flp` állományra, azonban előtte mindig tegyünk be egy újabb floppyt, és a ráírt image-ek neveivel folyamatosan címkézzük fel a lemezeket. A megadott parancsot természetesen mindig írjuk át a konkrét `.flp` állományok tényleges elérési útvonalainak megfelelően. Ha nincs CD-nk, akkor az `fdimage` programot az FreeBSD FTP oldalán található `tools` könyvtárból (<ftp://ftp.FreeBSD.org/pub/FreeBSD/tools/>) is letölthetjük.

Amikor a lemezeket egy UNIX rendszeren készítenénk el (például egy másik FreeBSD rendszeren), akkor a `dd(1)` parancs is használható az image állományok közvetlen lemezreírásához. FreeBSD alatt így néz ki a paraméterezése:

```
# dd if=boot.flp of=/dev/fd0
```

FreeBSD-n a `/dev/fd0` az első hajlékonylemezes meghajtóra hivatkozik (tehát az `A:` betűjelű meghajtóra). Ennek megfelelően a `/dev/fd1` jelenti a `B:` meghajtót és így tovább. Más UNIX változatok esetleg más neveket használhatnak a hajlékonylemezes meghajtók megnevezésére, ezért erről érdemes ilyenkor tájékozódni az adott rendszerhez tartozó dokumentációban.

Most már készen állunk a FreeBSD telepítésére!

2.4. A telepítés megkezdése

Fontos: Alapértelmezés szerint a telepítés egészen addig nem fog semmit sem írni a lemezekre, amíg a következő üzenet fel nem bukkan:

```
Last Chance: Are you SURE you want continue the installation?
```

```
If you're running this on a disk with data you wish to save then WE  
STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!
```

```
We can take no responsibility for lost disk contents!
```

A szöveg fordítása:

```
Utolsó esély: BIZTOSAN folytatni kívánja a telepítést?
```

```
Ha olyan lemezre szeretne telepíteni, amelyen fontos adatok  
találhatóak, HATÁROZOTTAN JAVASOLJUK, hogy a továbblépés előtt  
KÉSZÍTSSEN RÓLUK MEGBÍZHATÓ BIZTONSÁGI MÁSOLATOT!
```

```
Nem vállalunk semmilyen felelősséget az elveszett adatokért!
```

A telepítőtől tehát a fenti, végső figyelmeztetés előtt bármikor ki lehet lépni anélkül, hogy a merevlemezünkön levő adatokat veszélyeztetnénk. Ha úgy érezzük, hogy valamit véletlenül rosszul állítottunk volna be a telepítés során, ekkor még minden komolyabb kár okozása nélkül kikapcsolhatjuk a számítógépünket.

2.4.1. A rendszer indítása

2.4.1.1. Rendszerindítás i386™-on

1. Kezdjük egy kikapcsolt számítógéppel.
2. Kapcsoljuk be a számítógépet. Az indulása során látnunk kell egy olyan opciót, amivel be tudunk lépni a rendszer beállításait tartalmazó menübe, avagy a BIOS-ba. Ezt többnyire a **F2**, **F10**, **Del** vagy a **Alt+S** lenyomásával érhetjük el. Ezek közül használjuk a képernyőn megjelenő billentyűket. Előfordulhat, hogy induláskor a számítógépünk semmilyen szöveget, csak egy képet mutat. Ilyenkor általában a **Esc** billentyű megnyomására eltűnik a kép és láthatóvá válnak a számunkra fontos üzenetek.
3. Miután beléptünk a menübe, keressük meg azt a beállítást, amely a rendszerindításhoz használt eszközt határozza meg. Ennek a neve sokszor "Boot Order" (rendszerindítási sorrend) vagy valami hozzá hasonló. Itt mindenféle eszköz felsorolását találjuk: Floppy, CDROM, First Hard Disk (első merevlemez meghajtó) és így tovább.

Ha CD-ről akarjuk a telepítést elindítani, akkor akkor a CDROM eszközt válasszuk. Ha bármilyen kétség merülne fel bennünk, keressük meg ezt a beállítást a számítógéphez és/vagy az alaplaphoz kapott kézikönyvben.

Igényeink szerint végezzük el a beállítást, majd mentsük el és lépünk ki. Most indítsuk újra a számítógépet.

4. Ha a 2.3.7 Szakaszban leírtak szerint rendszerindító pendrive-ot készítettünk, akkor bekapcsolás előtt csatlakoztassuk a számítógéphez.

Ha CD-ről indítjuk a telepítést, akkor kapcsoljuk be a számítógépet és az elindulása után igyekezzünk minél hamarabb betenni a lemezt a meghajtóba.

Megjegyzés: A FreeBSD 7.3 és az azt megelőző változatokban a 2.3.7 Szakaszban leírtak szerint előkészített floppy-ról is el tudjuk kezdeni a telepítést. Ezek egyike lesz az első rendszerindító lemez, a `boot.flp`. Helyezzük ezt a lemezt a meghajtóba, és indítsuk el vele a számítógépet.

Ha minden próbálkozásunk ellenére a számítógépünk a megszokott módon indul és a meglevő operációs rendszert tölti be, akkor a következőkkel lehet a gond:

1. A lemezeket nem raktuk be eléggé korán. Hagyjuk benn ezeket és próbáljuk meg ismét újraindítani a számítógépet.
2. Nem állítottuk be jól a BIOS-t. Próbáljuk meg egészen addig újra végrehajtani az előző lépést, amíg a megfelelő beállítást el nem találjuk.
3. A BIOS nem támogatja a kiválasztott eszközről történő rendszerindítást.
5. A FreeBSD megkezdte az indulását. Ha CD-ről indítjuk, akkor valami ehhez hasonló fogunk látni (a konkrét verzióra vonatkozó adatokat itt most kihagytuk):

Booting from CD-Rom...

```
645MB medium detected
CD Loader 1.2
```

```
Building the boot loader arguments
Looking up /BOOT/LOADER... Found
Relocating the loader and the BTX
Starting the BTX loader
```

```
BTX loader 1.00 BTX version is 1.02
Console: internal video/keyboard
BIOS CD is cd0
BIOS drive C: is disk0
BIOS drive D: is disk1
BIOS 639kB/261056kB available memory
```

```
FreeBSD/i386 bootstrap loader, Revision 1.1
```

```
Loading /boot/defaults/loader.conf
/boot/kernel/kernel text=0x64daa0 data=0xa4e80+0xa9e40 syms=[0x4+0x6cac0+0x4+0x88e9d]
\
```

Amikor floppyról indítjuk a rendszert, ehhez hasonlóval találkozhatunk (itt sem szerepelnek most verzióadatok):

```
Booting from Floppy...
Uncompressing ... done
```

```
BTX loader 1.00 BTX version is 1.01
Console: internal video/keyboard
BIOS drive A: is disk0
BIOS drive C: is disk1
BIOS 639kB/261120kB available memory
```

```
FreeBSD/i386 bootstrap loader, Revision 1.1
```

```
Loading /boot/defaults/loader.conf
/kernel text=0x277391 data=0x3268c+0x332a8 |
```

```
Insert disk labelled "Kernel floppy 1" and press any key...
```

Kövessük a képernyőn megjelenő utasítást ("Helyezze be a "Kernel floppy 1" címkéjű lemezt és nyomjon meg egy billentyűt..."), tehát vegyük ki a `boot.flp` image-hez tartozó lemezt és tegyük be helyette a `kern1.flp` image-hez tartozó lemezt, majd nyomjuk le az **Enter** billentyűt. Várjuk meg amíg a rendszer megkezd az indulást az első lemezről, majd az utasításoknak megfelelően folyamatosan tegyük be a soron következő lemezeket.

6. Miután elindítottuk a rendszert CD-ről, pendrive-ról vagy floppy-ról, a rendszerindítási folyamat be fogja hozni a FreeBSD rendszertöltőjének menüjét:

Ábra 2-1. FreeBSD rendszerbetöltő menüje



Várjuk ki a tíz másodperces szünetet vagy egyből nyomjuk le az **Enter** billentyűt.

2.4.1.2. Rendszerindítás SPARC64®-en

A legtöbb SPARC64® alapú rendszert úgy állították be, hogy automatikusan lemezzről induljon. A FreeBSD telepítéséhez azonban hálózaton keresztül vagy CD-ről kell indítanunk a rendszert, ezért módosítanunk kell a PROM (az OpenFirmware) beállításait.

Mindehhez indítsuk újra a rendszert és várjuk meg, amíg feltűnik a rendszerindító üzenet. A konkrét üzenet nagyjában függ a számítógép típusától, azonban valami ilyesmi lesz:

```
Sun Blade 100 (UltraSPARC-IIe), Keyboard Present
Copyright 1998-2001 Sun Microsystems, Inc. All rights reserved.
OpenBoot 4.2, 128 MB memory installed, Serial #51090132.
Ethernet address 0:3:ba:b:92:d4, Host ID: 830b92d4.
```

Amikor megpróbálja a rendszert elindítani a lemezzől, a PROM parancssorának bekéréshez nyomjuk le a billentyűzetten az **L1+A** vagy a **Stop+A** billentyűket, esetleg a soros konzolon keresztül küldjünk egy **BREAK** parancsot (például a `tip(1)` vagy `cu(1)` man oldalakon szereplő `~#` parancs használatával). Körülbelül így néz ki:

```
ok ❶
ok {0} ❷
```

- ❶ Ez a fajta parancssor csak az egy processzorral rendelkező rendszereken jelenik meg.
- ❷ Ez a fajta parancssor többprocesszoros (SMP) rendszereken jelenik meg, ahol a szám az éppen aktív processzor sorszámát jelöli.

Most helyezzük a CD-t a meghajtóba, és a PROM parancssorában pedig gépeljük be `boot cdrom` parancsot.

2.4.2. Az eszközkérés eredményeinek vizsgálata

A képernyőn megjelenő utolsó pár száz sor mindig eltárolódik, később tetszőlegesen átvizsgálhatóak.

A puffer tartalmának átnézéséhez nyomjuk le a **Scroll Lock** billentyűt, amivel bekapcsoljuk a korábban megjelölt üzenetek közti visszalépést. Itt a nyílbillentyűk, vagy a **PageUp** és **PageDown** billentyűk használhatóak a kiírások átböngészéséhez. A **Scroll Lock** ismételt lenyomásával kiléphetünk ebből a módból.

Tegyük most mi is ezt, és nézzük az összes olyan üzenetet, amely a rendszermag indulása során keletkezett. A 2-2 Ábrában látható szövegekhez hasonlóakat fogunk találni, habár ez a számítógépben található konkrét eszközöktől függően eltérő lehet.

Ábra 2-2. Példa az eszközkérés eredményeire

```
avail memory = 253050880 (247120K bytes)
Preloaded elf kernel "kernel" at 0xc0817000.
Preloaded mfs_root "/mfsroot" at 0xc0817084.
md0: Preloaded image </mfsroot> 4423680 bytes at 0xc03ddcd4

mdl: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1:<VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
isa0: <iSA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
uhci0 <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci
0
usb0: <VIA 83572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr1
uhub0: 2 ports with 2 removable, self powered
pci0: <unknown card> (vendor=0x1106, dev=0x3040) at 7.3
dc0: <ADMtek AN985 10/100BaseTX> port 0xe800-0xe8ff mem 0xdb000000-0xeb0003ff ir
q 11 at device 8.0 on pci0
dc0: Ethernet address: 00:04:5a:74:6b:b5
miibus0: <MII bus> on dc0
ukphy0: <Generic IEEE 802.3u media interface> on miibus0
ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xec00-0xec1f irq 9 at device 10.
0 on pci0
ed0 address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
orm0: <Option ROM> at iomem 0xc0000-0xc7fff on isa0
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
```

```

fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbd0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq1 on atkbd0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbd0
psm0: model Generic PS/@ mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
siol at port 0x2f8-0x2ff irq 3 on isa0
siol: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
pppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
plip0: <PLIP network interface> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master UDMA33
acd0: CD-RW <LITE-ON LTR-1210B> at atal-slave PIO4
Mounting root from ufs:/dev/md0c
/stand/sysinstall running as init on vty0

```

Figyelmesen olvassuk át az üzeneteket, és bizonyosodjunk meg róla, hogy a FreeBSD minden számunkra fontos eszközt felismert. Ha nem látunk egy eszközt, akkor azt valószínűleg nem találta meg. Egy saját rendszermag létrehozásával azonban fel tudunk ismertetni olyan eszközöket is, amelyek támogatása eredetileg nem szerepel a GENERIC rendszermagban. Ilyenek például a hangkártyák.

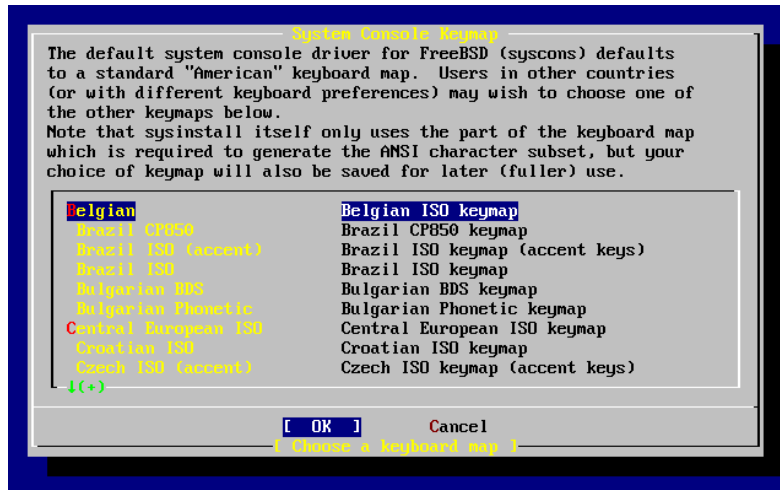
A FreeBSD 6.2 vagy későbbi változataiban az eszközök felkutatása után a 2-3 Ábrában láthatóak következnek. Itt a nyílbillentyűk segítségével választhatjuk ki az országot (country), térséget (region) vagy csoportot (group). Az **Enter** lenyomása után pillanatok alatt beállítódik az országunk. Ha meg akarjuk ismételni az iménti beállítást, pillanatok alatt ki tudunk lépni a **sysinstall** programból.

Ábra 2-3. Az ország kiválasztása

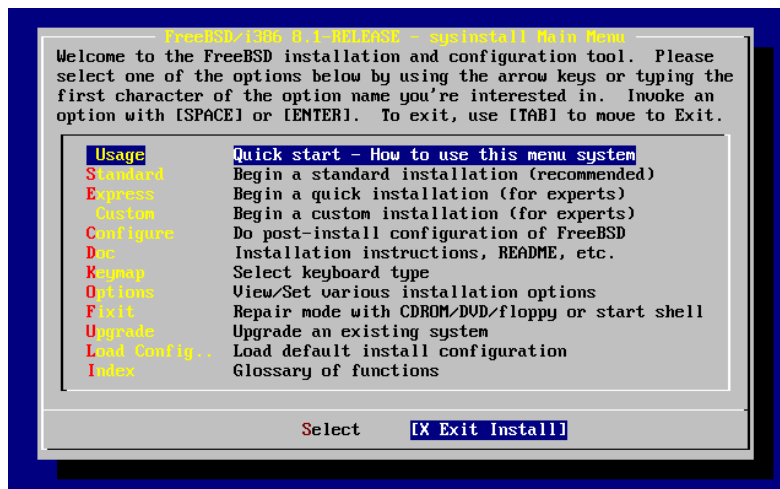


Ha országment United States (Egyesült Államok) került beállításra, akkor a szabványos amerikai billentyűzet-kiosztás állítódik be. A többi ország esetében az alábbi menü jelenik meg. A kurzormozgató billentyűk segítségével ekkor keressük meg ki a számunkra megfelelő kiosztást, és az **Enter** billentyű lenyomásával válasszuk ki.

Ábra 2-4. A billentyűzet típusának kiválasztása



Ábra 2-5. Kilépés a sysinstall programból



A telepítőprogram főképernyőjén válasszuk ki a nyílbillentyűkkel az Exit Install (“Kilépés a telepítésből”) menüpontot. Erre a következő üzenet fog megjelenni:

```

User Confirmation Requested
Are you sure you wish to exit? The system will reboot

[ Yes ]    No

```

Az üzenet fordítása:

```
Felhasználói megerősítés szükséges
Valóban ki akar lépni? A rendszer ezt követően újra fog
indulni
```

```
[ Igen ]      Nem
```

Ha a [**Yes**] választ adjuk és a CD-t az újraindításkor is a meghajtóban hagyjuk, akkor a telepítőprogram még egyszer el fog indulni.

Ha floppyról indítottuk volna a rendszert, az újraindítás előtt vegyük ki a `boot.flp` image-et tartalmazó lemezt.

2.5. A sysinstall bemutatása

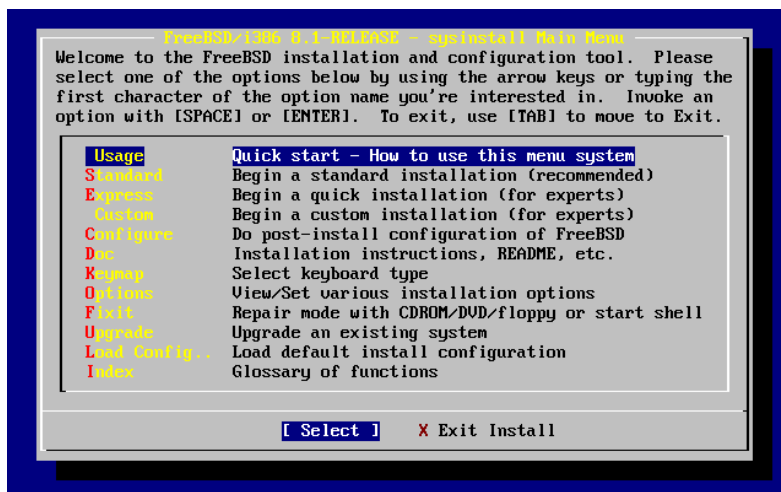
A **sysinstall** a FreeBSD Projekt által fejlesztett telepítőprogram. Konzol alapú, menükre és képernyőkre oszlik, amelyeken a beállításokat és a telepítési folyamat irányítását tudjuk elvégezni.

A **sysinstall** menürendszerét több más billentyű mellett legfőképpen a nyílbillentyűkkel, az **Enter**, **Tab** és a **Szókőz** billentyűkkel kezelhetjük. Ezek és az általuk elvégezhető feladatok részletes leírása a **sysinstall** használatáról szóló információk között található.

Ennek megtekintéséhez először győződjünk meg róla, hogy a 2-6 Ábra által illusztrált helyzetnek megfelelően kiválasztottuk a **Usage** ("Használat") menüpontot és a [Select] ("Kiválaszt") feliratú gombon állunk, majd nyomjuk le az **Enter** billentyűt.

Ezt követően megjelenik a menürendszer használatát bemutató leírás. Miután végigolvastuk, a főmenübe az **Enter** billentyű lenyomásával tudunk visszajutni.

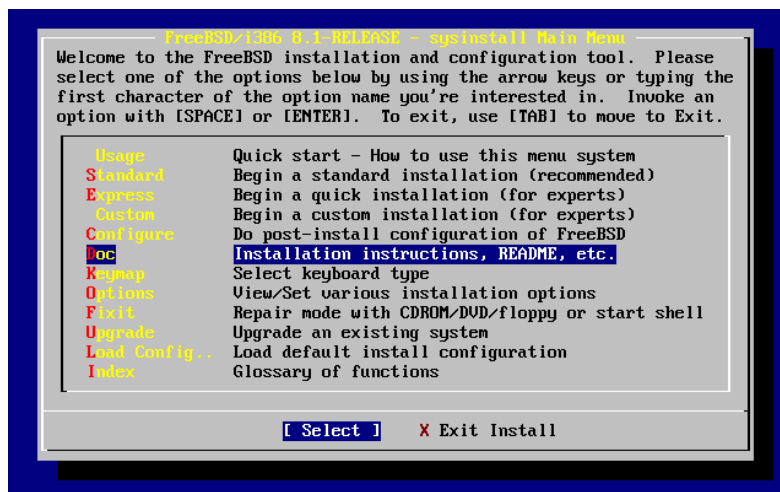
Ábra 2-6. A "Usage" kiválasztása a sysinstall főmenüjében



2.5.1. A dokumentációs menü kiválasztása

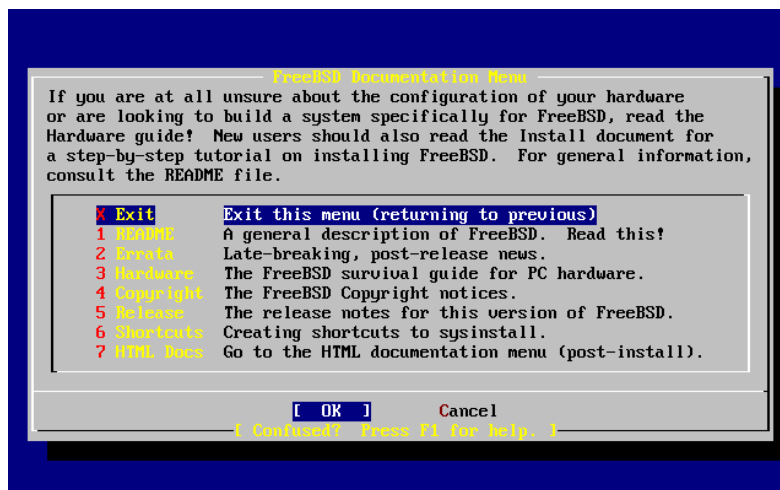
A főmenüben a nyílbillentyűkkel válasszuk a **DOC** feliratú menüpontot és nyomjuk meg az **Enter** billentyűt.

Ábra 2-7. A dokumentációs menü kiválasztása



Ezzel megjelenik a dokumentációs menü.

Ábra 2-8. A sysinstall dokumentációs menüje



Feltétlenül olvassuk el az itt található leírásokat.

A dokumentumok elolvasásához először válasszunk közülük a nyílbillentyűkkel, majd nyomjuk meg az **Enter** billentyűt. A dokumentum elolvasása után az **Enter** lenyomásával tudunk visszatérni a dokumentációs menübe.

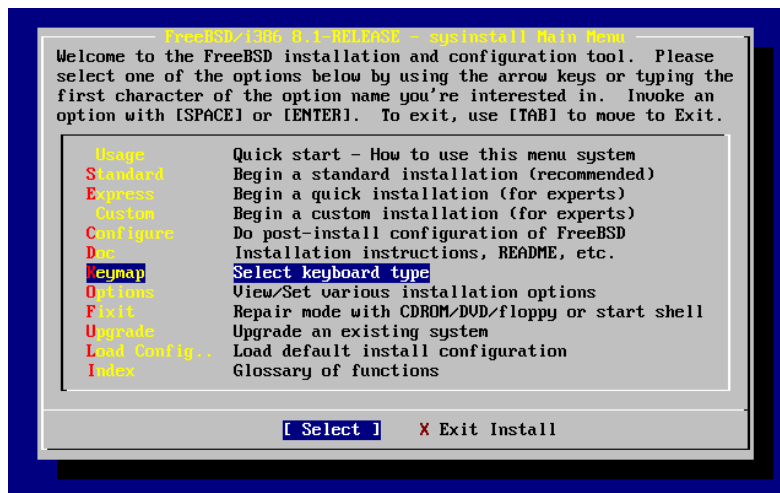
A dokumentációs menüből a főmenübe úgy tudunk kilépni, ha a nyílbillentyűkkel kiválasztjuk az Exit ("Kilépés") menüpontot és megnyomjuk az **Enter** billentyűt.

2.5.2. A billentyűkiosztás menüjének kiválasztása

A billentyűzetkiosztás megváltoztatásához válasszuk ki a nyílbillentyűk segítségével a Keymap menüpontot a menüből és nyomjuk meg az **Enter** billentyűt. Erre természetesen csak akkor lesz szükségünk, ha nem szabványos

vagy nem angol billentyűzetet használunk.

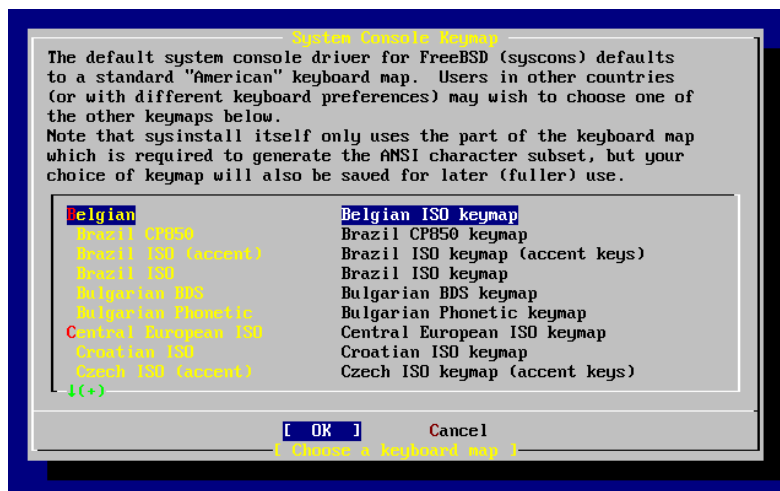
Ábra 2-9. A sysinstall főmenüje



A különböző billentyűkiosztásoknak megfelelő menüpontok a fel/le nyílak és a **Szókőz** billentyű segítségével választhatók ki. A **Szókőz** ismételt lenyomásával töröljük a választásunkat. A befejezéshez válasszuk ki a nyilakkal a [OK] gombot és nyomjuk le az **Enter** billentyűt.

A mellékelt képen a lista egy része látható csupán. Ha a **Tab** billentyűvel a [Cancel] gombot választjuk, akkor az alapértelmezett billentyűkiosztást kapjuk és visszerülünk a főmenübe.

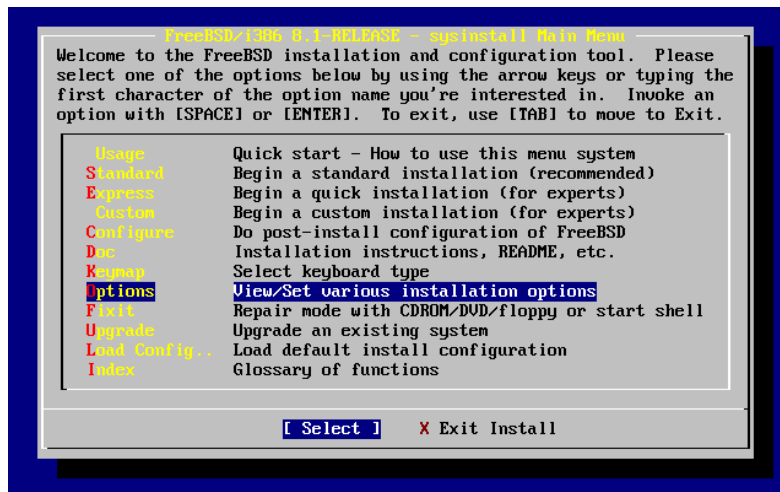
Ábra 2-10. A sysinstall billentyűkiosztást beállító menüje



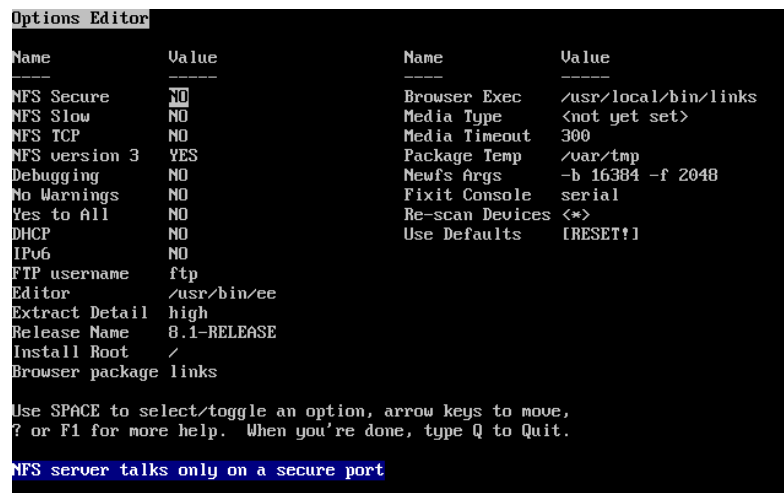
2.5.3. A telepítés beállításai tartalmazó képernyő

Válasszuk az Options ("Beállítások") menüpontot, majd nyomjuk le az **Enter** billentyűt.

Ábra 2-11. A sysinstall főmenüje



Ábra 2-12. A sysinstall beállításai



Az itt szereplő alapértelmezett értékek a legtöbb felhasználó számára minden további nélkül megfelelnek, nem szükséges a megváltoztatásuk. A kiadás neve (“release name”) mező értéke a telepítendő verziótól függően változhat.

A kiválasztott mező rövid leírása a képernyő alján, kékkel kiemelten jelenik meg. A Use Defaults (“Az alapértelmezések használata”) beállítás az alapértelmezésére állítja vissza az összes értéket.

Az **F1** lenyomásával elolvashatjuk a különböző beállításokhoz tartozó súgót.

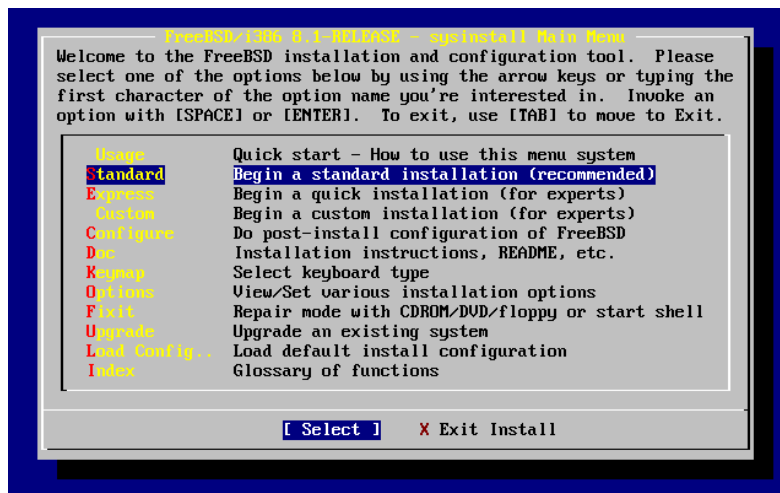
A **Q** billentyűvel visszatérhetünk a főmenübe.

2.5.4. Egy szabványos telepítés megkezdése

A Standard (“Szabványos”) elnevezésű menüpont által felkínált telepítési módszer ajánlott a UNIX-szal vagy a FreeBSD-vel most ismerkedők számára. A telepítés megkezdéséhez a nyilakkal válasszuk ki a Standard

menüpontot, majd nyomjuk meg az **Enter** billentyűt.

Ábra 2-13. Egy szabványos telepítés megkezdése



2.6. Lemezterület lefoglalása

Első feladatunk lemezterületet foglalni a FreeBSD számára, majd megcímkézni azt, hogy a **sysinstall** elő tudja készíteni. Ehhez tisztában kell lennünk azzal, hogy a FreeBSD milyen formában is keresi az adatokat a lemezünkön.

2.6.1. A BIOS meghajtószámozása

Egy témára különösen tekintettel kell lennünk mielőtt telepítenénk és beállítanánk a FreeBSD-t a rendszerünkön, főleg abban az esetben, ha több merevlemezünk is van.

Egy BIOS-függő operációs rendszert, például MS-DOS-t vagy Windows-t futató PC esetén a BIOS az operációs rendszer beleegyezésével képes elvonatkoztatni a lemezek megszokott sorrendjétől. Ennek köszönhetően a felhasználó nem csak az ún. “primary master” (elsődleges master) merevlemez meghajtótól tudja elindítani a rendszert. Ez kifejezetten kényelmes megoldás az olyan felhasználók számára, akik az elsővel teljesen megegyező második merevlemez megvásárlásával kialakították a rendszerük egyszerű és egyben a legolcsóbb biztonsági mentését, amire a **Ghost®** vagy **XCOPY** programokkal tudnak rendszeres másolatokat készíteni. Így, ha az elsődleges meghajtó tönkremegy vagy vírus támadja meg, esetleg az operációs rendszer egy hiba miatt használhatatlanná teszi, akkor a BIOS-t utasíthatjuk a meghajtók logikai cseréjére és ezzel könnyen helyre tudjuk állítani. Olyan, mintha a ház felnyitása nélkül felcseréltük volna a lemezeket bekötő kábeleket.

A SCSI-vezérlőkkel szerelt drágább rendszerek gyakran tartalmaznak olyan BIOS-bővítéseket, amelyeken keresztül a SCSI-lemezek ugyanígy tetszőlegesen átrendezhetők, egészen hét meghajtóig.

Az ilyen lehetőségek használatához szokott felhasználókat azonban könnyen csalódás érheti, amikor a FreeBSD nem az elvárásaiknak megfelelően cselekszik. A FreeBSD ugyanis nem használja a BIOS-t és nem ismeri a “BIOS logikai meghajtókiosztását”. Ez meglehetősen meglehetősen vezethet, főleg akkor, amikor paramétereiket tekintve a meghajtók fizikailag teljesen megegyeznek és ráadásul egymás másolatait tartalmazzák.

A FreeBSD telepítése előtt mindig állítsuk vissza a BIOS-ban a meghajtók eredeti sorrendjét, és a használatához hagyjuk is így ezt a beállítást. Ha valamiért mégis meg kellene cserélnünk a meghajtókat, akkor ezentúl válasszuk a nehezebb utat: nyissuk ki a gépházat és kössük át a kábeleket, tegyük át a jumpereket mi magunk.

SzéljegyzetRészlet Frédi és Vili különleges kalandjaiból:

Vili fogott egy öreg Winteles számítógépet, hogy készítsen belőle egy FreeBSD-s rendszert Frédinek. Vili ehhez beszerel egy SCSI-meghajtót, ami így nullás SCSI-egység lesz, majd telepíti rá a FreeBSD-t.

Frédi nekilát használni a rendszert, azonban pár nap elteltével tapasztalja, hogy az öregecske SCSI-meghajtó számos apróbb hibát jelez, és ezért szól Vilinek.

Néhány nappal később Vili eldönti, ideje pontot tenni az ügy végére, ezért a raktárban levő SCSI-lemezek köztül elhoz az eredetivel egy teljesen megegyezőt. Az előzetes felületellenőrzés eredményei szerint a meghajtó tökéletesen működik, ezért Vili beszerelni ezt a meghajtót a négyes SCSI-egységként, majd lemásolja a nullás meghajtó tartalmát a négyesre. Miután beszerelte a tökéletesen üzemelő új meghajtót, Vili úgy határoz, ideje megkezdeni a használatát, ezért beállítja a SCSI BIOS-át, hogy a rendszer a nullás helyett ezentúl a négyes egységről induljon. A FreeBSD elindul és mindenki örül.

Frédi ezután folytatja megszokott munkáját, majd Vili és Frédi úgy gondolják, itt az ideje az újabb izgalmaknak — frissítsünk a FreeBSD egy újabb változatára. Vili ekkor eltávolítja a nullás SCSI-egységet, mivel már egyébként is kezdett tönkremenni, és kicseréli egy másik teljesen azonos lemezes meghajtóra. Vili ezt követően Frédi internetről letöltött varázslatos floppyjainak segítségével feltelepíti a FreeBSD új verzióját az új nullás SCSI-egységre. A telepítés minden gond nélkül lezajlik.

Frédi próbálgatja is a FreeBSD új változatát néhány napig, és számára ez elegendő bizonyíték ahhoz, hogy a munkahelyén is használja. Ideje hát átmásolni a régi munkáit, ezért Frédi csatlakoztatja a (korábbi FreeBSD változat legfrissebb változatát tartalmazó) négyes SCSI-egységet. Frédin azonban hirtelen aggodalom tör ki, hiszen a négyes SCSI-egységen sehol sem találja munkája féltett eredményeit.

Hova tűntek azok a komisz adatok?

Amikor Vili másolatot készített az eredeti nullás SCSI-egységről a négyes SCSI-egységre, a négyes egység egy “új klón” lett. Amikor a rendszerindításhoz Vili átrendezte a meghajtókat a SCSI BIOS-ban, azzal csak magát csapta be, ugyanis a FreeBSD továbbra is a nullás SCSI-egységről indult el! A BIOS által kiválasztott meghajtóról az effajta beállítások hatására ugyan behozható a rendszerindító és -betöltő programok egy része, de amikor a FreeBSD rendszermagja átveszi a vezérlést, a BIOS által meghatározott sorrendiség figyelmen kívül marad és a FreeBSD visszatér a meghajtók eredeti rendezéséhez. Tehát ebben az esetben a rendszer továbbra is az eredeti nullás SCSI-egységről folytatja a működést, és Frédi összes adata itt található, nem pedig a négyes SCSI-egységen. A négyes SCSI-egységről futó rendszer illúziója így mindössze az emberi elvárások szüleménye.

Örömmel említjük meg, hogy egyetlen byte-nyi adat sem sérült meg vagy pusztult el a jelenség felfedezése során. A korábbi nullás SCSI-egységet még sikerült megmenteni a szemétdombról és Frédi összes munkája visszakerült (és Vili most már el tud számolni nulláig).

Habár a tanmesénkben SCSI-meghajtókról esett szó, ugyanez fennáll az IDE-meghajtókra is.

2.6.2. Slice-ok létrehozása az FDisk használatával

Megjegyzés: Itt még semmilyen változtatás nem kerül lemezre. Ha úgy érezzük, hogy valamit rosszul csináltunk és újra el akarjuk kezdeni a telepítést, a menük segítségével büntetlenül távozhatunk a **sysinstall**ból és újra

próbálkozhatunk, vagy az **U** billentyű lenyomásával aktiválhatjuk az Undo ("Visszacsinál") funkciót. Ha véletlenül összezavarodtunk volna és nem találunk kilépési lehetőséget, akkor bármikor ki tudjuk kapcsolni a számítógépet.

A **sysinstall**ban a szabványos telepítés megkezdésekor az alábbi üzenet jelenik meg:

Message

In the next menu, you will need to set up a DOS-style ("fdisk") partitioning scheme for your hard disk. If you simply wish to devote all disk space to FreeBSD (overwriting anything else that might be on the disk(s) selected) then use the (A)ll command to select the default partitioning scheme followed by a (Q)uit. If you wish to allocate only free space to FreeBSD, move to a partition marked "unused" and use the (C)reate command.

[OK]

[Press enter or space]

Az üzenet fordítása:

Üzenet

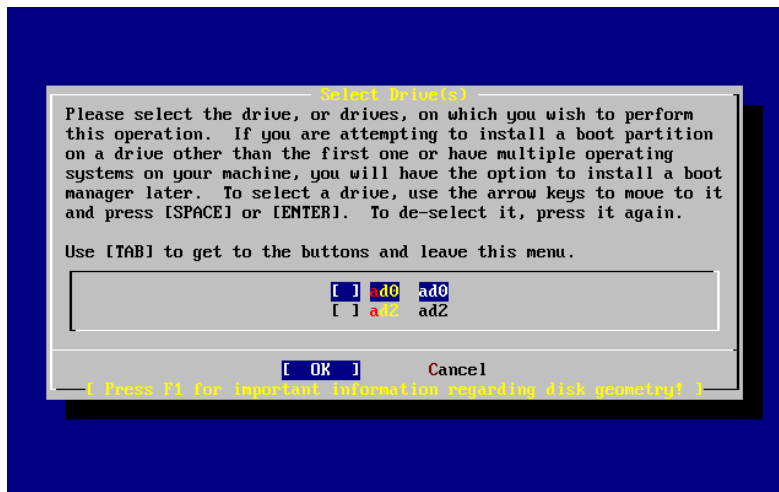
A most következő menüben össze kell állítanunk a merevlemezünk DOS-szerű ("fdiskes") partícióit. Amennyiben egyszerűen csak át akarjuk adni az összes lemezterületet a FreeBSD számára (ezzel felülírva mindent, ami a kiválasztott lemezeken található), akkor az alapértelmezett partíció-kiosztás kiválasztásához használjuk az (A)ll (Mind), majd utána a (Q)uit (Kilépés) parancsokat. Ha viszont csak az éppen szabad területet szánjuk a FreeBSD-nek, lépünk egy "unused" ("üres") feliratú partícióra és használjuk a (C)reate (Létrehozás) parancsot.

[OK]

[Nyomja le az Enter vagy a Szóköz billentyűt]

Az utasításnak megfelelően nyomjuk le az **Enter** billentyűt. Ezután a rendszermag által az eszközök felkutatása során megtalált összes merevlemez meghajtót láthatjuk. A 2-14 Ábra egy két IDE-lemezzel rendelkező rendszert mutat be, amelyeknek nevei rendre ad0 és ad2.

Ábra 2-14. A meghajtó kiválasztása az FDisk számára



Feltűnhet, hogy itt nem szerepel az `ad1`. Vajon miért maradt ki?

Képzeld el, mi történne, ha két IDE-csatolós merevlemezünk lenne: az egyik az első IDE-vezérlőn, a másik pedig a második IDE-vezérlőn lenne master. Ha a FreeBSD a megtalálásuk szerint `ad0` és `ad1` nevekké számozná ezeket, attól még minden remekül működhetne.

Ha azonban beszerelnénk egy harmadik lemezt, például egy slave eszközt kapcsolnánk az első IDE-vezérlőre, akkor már ez lenne a `ad1`, és ennek megfelelően a korábban `ad1` megnevezésű meghajtó pedig az `ad2`. Mivel az állományrendszerek felkutatására általában az eszközneveket (mint amilyen a `ad1s1a`) használják, ezért ilyenkor azt tapasztalhatnánk, hogy bizonyos állományrendszerek helytelenül jelennek meg, ezért meg kell változtatnunk a FreeBSD ezeket érintő beállításait.

A probléma megoldására a rendszermag beállítható úgy, hogy az IDE-lemezeket a kapcsolódásuk szerint azonosítsa, ne pedig a megtalálásuk sorrendje szerint. Ezzel a kialakítással a második IDE-vezérlőn található master lemez *mindig* az `ad2` eszköz lesz, tehát még olyankor is, amikor egyáltalán nincs a rendszerünkben `ad0` vagy `ad1` eszköz.

Ez a beállítás alapértelmezés a FreeBSD rendszermagjában, és ez magyarázza, hogy az iménti ábra miért csak `ad0` és `ad2` eszközöket mutat. Tehát a képen szereplő számítógép mind a két IDE-vezérlőjének master csatornáján található egy-egy IDE-lemez, a slave csatornákon pedig nincs egy sem.

Itt válasszuk ki azt a lemezt, amelyre a FreeBSD-t telepíteni kívánjuk, majd nyomjuk meg a [OK] gombot. Erre az 2-15 Ábra által bemutatott képernyővel elindul az **FDisk**.

Az **FDisk** képernyője három részre osztható.

Az első részben, amely a képernyő felső két sorát foglalja össze, láthatjuk az éppen kiválasztott lemez adatait: a FreeBSD szerinti nevét, a paramétereit és az összméretét.

A második részben láthatjuk a lemezen megtalálható slice-okat: hol kezdődnek (Offset) és hol érnek véget (End); mekkorák (Size); a FreeBSD milyen névvel hivatkozik rájuk (Name); milyen leírás (Description) és altípus (Subtype) tartozik hozzájuk. A példában két kicsi üres slice-ot láthatunk, ami a PC-k lemezkiosztására jellemző. Ezenkívül felfedezhetünk egy nagyobb méretű FAT típusú slice-ot is, amely az MS-DOS / Windows világban szinte minden bizonnyal a `C:` betűjelet viseli, valamint egy kiterjesztett slice-ot is, amely az MS-DOS / Windows számára további meghajtókat is tartalmazhat.

A harmadik részben az **FDisk** működtetésére használható parancsok láthatóak.

Ábra 2-15. Átlagos Fdisk partíciók szerkesztés előtt

```

Disk name:      ad0                      FDISK Partition Editor
DISK Geometry: 16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset      Size(ST)      End      Name  PType      Desc  Subtype  Flags
-----
0           63           62      -      6      unused     0
63          4193217      4193279  ad0s1  2      fat       14      >
4193280     1008          4194287  -      6      unused     0      >
4194288     12319776      16514063  ad0s2  4      extended  15      >

The following commands are supported (in upper or lower case):

A = Use Entire Disk      G = set Drive Geometry  C = Create Slice      F = 'DD' mode
D = Delete Slice         Z = Toggle Size Units   S = Set Bootable     I = Wizard m.
T = Change Type          U = Undo All Changes    Q = Finish

Use F1 or ? to get more help, arrow keys to select.

```

A most következő teendőink attól függenek, hogy miként is akarjuk felosztani a lemezünket.

Ha az egész lemezt a FreeBSD használatára áldozzuk (és amikor majd megerősítjük a **sysinstall** számára a továbblépést, a lemezen így minden más adat törlődni fog), akkor nyomjuk le az **A** billentyűt, amely megfelel a **Use Entire Disk** (Az egész lemez használata) menüpontnak. A létező slice-ok eltávolításra kerülnek és helyettük megjelenik egy **unused** (üres) jelzésű kis méretű terület (elvégre PC-ről beszélünk), valamint egy nagyobb slice a FreeBSD számára. Ha így jártunk el, akkor válasszuk ki nyilakkal a frissen létrejött FreeBSD slice-ot és az **S** billentyű lenyomásával jelöljük be indíthatónak (bootable). A képernyő ekkor a 2-16 Ábra által mutatotthoz fog erősen hasonlítani. A **Flags** (Beállítások) oszlopban láthatjuk az **A** jelzést, amelyből kiderül, hogy az adott slice *aktív*, tehát róla tud indulni a rendszer.

Ha a FreeBSD számára egy meglevő slice törlésével szeretnénk helyet csinálni, akkor ehhez válasszuk ki nyílbillentyűkkel a használni kívánt slice-ot és nyomjuk le a **D** billentyűt. Ezután nyomjuk le a **C** billentyűt is, amire felbukkan a létrehozandó slice méretét kérdező ablak. Adjuk meg a számunkra megfelelő méretet a számunkra megfelelő formában, majd zárjuk le az **Enter** lenyomásával. Az ablakban szereplő alapértelmezett érték a létrehozható lehető legnagyobb méretű slice-ot adja meg, ami vagy a legnagyobb összefüggő üres terület, vagy pedig az egész merevlemez összterülete lehet.

Ha már korábban készítettünk elő helyet a FreeBSD-nek (például egy **PartitionMagic** vagy egy hozzá hasonló alkalmazás segítségével), akkor csak elegendő az új slice létrehozásához megnyomnunk a **C** billentyűt. Ekkor szintén megkérdezésre kerül a létrehozandó slice mérete.

Ábra 2-16. Particionálás az Fdisk “Using Entire Disk” funkciójával

```

Disk name:      ad0                      FDISK Partition Editor
DISK Geometry: 16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset      Size(ST)      End      Name  PType      Desc  Subtype  Flags
-----
0           63           62      -      6      unused     0
63      16514001      16514063      ad0s1  3      freebsd    165      CA

The following commands are supported (in upper or lower case):

A = Use Entire Disk      G = set Drive Geometry      C = Create Slice      F = 'DD' mode
D = Delete Slice        Z = Toggle Size Units      S = Set Bootable      I = Wizard m.
T = Change Type         U = Undo All Changes      Q = Finish

Use F1 or ? to get more help, arrow keys to select.

```

Amikor befejeztük, nyomjuk le a **Q** billentyűt. Ekkor a **sysinstall** elmenti a beállított értékeket, azonban a lemezre ekkor még nem kerülnek ki.

2.6.3. A rendszerválasztó telepítése

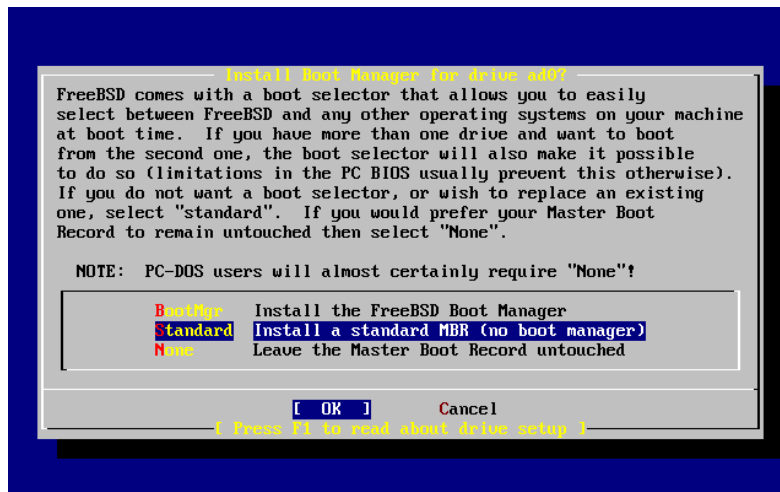
Mindezek után lehetőségünk nyílik telepíteni egy rendszerválasztót (boot manager). Általában véve akkor van szükségünk a FreeBSD rendszerválasztójának telepítésére, ha:

- Egynél több meghajtónk van, és közülük nem az első meghajtóra telepítjük a FreeBSD-t.
- A FreeBSD-t ugyanazon a lemezen más operációs rendszerek mellé telepítjük, és szeretnénk választhatóvá tenni, hogy a számítógép indításakor a FreeBSD vagy a többi operációs rendszer induljon-e el.

Amennyiben a FreeBSD lesz az egyetlen operációs rendszer a gépünkön és az első merevlemez meghajtóra telepítjük, akkor a **Standard** (Szabványos) rendszerválasztó tökéletesen megteszi. Ha viszont a FreeBSD indításához egy másik rendszerválasztót szeretnénk használni, válasszuk a **None** (Nincs) opciót.

Válasszunk, majd nyomjuk le az **Enter** billentyűt!

Ábra 2-17. A sysinstall rendszerválasztókat tartalmazó menüje



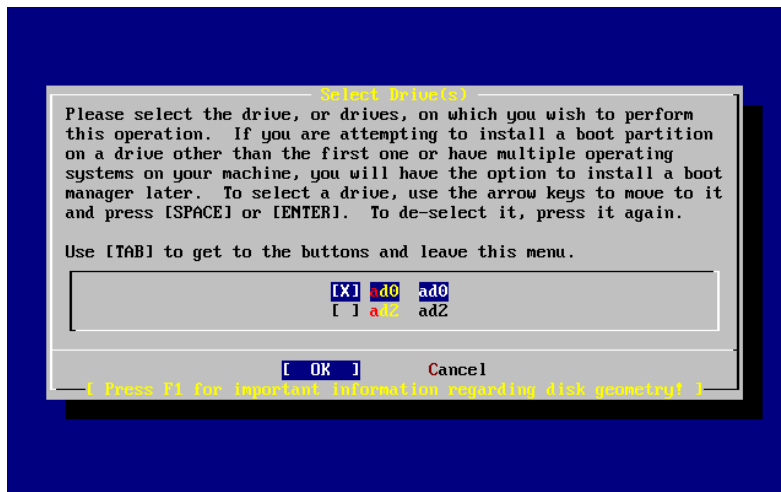
Az **F1** billentyű lenyomásán keresztül elérhető súgóképernyőn olvashatunk az egy merevlemezen több operációs rendszer használatával kapcsolatos problémákról.

2.6.4. Slice-ok létrehozása egy másik meghajtón

Ha egynél több meghajtónk van, a program a rendszerválasztó képernyője után ismét visszatér a meghajtók kiválasztásához. Amennyiben a FreeBSD-t egy másik meghajtóra is telepíteni szeretnénk, itt válasszuk ki azt és ismételjük meg vele az imént az **FDisk** programmal végzett felosztási folyamatot.

Fontos: Amikor a FreeBSD-t nem az első meghajtóra telepítjük, akkor a FreeBSD rendszerválasztóját mind a két meghajtóra telepíteni kell.

Ábra 2-18. Kilépés a meghajtóválasztó menüből



A **Tab** billentyűvel tudunk váltani a legutoljára kiválasztott meghajtó, a [OK] és a [Cancel] gombok között.

Az [OK] gombra álláshoz nyomjuk le egyszer a **Tab**ot, majd a telepítés folytatásához nyomjuk le az **Enter** billentyűt.

2.6.5. Partíciók létrehozása a Disklabel segítségével

A következő lépésként létre kell hoznunk partíciókat a frissen létrehozott slice-okban. Ne felejtsük el, hogy minden partíció rendelkezik egy a-tól h-ig terjedő betűjellel, amelyek közül a b, c és d jelzésűeknek külön szerepe van, amire tekintettel kell lennünk.

Bizonyos alkalmazások kedvelnek egyes partíciókiosztási sémákat, különösen az egynél több lemezen elhelyezkedő partíciókat. Azonban az első FreeBSD telepítésünk során még nem annyira fontos koncentrálnunk a lemezünk hatékony felosztására. Sokkal inkább fontosabb, hogy először egyszerűen csak telepítsük a FreeBSD-t és tanuljuk meg a használatát. Amikor már jobban ismerni fogjuk az operációs rendszert, a partíciók kiosztásának megváltoztatásához mindig újra tudjuk telepíteni a FreeBSD-t.

Ebben a sémában négy partíció szerepel — egy a lapozóállománynak és három az állományrendszereknek.

Táblázat 2-2. Az első lemez partícióinak kiosztása

Partíció	Állományrendszer	Méret	Leírás
a	/	1 GB	Ez a rendszerindításhoz használt, más néven a gyökér állományrendszer (root filesystem). Minden további állományrendszer ehhez csatlakozik valahol. Ennek az állományrendszernek 1 GB méret elfogadható, mivel nem fogunk túlságosan sok adatot tárolni rajta, a FreeBSD telepítője is csak nagyjából 128 MB adatot fog ide tenni. Az így fennmaradó lemezterület felhasználható átmeneti adatok tárolására, illetve a / könyvtárban helyet ad a FreeBSD későbbi változatainak terjeszkedéséhez is.

Partíció	Állományrendszertípus	Méret	Leírás
b	-	RAM mérete x 2-3	A rendszer lapozóállománya a b partíción tárolódik. Itt a megfelelő méret megválasztása egyfajta művészet, azonban minden esetben hasznosnak bizonyulhat, ha tudjuk, hogy méretnek mindig érdemes a fizikai avagy központi memória (RAM) méretének két, esetleg háromszorosát választani. Legyen mindig legalább 64 MB-nyi méretű lapozóállományunk, és ha 32 MB RAM-nál kevesebb van a számítógépünkben, akkor is legalább 64 MB-ra állítsuk be. Ha egynél több lemezünk van, mindegyikre rakhatunk lapozóállományt, ezzel a FreeBSD mindegyikőjüket fel tudja használni lapozásra, amivel pedig gyakorlatilag felgyorsítja a folyamatot. Ilyenkor számoljunk úgy, hogy először meghatározzuk a teljes lapozóállomány méretét (például 128 MB), majd ezt elosztjuk a rendelkezésünkre álló lemezek számával (például kettő). Ebből kiszámítható az egyes lemezek elhelyezendő lapozóállomány mérete, ami most a példánk szerint 64 MB lesz.
e	/var	512 MB-tl 4096 MB-ig	A /var könyvtár foglalja magában az állandó változó naplóállományokat, valamint a többi, adminisztrációhoz használt állományt. Ezek többsége a FreeBSD mindennapos működése közben folyamatosan íródik vagy olvasódik. Ha ezeket az állományokat egy külön állományrendszerre rakjuk, akkor ezzel segítünk a FreeBSD-nek optimalizálni az ilyen állományok elérését anélkül, hogy ez hatással lenne a többi, más hozzáférési gyakorisággal bíró állományra.
f	/usr	A lemez többi része (legalább 8 GB)	Az összes többi állomány többnyire a /usr könyvtárban és annak alkönyvtáraiban helyezkedik el.

Figyelem Az imént megadott értékeket csak példaként adtuk meg és csak a tapasztalt felhasználók számára ajánljuk. A többi felhasználónak inkább a partíciók automatikus kiosztását javasoljuk a FreeBSD partíciószerkesztőjében található `Auto Defaults` opció használatával.

Ha a FreeBSD-t egynél több lemezre telepítjük, akkor a korábban megadott többi slice-ban is létre kell hoznunk partíciókat. Ezt legegyszerűbben úgy tehetjük meg, ha minden lemezen létrehozunk két partíciót: egyet a lapozóállománynak, egyet pedig az állományrendszernek.

Táblázat 2-3. Több lemez partícióinak kiosztása

Partíció	Állományrendszertípus	Méret	Leírás
----------	-----------------------	-------	--------

Partíció	Állományrendszertípus	Méret	Leírás
b	-	Lásd a leírást	Ahogy már korábban is említettük, szét tudjuk osztani a lapozóállományt a lemezek között. Habár az a partíció szabad, a hagyományok mégis azt diktálják, hogy a lapozáshoz használt terület maradjon a b partíción.
e	/diskn	A lemez többi része	A lemez fennmaradó része egyetlen nagy partícióval fedhető le. Ez az e partíció helyett lehetne minden további nélkül az a partíció, azonban a hagyományok szerint az a partíciónak a rendszer gyökér állományrendszerét (/) kell tartalmaznia. Nekünk ugyan nem kellene ezt a megszokást követnünk, azonban a sysinstall viszont így tesz, ezért ezzel a választással csak magunkkal teszünk jót. Az állományrendszer bárhová csatlakoztatható — ebben a példában a lemezeket rendre a /diskn könyvtárakhoz csatoltuk, ahol az n az adott lemez sorszáma. De itt természetesen más rendszert is követhetünk.

A partíciók elrendezésének kigondolása után most már létre is hozhatjuk ezeket a **sysinstall** segítségével. Ekkor a következő üzenetet fogjuk látni:

```

Message
Now, you need to create BSD partitions inside of the fdisk
partition(s) just created. If you have a reasonable amount of disk
space (1GMB or more) and don't have any special requirements, simply
use the (A)uto command to allocate space automatically. If you have
more specific needs or just don't care for the layout chosen by
(A)uto, press F1 for more information on manual layout.
```

```

[ OK ]
[ Press enter or space ]
```

Az üzenet fordítása:

```

Üzenet
Most létre kell hoznunk az fdiskkel nemrég elkészített partíciókban a
BSD-s partíciókat. Ha van hozzá elegendő helyünk (1G vagy több) és
nincs semmilyen különleges elvárásunk, akkor egyszerűen csak osszuk
fel automatikusan az (A)uto paranccsal. Amennyiben azonban ennél
többre lenne szükségünk, vagy csak nincs szükségünk az (A)uto által
felkínált sémára, az F1 lenyomására bővebb információkat is kaphatunk
a kézi kiosztás lehetőségeiről.
```

```

[ OK ]
[ Nyomja le az Enter vagy a Szóköz billentyűt ]
```

Nyomjuk le a **Enter** billentyűt a FreeBSD partíciószerkesztőjének, avagy a **Disklabel** elindításához.

A 2-19 Ábra mutatja a **Disklabel** első elindulásakor megjelenő képet. A képernyő három részre tagolható.

A felső pár sorban a jelenleg használt lemez nevét láthatjuk, valamint azt a slice-ot, ami az általunk létrehozott partíciókat tartalmazza (itt a **Disklabel** a Partition name megnevezéssel hivatkozik a slice-ra). A képernyőn továbbá láthatjuk a slice-ban levő szabad helyet is, vagyis azt a helyet, amely ugyan a slice-hoz tartozik, viszont még nem rendeltünk hozzá partíciót.

A képernyő közepén találhatók az eddig már létrehozott partíciók, az általuk tartalmazott állományrendszerek, azok mérete és az állományrendszerek létrehozására vonatkozó különböző beállítások.

A képernyő alsó harmadában a **Disklabel** programban használható billentyűk felsorolása szerepel.

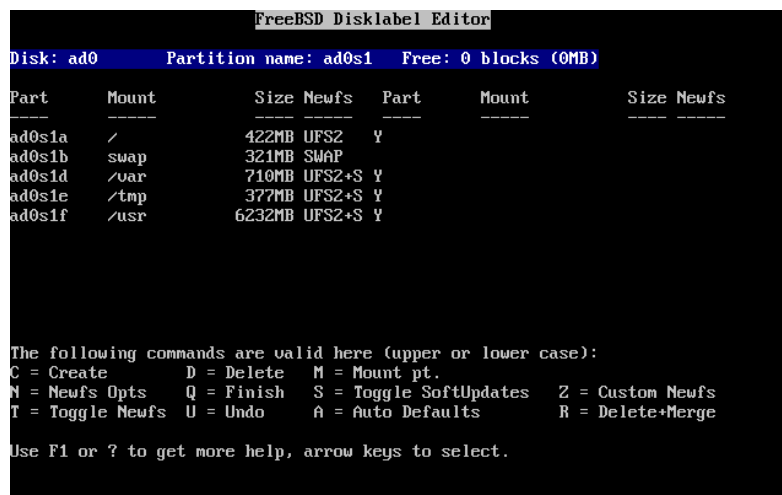
Ábra 2-19. A sysinstall Disklabel partíciószerkesztője



A **Disklabel** képes magától partíciókat készíteni a nekik megfelelő alapértelmezett méretekkel. A partíciók automatikus méretét egy belső partícióméretező algoritmus számítja ki a lemez összmérete alapján. Próbáljuk most mi is ezt ki, és nyomjuk le az **A** billentyűt. Ekkor a 2-20 Ábra szerint illusztráltaknak megfelelő képernyőt tapasztalhatunk. A használt lemez méretétől függően az alapértelmezett értékek megfelelőek lesznek vagy sem. Ez igazából nem számít, hiszen nem kell feltétlenül elfogadnunk az alapértelmezetten megállapított értékeket.

Megjegyzés: Az alapértelmezett partícionálási sémában a `/tmp` könyvtár nem a `/` könyvtár része lesz, hanem saját partíciót kapott. Ezzel igyekszünk elkerülni, hogy a `/` partíció átmenetileg tárolt állományokkal teljen be.

Ábra 2-20. A sysinstall Disklabel partíciószerkesztője, alapértelmezett értékekkel



Ha nem az alapértelmezett partíciókat szeretnénk használni, és le akarjuk váltani ezeket a saját magunk által megadottakra, akkor a nyílbillentyűkkel válasszuk ki az első partíciót és a törléséhez nyomjuk meg a **D** billentyűt. Hasonlóan járunk el az összes többi javasolt partíció törléséhez.

Az első (a, vagyis a / könyvtárként, azaz a gyökérként csatolt) partíció elkészítéséhez először győződjünk arról, hogy a felső sorban a megfelelő slice van kiválasztva, majd nyomjuk meg a **C** billentyűt. Ekkor az új partíció méretét kérdező párbeszédablak jelenik meg (lásd: 2-21 Ábra). Itt a méret a lemez blokkjainak számában adható meg, amit viszont M-mel lezárva megabyte-ban, G-vel gigabyte-ban vagy C-vel cylinderben is kifejezhetünk.

Ábra 2-21. Szabad hely a gyökérpartíción



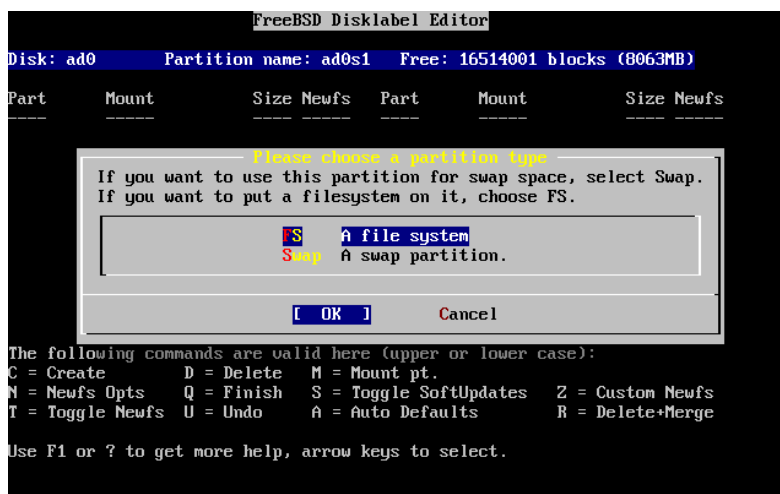
Az alapértelmezés szerint felkínált méret az egész slice-ot lefoglaló partíciót hoz létre. Amennyiben a korábbi példában tárgyalt partícióméreteket kívánjuk használni, akkor a **Backspace** billentyű használatával töröljük ki az így megadott értéket, és helyette gépeljük be, hogy **512M**, ahogy ez a 2-22 Ábra segítségével is látható. A bevittet zárjuk a [OK] gomb lenyomásával.

Ábra 2-22. A gyökérpartíció méretének szerkesztése



Miután meghatároztuk a partíció méretét, a telepítő megkérdezi, hogy a létrehozandó partícióban állományrendszer vagy lapozóállomány foglaljon-e helyet. Ennek a párbeszédablakját a 2-23 Ábra mutatja. Mivel az első partíciónk állományrendszert fog tartalmazni, ezért mindenképpen az **FS** paramétert választjuk ki, majd nyomjuk meg az **Enter** billentyűt.

Ábra 2-23. A gyökérpartíció típusának kiválasztása



Végezetül, mivel egy állományrendszert hoztunk létre, meg kell mondanunk a **Disklabel**nek, hova csatlakoztassa. A hozzá tartozó párbeszédablak a 2-24 Ábrán látható. A gyökér állományrendszer csatlakozási pontja a **/**, ezért itt csak annyit adjunk meg, hogy **/** és zárjuk az **Enter** billentyű lenyomásával.

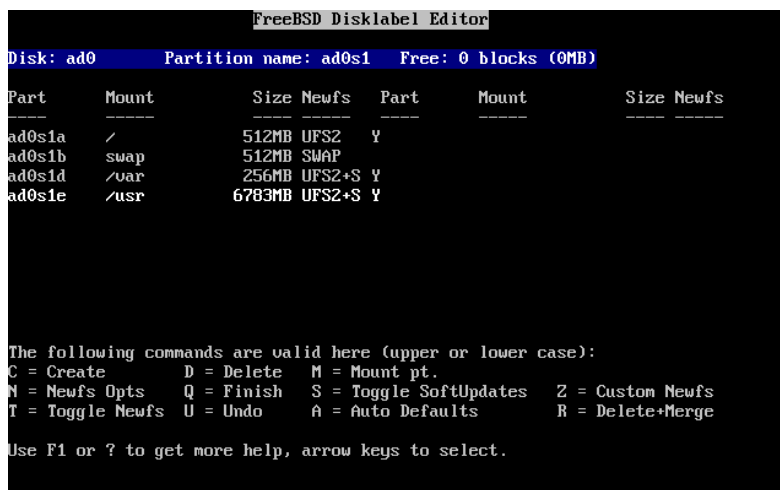
Ábra 2-24. A gyökér csatlakozási pontjának megadása



A képernyőn látható lista ezután az újonnan létrehozott partíciónak megfelelően frissül. A többi partícióra ugyanígy meg kell ismételniünk ezt a műveletsort. Arra azonban figyeljünk, hogy a lapozásra használt partíciót létrehozásánál a szerkesztő nem fogja megkérdezni a csatlakozási pontot, hiszen az ilyen típusú partíciókat sosem csatlakoztatjuk. A /usr, vagyis az utolsó partíció készítése során a slice fennmaradó részének lefoglalásához már nyugodtan meghagyhatjuk a felajánlott értéket.

A FreeBSD partíciószerkesztőjének utolsó képernyője a 2-25 Ábrán hasonlóhoz, habár az általunk választott értékek minden bizonnyal eltérnek. A művelet befejezéséhez nyomjuk le a Q billentyűt.

Ábra 2-25. A Disklabel partíciószerkesztő



2.7. A telepítendő összetevők kiválasztása

2.7.1. A terjesztések típusának kiválasztása

A telepítendő terjesztések típusa nagyban függ attól, hogy a rendszerünket mire szándékozzuk majd használni és mennyi szabad hely áll rendelkezésünkre. Az előre megadott beállítások a lehető legkisebb konfiguráció telepítésétől egészen a komplett rendszer telepítéséig terjednek. A UNIX és/vagy FreeBSD világában még az új felhasználók számára szinte tökéletesen megfelelőnek bizonyulhat az egyik ilyen előkészített beállítás kiválasztása. A terjesztések kiválogatása pedig általában a tapasztaltabb felhasználók számára lehet hasznos.

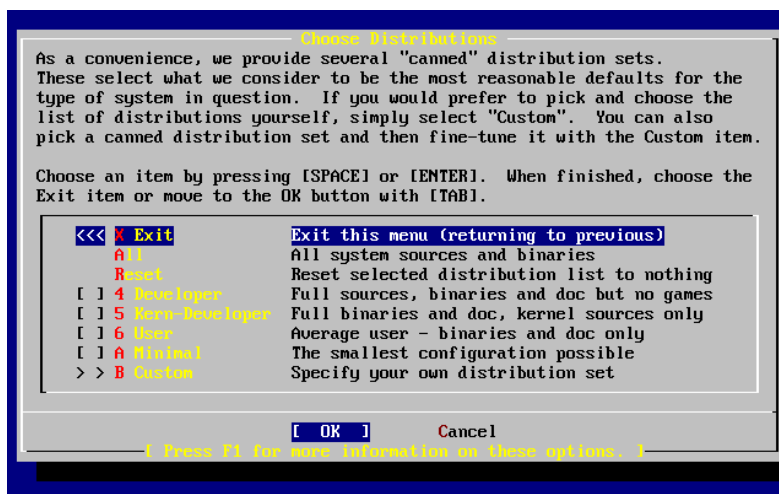
Az **F1** billentyűvel többet is megtudhatunk a terjesztések különböző típusairól és bennük található összetevőkről. Miután befejeztük a súgó áttanulmányozását, nyomjuk le az **Enter** billentyűt, és ezzel visszatérünk a terjesztések kiválasztását tartalmazó menübe.

Ha grafikus felületet szeretnénk használni, akkor az X szerver beállítását az alapértelmezett munkakörnyezet beállítását a FreeBSD telepítése után kell megtenni. Az X szerver beállításáról részletesebben a 5 fejezetben olvashatunk.

Ha egy saját rendszermag építését is fontolgatjuk, akkor olyan terjesztést válasszuk, amiben a forráskód (kernel source) is megtalálható. A saját rendszermag építésének háttéréről és mikéntjéről lásd a 8 fejezetet.

Értelemszerűen a legsokoldalúbb rendszer az, amiben minden megtalálható. Így aztán, ha a lemezünk is megengedi, a nyílak és az **Enter** használatával válasszuk a All (Minden) opciót, ahogy azt az 2-26 Ábra is mutatja. Ha viszont úgy érezzük, hogy ehhez nem eléggé nagy a lemezünk, akkor válasszuk az igényeinkhez jobban illeszkedő típust. Sokat azonban ne üljünk a tökéletes megoldás kiötlésén, hiszen ezek a terjesztések még a telepítés befejezése után is hozzáadhatóak a rendszerünkhöz.

Ábra 2-26. A terjesztések kiválasztása



2.7.2. A Portgyűjtemény telepítése

Miután kiválasztottuk a nekünk megfelelő terjesztést, a telepítőprogram felajánlja a FreeBSD Portgyűjteményének (Ports Collection) telepítésének lehetőségét. A portok gyűjteménye a szoftverek telepítésének egyszerű és kényelmes

módja. A Portgyűjtemény önmaga nem tartalmazza a szoftverek lefordításához szükséges forráskódot, hanem helyette csupán azokat az állományokat, amelyek a különböző külsős programok letöltéséhez, fordításához és telepítéséhez kellenek. A 4 fejezetben megtalálhatjuk, miként is kell használni ezt a gyűjteményt.

A telepítőprogram nem fogja ellenőrizni a kibontásához szükséges helyet, ezért csak abban az esetben válasszuk ezt a lehetőséget, ha mindenképpen elfér a merevlemezünkön. A FreeBSD jelenlegi, 8.1 változatában a Portgyűjtemény nagyjából 417 MB helyet foglal el a lemezen. A FreeBSD frissebb verzióiban nyugodtan feltételezhetünk ennél valamivel nagyobb értéket is.

```

                                User Confirmation Requested
Would you like to install the FreeBSD ports collection?

This will give you ready access to over 20 000 ported software packages,
at a cost of around 417 MB of disk space when "clean" and possibly much
more than that if a lot of the distribution tarballs are loaded
(unless you have the extra CDs from a FreeBSD CD/DVD distribution
available and can mount it on /cdrom, in which case this is far less
of a problem).

The Ports Collection is a very valuable resource and well worth having
on your /usr partition, so it is advisable to say Yes to this option.

For more information on the Ports Collection & the latest ports,
visit:
    http://www.FreeBSD.org/ports

                                [ Yes ]      No

```

Az üzenet fordítása:

```

                                Felhasználói megerősítés szükséges
Szeretné telepíteni a FreeBSD portjainak gyűjteményét?

Ezen keresztül közel 20 000 portolt szoftvercsomaghoz tudunk
könnyedén hozzáférni, amelyek "tisztá" állapotukban nagyjából
417 MB lemezterületünkbe kerülnek, ami a későbbiekben
valószínűleg majd növekedni fog, ahogy letöltjük a különböző
szoftverekhez tartozó állományokat (hacsak nincs meg a FreeBSD
valamelyik CD- vagy DVD alapú terjesztésének az összes lemeze,
amelyeket a /cdrom könyvtárba csatlakoztatva el tudjuk ezeket érni,
mert ekkor kevesebb gondunk lesz vele).

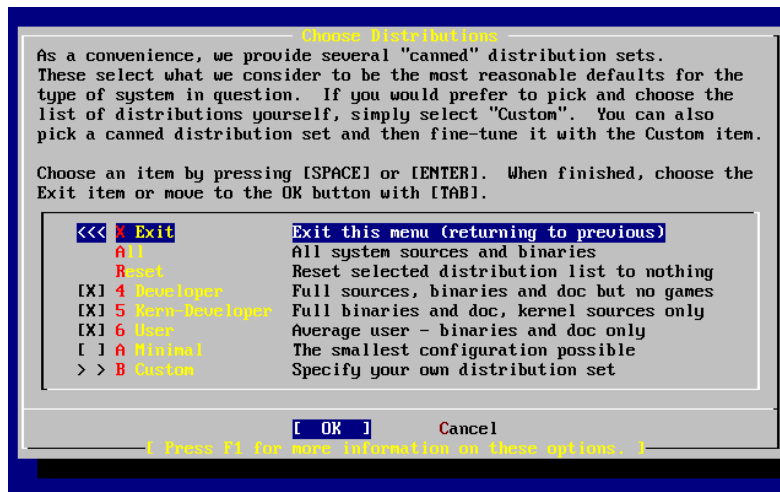
A Portgyűjtemény egy nagyon értékes erőforrás, amelynek megéri helyet
szentelni a /usr partíciónkon, ezért javasoljuk, hogy válassza az
"Igen" opciót. A Portgyűjteményről és annak legújabb portjairól a
http://www.FreeBSD.org/ports oldalon olvashat részletesebben.

                                [ Igen ]      Nem

```

A Portgyűjtemény telepítéséhez a [Yes] gombot, ennek kihagyásához pedig a [No] gombot válasszuk ki a nyilakkal, majd az **Enter** lenyomásával mehetünk tovább. Ekkor a kiválasztott terjesztések menüje fog újra megjelenni.

Ábra 2-27. A terjesztések telepítésének megerősítése



Ha elégedettek vagyunk a beállításokkal, válasszuk ki a nyilakkal az Exit menüpontot, győződjünk meg róla, hogy a [OK] gombon állunk, majd nyomjuk le az **Enter** billentyűt a folytatáshoz.

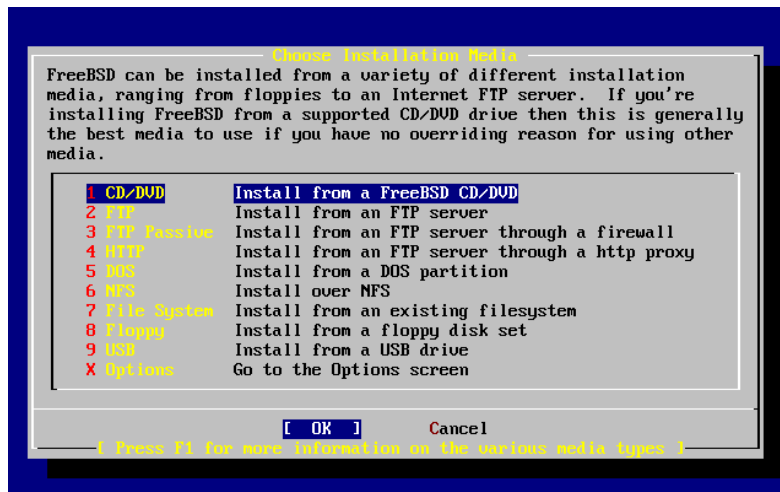
2.8. A telepítés eszközének kiválasztása

Ha CD-ről vagy DVD-ről telepítünk, akkor a következő képernyőn a nyílbillentyűkkel válasszuk ki a Install from a CDROM or DVD (Telepítés CD-ről vagy DVD-ről) menüpontot. Ügyeljünk a [OK] gomb kiválasztására is, majd a telepítés megkezdéséhez nyomjuk meg az **Enter** billentyűt.

A telepítés másfajta módszereinek alkalmazásához válasszuk ki a menüpontok közül a nekünk megfelelőt és kövessük a megjelenő utasításokat.

Az **F1** billentyű lenyomására megjelenik az adott telepítőeszközhöz tartozó súgó. Innen az **Enter** lenyomása után térhetünk vissza a menühöz.

Ábra 2-28. A telepítési eszköz kiválasztása



Telepítés FTP szerverről: Három FTP-s telepítési mód közül választhatunk: aktív, passzív vagy HTTP proxyn keresztül.

Aktív FTP: Install from an FTP server (Telepítés FTP szerverről)

Ezzel a beállítással az összes FTP-n keresztüli átvitel "aktív" módban történik. Ez tűzfalak esetén nem működik, de gyakran alkalmazható olyan régebbi FTP szerverek esetén, amelyek nem ismerik az passzív adatátvitelt. Ha (az alapértelmezett) passzív módban megakadna a kapcsolat, próbáljunk meg helyette az aktívát.

Passzív FTP: Install from an FTP server through a firewall (Telepítés tűzfalon keresztül FTP szerverről)

Ezzel a beállítással a **sysinstall** programot az FTP művelet végrehajtásakor a "passzív" mód használatára utasítjuk. Így át tudunk menni olyan tűzfalakon is, amelyek nem engedik a véletlenszerű TCP portokon érkező kapcsolatokat.

FTP HTTP proxyn keresztül: Install from an FTP server through a http proxy (Telepítés HTTP proxyn keresztül FTP szerverről)

Ezzel a beállítással megmondhatjuk a **sysinstall** programnak, hogy (egy böngészőhöz hasonlóan) a HTTP protokollon keresztül használja az FTP műveletek elvégzéséhez használt proxyt. Ennek a proxynak lesz a feladata az átadott kérések lefordítása és elküldése az FTP szervernek. Ennek köszönhetően át tudunk menni olyan tűzfalakon is, amelyek egyáltalán nem engednek semmilyen FTP műveletet, azonban tartozik hozzájuk egy HTTP proxy. Ilyenkor az FTP szerver beállításai mellett meg kell adnunk ezt a HTTP proxyt is.

Az FTP szerveret proxyn keresztül általában úgy érjük el, hogy a felhasználói név részeként egy "@" jellel elválasztva megadjuk a ténylegesen elérni kívánt szerver nevét. A proxy szerver ezután "helyettesíti" a valódi szervert. Például tegyük fel, hogy a `ftp.FreeBSD.org` szerverről akarunk telepíteni az 1234 porton várakozó `ize.minta.com` proxy használatával.

Ehhez lépünk be a beállításokat tartalmazó menübe, állítsuk az FTP kapcsolathoz használt felhasználói nevet az `ftp@ftp.FreeBSD.org` értékre, majd jelszónak adjuk meg az e-mail címünket. Telepítési eszközként adjuk

meg az FTP-t (vagy a passzív FTP-t, amennyiben a proxy ismeri) és a
ftp://ize.minta.com:1234/pub/FreeBSD címet.

Mivel az ftp.FreeBSD.org címről származó /pub/FreeBSD könyvtár a ize.minta.com szerveren keresztül
érhető el számunkra, ezért lényegében arról a gépről fogunk telepíteni (amely pedig a telepítő kéréseire elhozza
a ftp.FreeBSD.org szerverről az állományokat).

2.9. A telepítés véglegesítése

Ezután ha óhajtjuk, megkezdhetjük a telepítést. Ez egyben az utolsó lehetőségünk a telepítés megszakítására és
mrevlemezünket érintő változtatások érvénytelenítésére.

```

User Confirmation Requested
Last Chance! Are you SURE you want to continue the installation?

If you're running this on a disk with data you wish to save then WE
STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!

We can take no responsibility for lost disk contents!

[ Yes ]    No

```

Az üzenet fordítása:

```

Felhasználói megerősítés szükséges
Utolsó esély: BIZTOSAN folytatni kívánja a telepítést?

Ha olyan lemezre szeretne telepíteni, amelyen fontos adatok
találhatóak, HATÁROZOTTAN JAVASOLJUK, hogy a továbblépés előtt
KÉSZÍTSEN RÓLUK MEGBÍZHATÓ BIZTONSÁGI MÁSOLATOT!

Nem vállalunk semmilyen felelősséget az elvesztett adatokért!

[ Igen ]    Nem

```

A továbblépéshez válasszuk a [Yes] gombot és nyomjuk meg az **Enter** billentyűt.

A telepítés időtartama a kiválasztott terjesztéstől, a telepítésre használt eszköztől és számítógépünk sebességétől
függ. A folyamat előrehaladásáról üzenetek sorozata tájékoztat minket.

A telepítés befejezése után a következő üzenet jelenik meg:

```

Message

Congratulations! You now have FreeBSD installed on your system.

We will now move on to the final configuration questions.
For any option you do not wish to configure, simply select No.

If you wish to re-enter this utility after the system is up, you may
do so by typing: /usr/sbin/sysinstall.

```

[OK]

[Press enter or space]

A szöveg fordítása:

Üzenet

Gratulálunk, sikeresen telepítette a FreeBSD rendszert a számítógépére!

Most rátérünk az utolsó néhány kérdésre. A "Nem" választásával egyszerűen átugorhatjuk mindazt, amit nem szeretnénk beállítani. Ezt a segédprogramot a rendszer újbóli elindítása után a `"/usr/sbin/sysinstall"` parancs begépelésével tudjuk elérni.

[OK]

[Nyomja le az Enter vagy a Szóköz billentyűt]

Az **Enter** billentyű lenyomásával megkezdhetjük a telepítés utáni beállításokat.

A [No] gomb kiválasztásával és az **Enter** lenyomásával megszakíthatjuk a telepítést, így a rendszerünkön semmilyen változtatás nem történik. Ilyenkor a következő üzenet jelenik meg:

Message

Installation complete with some errors. You may wish to scroll through the debugging messages on VT1 with the scroll-lock feature. You can also choose "No" at the next prompt and go back into the installation menus to retry whichever operations have failed.

[OK]

Az üzenet fordítása:

Üzenet

A telepítés során hiba történt. A Scroll Lock használatával érdemes átnézni a VT1 terminál megjelenő üzeneteket. A következő ablakban a "Nem" választásával vissza tudunk menni a telepítőmenühöz és megpróbálkozhatunk ismét a sikertelen műveletek végrehajtásával.

[OK]

Ez az üzenet azért jelent meg, mert semmit sem sikerült telepíteni. Innen az **Enter** megnyomásával térhetünk vissza a főmenübe, majd onnan tudunk kilépni a telepítőből.

2.10. A telepítés után

A sikeres telepítést különféle beállítások követik. Közülük az új FreeBSD rendszer indítása előtt bármelyik megismételhető a beállítások opcióit tartalmazó menü újbóli használatával, vagy pedig a telepítés után a `sysinstall` parancs kiadásával, majd a **Configure** (Beállítások) menüpont kiválasztásával.

2.10.1. A hálózati eszközök beállítása

A következő képernyő már nem jelenik meg, ha az FTP szerveren keresztüli telepítéshez korábban már beállítottuk a PPP kapcsolatot. Ez a korábbiakban említettek szerint állítható be.

Ha többet szeretnénk megtudni a helyi hálózatokról (LAN), vagy a FreeBSD-t átjáróként, illetve útválasztóként kívánjuk beállítani, olvassuk el az Egyéb haladó hálózati témák című fejezetet.

```
User Confirmation Requested
Would you like to configure any Ethernet or PPP network devices?

[ Yes ]   No
```

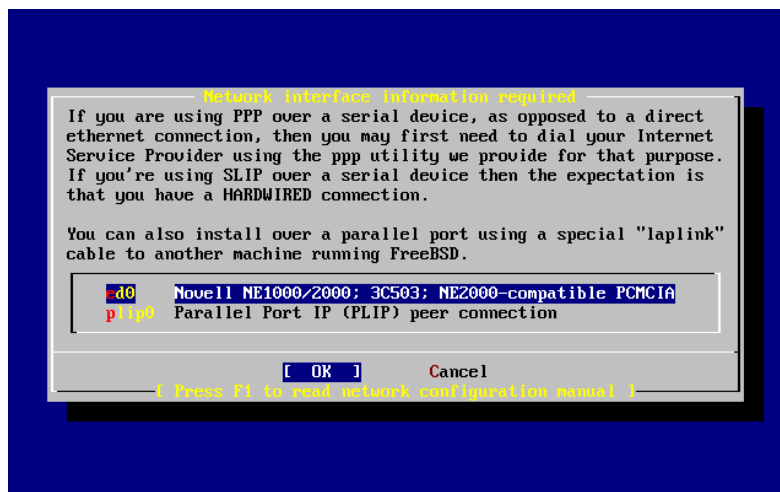
Fordítása:

```
Felhasználói megerősítés szükséges
Szeretnénk beállítani valamilyen Ethernet- vagy PPP hálózati eszközt?

[ Igen ]   Nem
```

A hálózati eszközeink beállításához válasszuk a [Yes] gombot, majd nyomjuk meg az **Enter** billentyűt. Ellenkező esetben a [No] gombbal mehetünk tovább.

Ábra 2-29. Az Ethernet-eszköz kiválasztása



A beállítandó csatló kiválasztásához használjuk a nyílbillentyűket és utána nyomjuk meg az **Enter** billentyűt.

```
User Confirmation Requested
Do you want to try IPv6 configuration of the interface?

Yes    [ No ]
```

Fordítás:

```
Felhasználói megerősítés szükséges
Megpróbálkozik az IPv6 beállításával a csatlón?
```

Igen [Nem]

A példánkban szereplő helyi hálózatban az aktuális internetes protokoll (IPv4) egyelőre megfelelő, ezért válasszuk a [No] gombot és nyomjuk meg az **Enter** billentyűt.

Amennyiben RA-szerveren keresztül egy már létező IPv6 hálózathoz csatlakozunk, akkor válasszuk a [Yes] gombot és nyomjuk meg az **Enter** billentyűt. Ezt követően az RA-szerverek felderítése kezdődik meg, ami néhány másodpercig eltarthat.

```
User Confirmation Requested
Do you want to try DHCP configuration of the interface?
```

Yes [No]

Az üzenet fordítása:

```
Felhasználói megerősítés szükséges
Megpróbálkozik a DHCP használatával a csatolón?
```

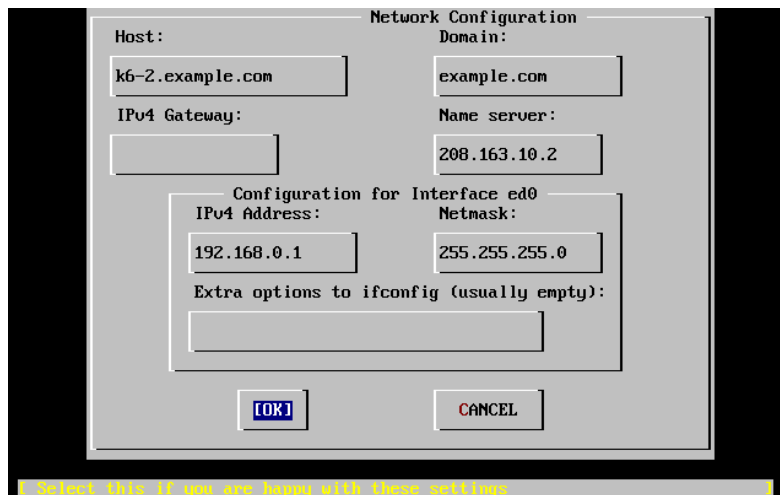
Igen [Nem]

Ha nincs szükségünk a DHCP (Dynamic Host Configuration Protocol, azaz a Dinamikus állomáskonfigurációs protokoll) használatára, akkor a [No] gomb kiválasztásával majd az **Enter** lenyomásával továbbléphetünk.

A [Yes] gomb kiválasztására elindul a **dhclient** nevű program, és amennyiben sikerrel jár, magától kitölti a hálózati beállításokra vonatkozó adatokat. Ennek részleteit a 29.5 Szakaszban találhatjuk meg.

Az alábbi hálózati beállító képernyő mutatja a helyi hálózat átjárójaként használni kívánt Ethernet-eszköz konfigurációját.

Ábra 2-30. Az ed0 hálózati beállítása



A **Tab** billentyűvel tudunk navigálni az adatlap mezői között és kitölteni ezeket a megfelelő információkkal:

Host (Számítógépnév)

A számítógépünk teljes neve, amely a példában most `k6-2.example.com`.

Domain (Tartomány)

Annak a tartománynak a neve, amelyben a számítógépünk a található. Ez itt konkrétan a `example.com`.

IPv4 Gateway (IPv4-átjáró)

A helyben nem elérhető célok megközelítésére használt gép IP-címe. Ezt a mezőt mindenképpen töltsük ki akkor, ha a számítógépünk valamilyen hálózatra van kötve. Azonban *hagyjuk üresen*, ha a számítógép a hálózat átjárója az internet felé. Az IPv4 átjárót más néven “default gateway”-nek (alapértelmezett átjárónak) vagy “default route”-nak (alapértelmezett útvonalnak) is nevezik.

Name server (Névszerver)

A helyi DNS (névfeloldó) szerverünk IP-címe. Ha nem található ilyen a helyi hálózatunkon, akkor az internet-szolgáltató DNS szerverének címét (a példában ez a `208.163.10.2`) adjuk meg.

IPv4 address (IPv4-cím)

A csatoló IP-címe, amely az ábrán a `192.168.0.1`.

Netmask (Hálózati maszk)

A helyi hálózatban használt címtartomány a `192.168.0.0 - 192.168.0.255`, amihez a `255.255.255.0` hálózati maszk tartozik.

Extra options to ifconfig (Az ifconfig további beállításai)

Az `ifconfig` parancs adott csatolóra vonatkozó egyéb beállításai. Jelen esetünkben itt semmi sem szerepel.

Miután végeztünk, a **Tab** billentyű lenyomásával válasszuk ki a [OK] gombot és nyomjuk le az **Enter** billentyűt.

```
User Confirmation Requested
Would you like to bring the ed0 interface up right now?

[ Yes ]    No
```

A fordítás:

```
Felhasználói megerősítés szükséges
Aktiválja most az ed0 csatolót?

[ Igen ]   Nem
```

A [Yes] gomb kiválasztásával, majd az **Enter** lenyomásával csatlakoztatjuk a számítógépet a hálózathoz, ami ezután használhatóvá válik. Ez azonban a telepítés számára nem jelent túlságosan sokat, hiszen ettől függetlenül a számítógépet egyébként is újra kell majd indítanunk.

2.10.2. Az átjáró beállítása

```
User Confirmation Requested
Do you want this machine to function as a network gateway?

[ Yes ]    No
```

A fordítás:

Felhasználói megerősítés szükséges
Ezt a számítógépet hálózati átjáróként is használni akarja?

[Igen] Nem

Ha a számítógépet a helyi hálózat átjárójaként használni akarjuk gépek közti csomagok továbbítására, akkor válasszuk a [**Yes**] gombot és nyomjuk meg hozzá az **Enter** billentyűt. Ha viszont ez a gép csupán a hálózat egy tagja, akkor válasszuk a [**No**] gombot és a folytatáshoz nyomjuk meg az **Enter** billentyűt.

2.10.3. A hálózati szolgáltatások beállítása

User Confirmation Requested
Do you want to configure inetd and the network services that it provides?

Yes [No]

Fordítás:

Felhasználói megerősítés szükséges
Beállítja az inetd démont és az általa felkínált hálózati szolgáltatásokat?

Igen [Nem]

Ha itt a [**No**] gombot választjuk, akkor ezzel kikapcsoljuk a különböző szolgáltatásokat, például a **telnetd** démont. Ez azt jelenti, hogy a távoli felhasználók nem lesznek képesek a **telnet** program használatával belépni erre a számítógépre. A helyi felhasználók viszont továbbra is képesek lesznek távoli számítógépeket elérni a **telnet** segítségével.

Az `/etc/inetd.conf` átirásával azonban ezek a szolgáltatások később természetesen engedélyezhetők. A 29.2.1 Szakasz foglalkozik a téma részleteivel.

A [**Yes**] gomb választásával már a telepítés során beállíthatjuk a szolgáltatásokat. Ekkor egy további párbeszédablak is felbukkan:

User Confirmation Requested
The Internet Super Server (inetd) allows a number of simple Internet services to be enabled, including finger, ftp and telnetd. Enabling these services may increase risk of security problems by increasing the exposure of your system.

With this in mind, do you wish to enable inetd?

[Yes] No

Fordítása:

Felhasználói megerősítés szükséges
A fő internetes kiszolgáló (az inetd) számos egyszerű internetes szolgáltatás, többek közt a finger, ftp és telnet elérését teszi lehetővé. Ezen szolgáltatások engedélyezése azonban a felmerülő biztonsági problémák kockázatát, mivel ezzel rendszerünket jobban kitesszük támadásoknak.

Mindezek tudatában használni kívánja az inetd démonot?

[Igen] Nem

A folytatáshoz válasszuk a [Yes] gombot.

User Confirmation Requested

inetd(8) relies on its configuration file, /etc/inetd.conf, to determine which of its Internet services will be available. The default FreeBSD inetd.conf(5) leaves all services disabled by default, so they must be specifically enabled in the configuration file before they will function, even once inetd(8) is enabled. Note that services for IPv6 must be separately enabled from IPv4 services.

Select [Yes] now to invoke an editor on /etc/inetd.conf, or [No] to use the current settings.

[Yes] No

Fordítás:

Felhasználói megerősítés szükséges

Az inetd(8) démonnak az elérhető internetes szolgáltatások megállapításához szüksége van a beállításait tartalmazó /etc/inetd.conf állományra. A FreeBSD-hez tartozó inetd.conf(5) állomány alapértelmezés szerint az összes szolgáltatást letiltja, ezért a működéséhez minden egyes szolgáltatást külön kell engedélyezni az említett állományban, még abban az esetben is, ha az inetd(8) démon korábban már engedélyeztük. Az IPv6 szolgáltatások az IPv4 szolgáltatásoktól külön engedélyezendőek.

Az [Igen] választásával behívjuk az /etc/inetd.conf szerkesztését, míg a [Nem] választásával pedig az imént felvázolt beállításokat fogadjuk el.

[Igen] Nem

A [Yes] gomb kiválasztásával lehetőségünk nyílik szolgáltatásokat engedélyezni a sorok elején található # jel törlésével.

Ábra 2-31. Az inetd.conf módosítása

```

^_ (escape) menu    ^y search prompt    ^k delete line      ^p prev li          ^g prev page
^o ascii code      ^x search           ^l undelete line    ^n next li          ^v next page
^u end of file      ^a begin of line    ^w delete word      ^b back 1 char
^t top of text      ^e end of line      ^r restore word     ^f forward 1 char
^c command          ^d delete char      ^j undelete char    ^z next word
=====line 1 col 0 lines from top 1=====
# $FreeBSD: src/etc/inetd.conf,v 1.73.10.2.4.1 2010/06/14 02:09:06 kensmith Exp
#
# Internet server configuration database
#
# Define *both* IPv4 and IPv6 entries for dual-stack support.
# To disable a service, comment it out by prefixing the line with '#'.
# To enable a service, remove the '#' at the beginning of the line.
#
#ftp    stream  tcp    nowait  root    /usr/libexec/ftpd    ftpd -l
#ftp    stream  tcp6   nowait  root    /usr/libexec/ftpd    ftpd -l
#ssh    stream  tcp    nowait  root    /usr/sbin/sshd       sshd -i -4
#ssh    stream  tcp6   nowait  root    /usr/sbin/sshd       sshd -i -6
#telnet stream  tcp    nowait  root    /usr/libexec/telnetd telnetd
#telnet stream  tcp6   nowait  root    /usr/libexec/telnetd telnetd
#shell  stream  tcp    nowait  root    /usr/libexec/rshd     rshd
#shell  stream  tcp6   nowait  root    /usr/libexec/rshd     rshd
#login  stream  tcp    nowait  root    /usr/libexec/rlogind  rlogind
#login  stream  tcp6   nowait  root    /usr/libexec/rlogind  rlogind
file "/etc/inetd.conf", 118 lines

```

Miután felvettük az összes használni kívánt szolgáltatást, az **Esc** billentyű lenyomásával előhozhatjuk azt a menüt, ahol elmenthetjük a módosításainkat és kiléphetünk.

2.10.4. Az SSH-n keresztüli bejelentkezés engedélyezése

```

User Confirmation Requested
Would you like to enable SSH login?
Yes          [ No ]

```

Fordítás:

```

Felhasználói megerősítés szükséges
Engedélyezi az SSH-n keresztüli bejelentkezést?
Igen          [ Nem ]

```

A [Yes] gomb kiválasztása engedélyezi az **OpenSSH**-hoz tartozó sshd(8) démont, aminek segítségével a számítógépünkre biztonságosan be tudunk jelentkezni távolról. Az **OpenSSH** részleteiről lásd a 14.11 Szakaszt.

2.10.5. Anonim FTP

```

User Confirmation Requested
Do you want to have anonymous FTP access to this machine?

Yes          [ No ]

```

Fordítás:

```

Felhasználói megerősítés szükséges
Hozzáférhető legyen ez a számítógép anonim FTP használatán keresztül?

Igen          [ Nem ]

```

2.10.5.1. Az anonim FTP tiltása

Az alapértelmezett [No] gomb kiválasztásával és az **Enter** billentyű lenyomásával a jelszóval védett FTP hozzáféréssel rendelkező felhasználók továbbra is elérhetik a számítógépünket.

2.10.5.2. Az anonim FTP engedélyezése

Ha ezt választjuk, akkor anonim FTP kapcsolaton keresztül bárki hozzáférhet a számítógépünkhöz. Ebben az esetben azonban alaposan meg kell fontolnunk néhány biztonsági következményt. A beállítással járó kockázatokról az 14 fejezetben olvashatunk többet.

Az anonim FTP bekapcsolásához a nyílbillentyűkkel válasszuk ki a [Yes] feliratú gombot és nyomjuk meg az **Enter** billentyűt. Ekkor egy további párbeszédablak is megjelenik:

```

User Confirmation Requested
Anonymous FTP permits un-authenticated users to connect to the system
FTP server, if FTP service is enabled. Anonymous users are
restricted to a specific subset of the file system, and the default
configuration provides a drop-box incoming directory to which uploads
are permitted. You must separately enable both inetd(8), and enable
ftpd(8) in inetd.conf(5) for FTP services to be available. If you
did not do so earlier, you will have the opportunity to enable inetd(8)
again later.
```

```

If you want the server to be read-only you should leave the upload
directory option empty and add the -r command-line option to ftpd(8)
in inetd.conf(5)
```

```

Do you wish to continue configuring anonymous FTP?
```

```

[ Yes ]          No
```

Az üzenet fordítása:

Felhasználói megerősítés szükséges

Az anonim FTP használatával a rendszer FTP szolgáltatásához hitelesítetlen felhasználók is hozzáférhetnek, amennyiben az aktív. A névtelen felhasználók az állományrendszernek csak egy részét érhetik el, valamint az alapbeállítások szerint a feltöltést egy külön erre a célra fenntartott könyvtárba végezhetik el. Az FTP szolgáltatás használatát külön engedélyeznünk kell az inetd(8) démon részéről és az inetd.conf(5) állományban található ftpd(8) démon aktiválásával. Ha eddig még nem tettük volna meg, akkor az inetd(8) használatát később még újra engedélyezhetjük.

Ha csak letöltést kívánunk engedni, akkor hagyjuk a feltöltési könyvtárra vonatkozó paramétert üresen és az inetd.conf(5) állományban az ftpd(8) parancssorához adjuk hozzá az -r kapcsolót.

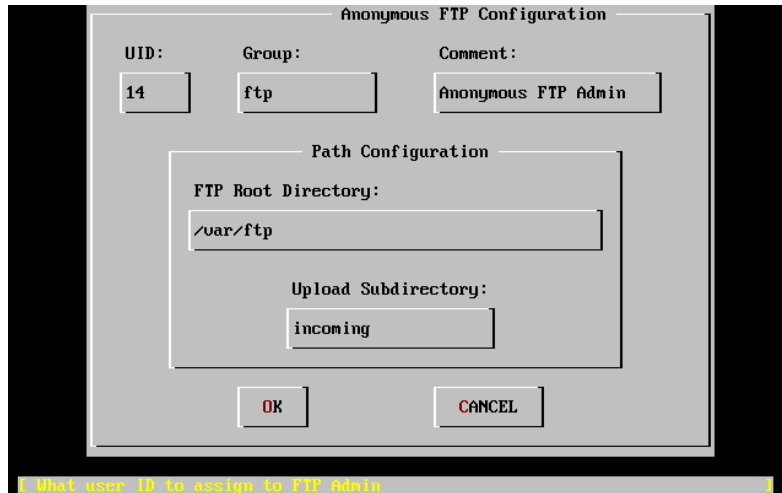
Folytatja az anonim FTP beállítását?

```

[ Igen ]          Nem
```

Az üzenet értesít minket arról, hogy az anonim FTP kapcsolatok engedélyezéséhez az FTP szolgáltatást az `/etc/inetd.conf` állományban is be kell majd kapcsolni, lásd 2.10.3 Szakasz. Válasszuk a [Yes] gombot és a folytatáshoz nyomjuk meg az **Enter** billentyűt. Ekkor a következő képernyő jön elő:

Ábra 2-32. Az anonim FTP alapbeállításai



A beállítások kitöltése során a **Tab** billentyűvel mozoghatunk az adatmezők között:

UID (felhasználói azonosító)

A névtelen FTP felhasználókhöz társított felhasználói azonosító. A feltöltött állomány tulajdonosa ez az azonosító lesz.

Group (csoport)

A névtelen FTP felhasználók csoportja.

Comment (megjegyzés)

Ez a szöveg szerepel a felhasználónál az `/etc/passwd` állományban.

FTP Root Directory (az FTP gyökere)

Itt találhatóak az anonim FTP-n keresztül elérhető állományok.

Upload Subdirectory (feltöltési könyvtár)

A névtelen FTP felhasználók által feltöltött állományok ide kerülnek.

Az FTP gyökere alpból a `/var` könyvtár lesz. Ha a becsült FTP-forgalom lebonyolításához itt nem rendelkezünk elegendő hellyel, akkor az `/usr` könyvtárban található `/usr/ftp` alkönyvtár is beállítható az FTP gyökerének.

Ha elfogadhatónak találjuk az értékeket, nyomjuk le az **Enter** billentyűt a folytatáshoz.

```

User Confirmation Requested
Create a welcome message file for anonymous FTP users?

[ Yes ]    No

```

Fordítás:

```

                                Felhasználói megerősítés szükséges
Létre kíván hozni egy köszöntő üzenetet tartalmazó állományt
az anonim FTP felhasználók számára?

```

```

                                [ Igen ]      Nem

```

A [Yes] választásával és az **Enter** megnyomásával az üzenet szerkesztéséhez egy szövegszerkesztő fog elindulni.

Ábra 2-33. Az FTP köszöntő üzenetének szerkesztése

```

^_ (escape) menu  ^y search prompt  ^k delete line    ^p prev line     ^g prev page
^o ascii code    ^x search         ^l undelete line  ^n next line     ^u next page
^u end of file   ^a begin of line  ^w delete word    ^b back char     ^z next word
^t begin of file ^e end of line    ^r restore word   ^f forward char
^c command       ^d delete char    ^j undelete char
                                     ESC-Enter: exit
=====
Your welcome message here.
=====
file "/var/ftp/etc/ftpnotd", 1 lines, read only

```

Ez az `ee` szövegszerkesztő. Az üzenet átírásához használjuk a megadott utasításokat, de akár később is módosíthatjuk ezt a kedvenc szövegszerkesztőnkkel. Ehhez a módosítandó állomány neve és helye a szerkesztő képernyőjének alján olvasható.

A kilépéshez az **Esc** lenyomására felbukkanó menüben alapból az **a) leave editor** (kilépés a szerkesztőből) menüpont érhető el, ezért itt az **Enter** lenyomásával léphetünk tovább. Az **Enter** ismételt lenyomásával elmenthetjük a módosításainkat.

2.10.6. A hálózati állományrendszer beállítása

A hálózati állományrendszer (Network File System, NFS) állományok közzétételét teszi lehetővé hálózaton keresztül. Használata során egy számítógép beállítható szervernek, kliensnek vagy akár mindkettőnek. Ezzel kapcsolatban a 29.3 Szakasz ajánlott elolvasásra.

2.10.6.1. Az NFS szerver

```

                                User Confirmation Requested
Do you want to configure this machine as an NFS server?

```

```

                                Yes      [ No ]

```

A fordítása:

```
Felhasználói megerősítés szükséges
Be akarja állítani NFS szervernek ezt a számítógépet?
```

Igen [Nem]

Ha nincs szükségünk a hálózati állományrendszer szerver részére, akkor válasszuk a [No] gombot és nyomjuk le az **Enter** billentyűt.

Amennyiben a [Yes] gombot választjuk, egy üzenet fogja közölni velünk, hogy létre kell hoznunk az `exports` állományt.

Message

```
Operating as an NFS server means that you must first configure an
/etc/exports file to indicate which hosts are allowed certain kinds of
access to your local filesystems.
Press [Enter] now to invoke an editor on /etc/exports
[ OK ]
```

Az üzenet fordítása:

Üzenet

Az NFS szerver működtetéséhez először az `/etc/exports` állomány összeállításán keresztül meg kell adnunk, hogy milyen gépek milyen típusú hozzáféréssel rendelkezzenek a helyi állományrendszereinken. Az [Enter] lenyomására megkezdődik az `/etc/exports` állomány szerkesztése.

[OK]

Az **Enter** billentyű lenyomásával továbbléphetünk. Ekkor az `exports` állomány létrehozására és szerkesztésére egy szövegszerkesztő indul el.

Ábra 2-34. Az `exports` szerkesztése

```
^_ (escape) menu ^y search prompt ^k delete line ^p prev li ^g prev page
^o ascii code ^x search ^l undelete line ^n next li ^u next page
^u end of file ^a begin of line ^w delete word ^b back 1 char
^t begin of file ^e end of line ^r restore word ^f forward 1 char
^c command ^d delete char ^j undelete char ^z next word
L: 1 C: 1 =====
#The following examples export /usr to 3 machines named after ducks,
#/usr/src and /usr/ports read-only to machines named after trouble makers
#/home and all directories under it to machines named after dead rock stars
#and, /a to a network of privileged machines allowed to write on it as root.
#/usr          huey louie dewie
#/usr/src /usr/obj -ro  calvin hobbes
#/home  -alldirs  janice jimmy frank
#/a      -maproot=0 -network 10.0.1.0 -mask 255.255.248.0
#
# You should replace these lines with your actual exported filesystems.
# Note that BSD's export syntax is 'host-centric' vs. Sun's 'FS-centric' one.

file "/etc/exports", 12 lines
```

A exportálni kívánt állományrendszerek felsorolásához használjuk képernyőn a megadott utasításokat, vagy tegyük meg ezt később az általunk választott szövegszerkesztő segítségével. Ilyenkor ne felejtsük el megjegyezni az állomány képernyő alján látható nevét és helyét.

Amikor végeztünk, az **Esc** billentyűvel felhozható menüben alából az **a) leave editor** (kilépés a szövegszerkesztőből) menüpont aktív, ezért itt a folytatáshoz egyszerűen nyomjuk le az **Enter** billentyűt.

2.10.6.2. Az NFS kliens

Az NFS kliens beállításával NFS szerverekhez tudunk hozzáférni.

```
User Confirmation Requested
Do you want to configure this machine as an NFS client?

Yes    [ No ]
```

Fordítás:

```
Felhasználói megerősítés szükséges
Beállítja NFS kliensnek ezt a számítógépet?

Igen   [ Nem ]
```

A nyílbillentyűkkel igényeinknek megfelelően válasszuk a [**Yes**] vagy [**No**] gombokat és utána nyomjuk meg az **Enter** billentyűt.

2.10.7. A rendszerkonzol beállításai

Számos beállítás kapcsolódik a rendszerben található konzolok testreszabásához.

```
User Confirmation Requested
Would you like to customize your system console settings?

[ Yes ] No
```

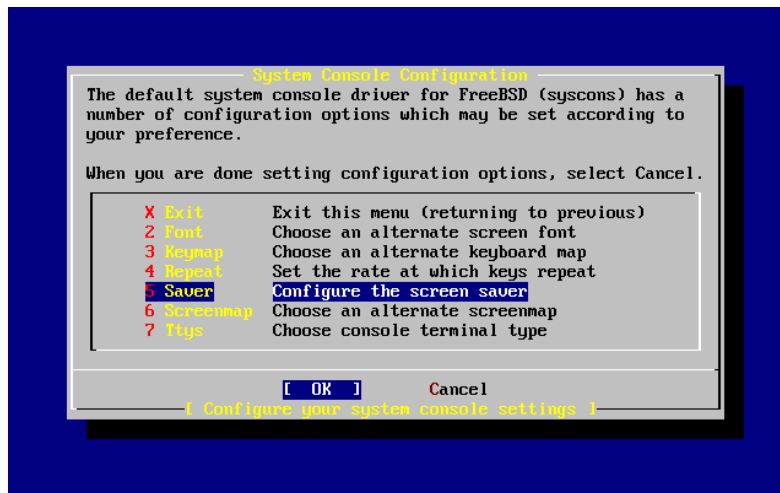
Fordítás:

```
Felhasználói megerősítés szükséges
Testreszabja a rendszerkonzol beállításait?

[ Igen ] Nem
```

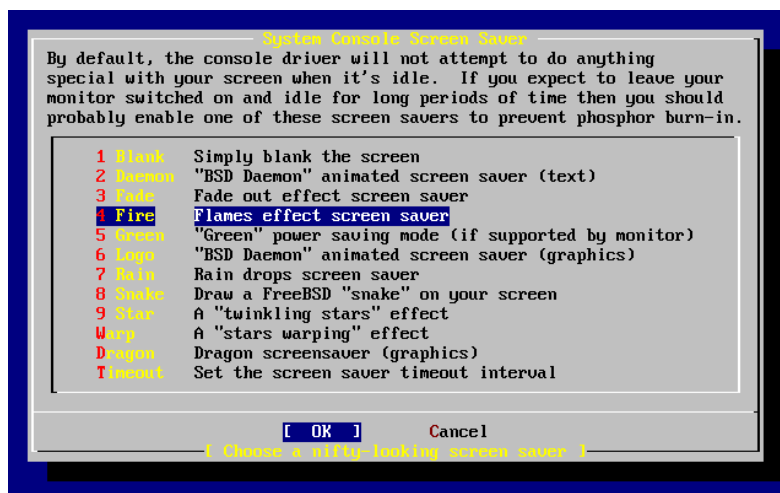
A beállítások megtekintéséhez és megváltoztatásához válasszuk a [**Yes**] gombot és nyomjuk le az **Enter** billentyűt.

Ábra 2-35. A rendszerkonzol beállításai



A képernyőkímélő beállítása egy gyakori opció. A nyílak használatával álljunk a Saver menüpontra, majd nyomjuk le az **Enter** billentyűt.

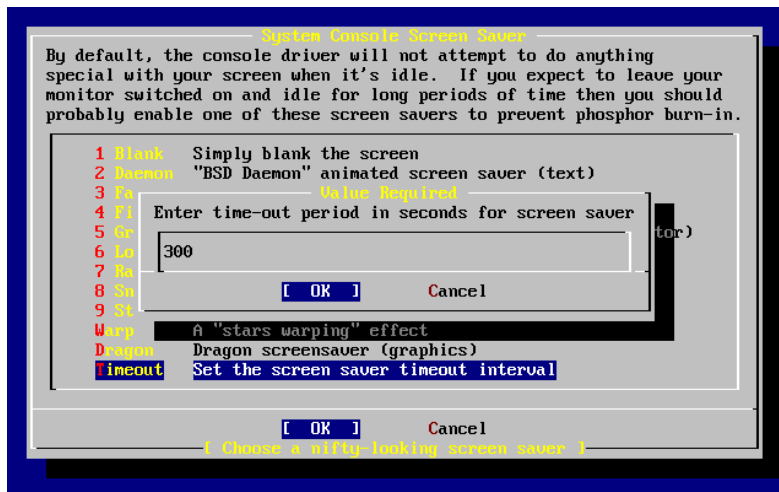
Ábra 2-36. A képernyőkímélő beállításai



A nyilakkal válasszuk ki a használni kívánt képernyőkímélőt és nyomjuk meg hozzá az **Enter** billentyűt. Ekkor a rendszerkonzol beállításait tartalmazó menü jelenik meg ismét.

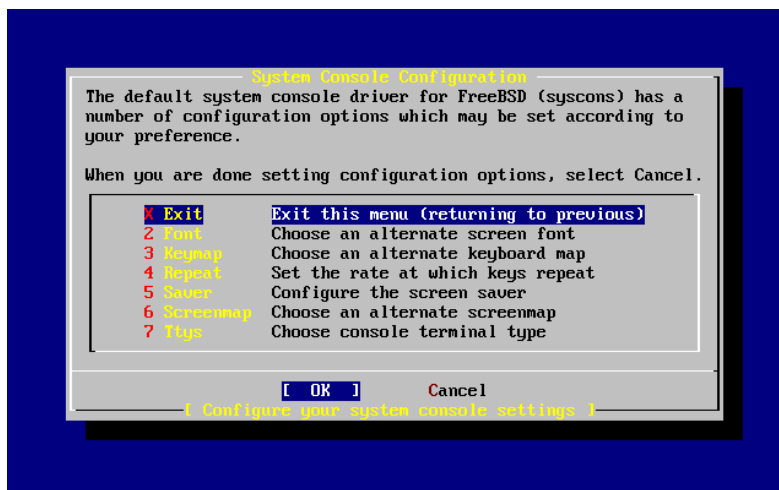
Az aktivizálódás ideje alapbeállítás szerint 300 másodperc. Ennek megváltoztatásához válasszuk ismét a Saver menüpontot. A képernyőkímélő beállításait tartalmazó menüben a nyílbillentyűkkel válasszuk a Timeout (Időkorlát) menüpontot és nyomjuk meg az **Enter** billentyűt. Ekkor egy párbeszédablak jelenik meg:

Ábra 2-37. A képernyőkímélőhöz tartozó időkorlát beállítása



Miután megváltoztattuk az értéket, a rendszerkonzol beállításához a [OK] gomb kiválasztásával, majd az **Enter** billentyű lenyomásával térhetünk vissza.

Ábra 2-38. Kilépés a rendszerkonzol beállító menüjéből



A Exit (Kilépés) választásával és az **Enter** lenyomásával folytathatjuk tovább a telepítés utólagos beállításait.

2.10.8. Az időzóna beállítása

Ha kiválasztjuk számítógépünk számára a megfelelő időzónát, akkor lehetővé tesszük, hogy magától elvégezze a helyi időhöz kapcsolódó összes szükséges korrekciót és helyesen kezelje az időzónákhoz kapcsolódó többi funkciót.

A példában az Egyesült Államok keleti időzónájában elhelyezkedő számítógépet láthatunk. A mi beállításaink természetesen a saját földrajzi helyzetünktől függenek.

```
User Confirmation Requested
Would you like to set this machine's time zone now?

[ Yes ]   No
```

Fordítás:

```
Felhasználói megerősítés szükséges
Beállítja most a számítógép időzónáját?

[ Igen ]   Nem
```

A [Yes] gomb és az **Enter** billentyű segítségével kiválaszthatjuk az időzóna beállítását.

```
User Confirmation Requested
Is this machine's CMOS clock set to UTC? If it is set to local time
or you don't know, please choose NO here!

Yes      [ No ]
```

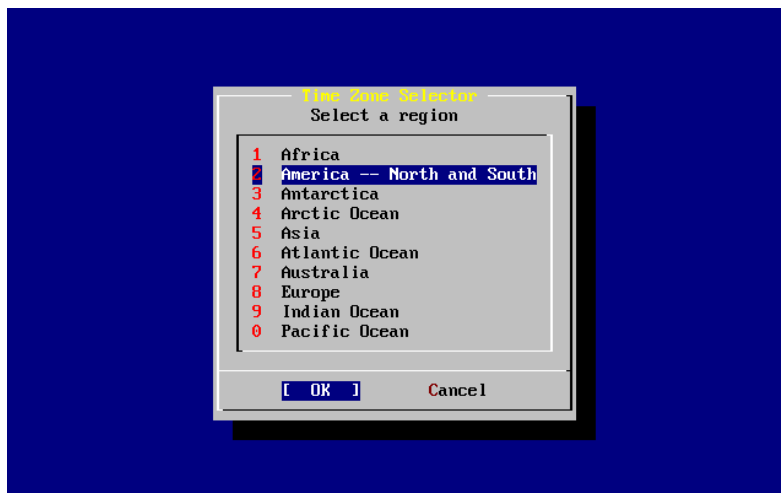
Fordítás:

```
Felhasználói megerősítés szükséges
A számítógép órája az egységes világidőhöz (UTC) van beállítva? Ha a
helyi időhöz vagy nem tudjuk, akkor itt válasszuk a NEM gombot!

Igen     [ Nem ]
```

A számítógépünk órájának beállításának megfelelően válasszuk a [Yes] vagy [No] gombot, és nyomjuk meg az **Enter** billentyűt.

Ábra 2-39. A térség kiválasztása



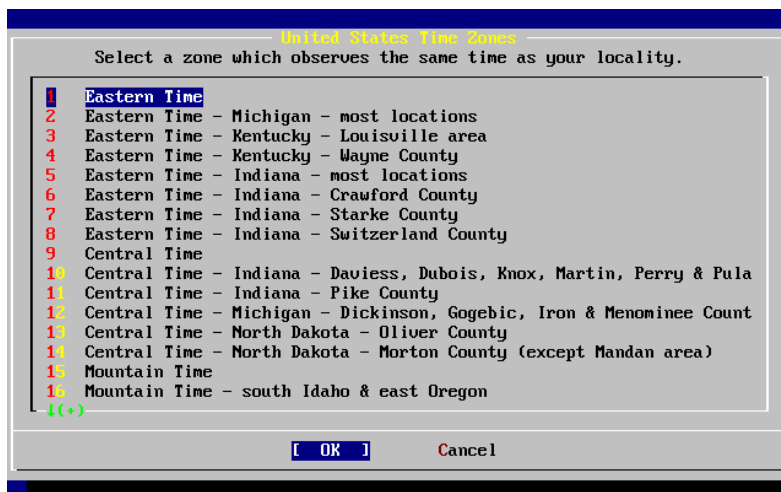
A nyilakkal kiválasztható a megfelelő térség, amit aztán az **Enter** billentyűvel tudunk lezárni.

Ábra 2-40. Az ország kiválasztása



A megfelelő ország a nyílbillentyűkkel, valamint az **Enter** billentyűvel választható ki.

Ábra 2-41. Az időzóna kiválasztása



A nekünk megfelelő időzóna a nyilakkal választható meg, amit ezután az **Enter** billentyűvel tudunk jóváhagyni.

```
Confirmation
Does the abbreviation 'EDT' look reasonable?

[ Yes ]   No
```

Az üzenet fordítása:

```
Megerősítés
Ezek szerint az 'EDT' elfogadható?

[ Igen ]   Nem
```

Erősítsük meg, hogy az időzóna helyes-e. Ha rendbenlevőnek látszik, nyomjuk meg az **Enter** billentyűt a folytatáshoz.

2.10.9. Linux binárisok használata

Megjegyzés: Ez a rész csak a FreeBSD 7.x telepítésére vonatkozik, FreeBSD 8.x esetén ez a képernyő nem jelenik meg.

```
User Confirmation Requested
Would you like to enable Linux binary compatibility?

[ Yes ]   No
```

A fordítás:

```
Felhasználói megerősítés szükséges
Engedélyezi a Linux binárisok futtatását?

[ Igen ]   Nem
```

A [Yes] gomb kiválasztásával és az **Enter** lenyomásával megengedjük, hogy a Linuxra készült szoftvereket futtassunk FreeBSD-n. A telepítő ennek biztosításához még további csomagokat is fel fog rakni.

Ha FTP-n keresztül telepítünk, akkor a számítógépnek csatlakoznia kell az internetre. Ilyenkor előfordulhat, hogy az FTP szerveren nem találhatóak meg a Linux kompatibilitással kapcsolatos csomagok. Ezeket azonban később is telepíthetjük.

2.10.10. Az egér beállításai

Ezen beállítás használatával egy háromgombos egérrel lehetőségünk adódik a konzol és a felhasználói programok között kivágni és bemásolni szövegeket. Kétgombos egér használata esetén nézzük meg a moused(8) man oldalán, miként tudjuk emulálni a háromgombos működést. A következő példa egy nem USB-s (tehát PS/2-es vagy soros portra csatlakozó) egér beállítását illusztrálja:

```
User Confirmation Requested
Does this system have a PS/2, serial, or bus mouse?

[ Yes ]   No
```

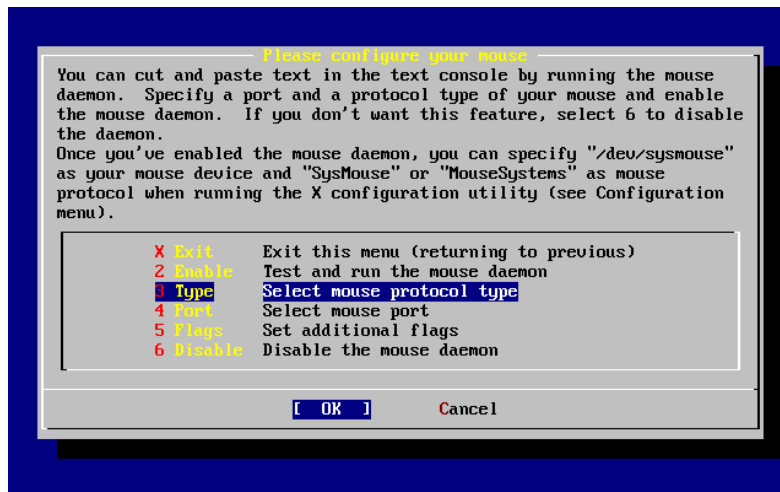
Fordítás:

```
Felhasználói megerősítés szükséges
Csatlakozik a rendszeréhez PS/2-es, soros vagy buszos egér?

[ Igen ]   Nem
```

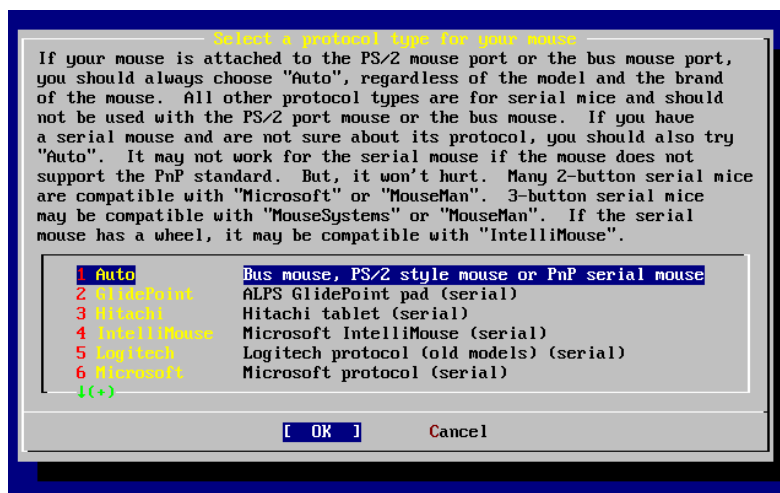
A PS/2, soros vagy buszos egér használatához válasszuk a [Yes] gombot, illetve az USB-s egérhez pedig a [No] gombot, majd nyomjuk meg az **Enter** billentyűt.

Ábra 2-42. Az egér által használt protokoll típusának beállítása



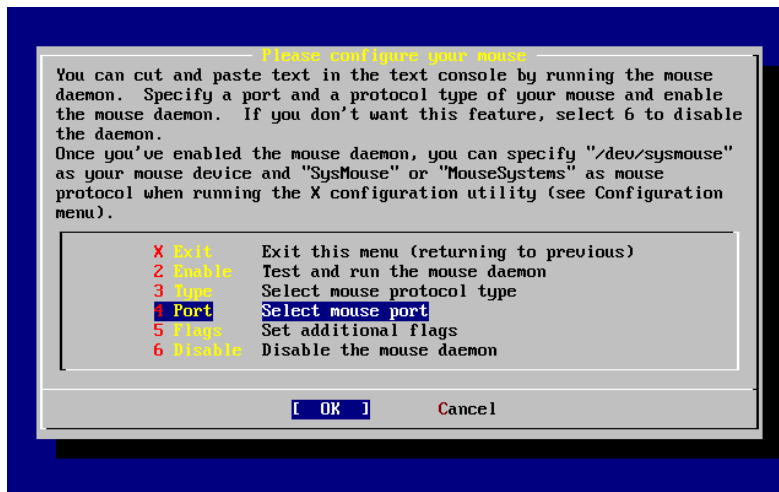
A nyílbillentyűk használatával keressük ki a Type (Típus) menüpontot és nyomjuk le az **Enter** billentyűt.

Ábra 2-43. Az egér protokolljának beállítása



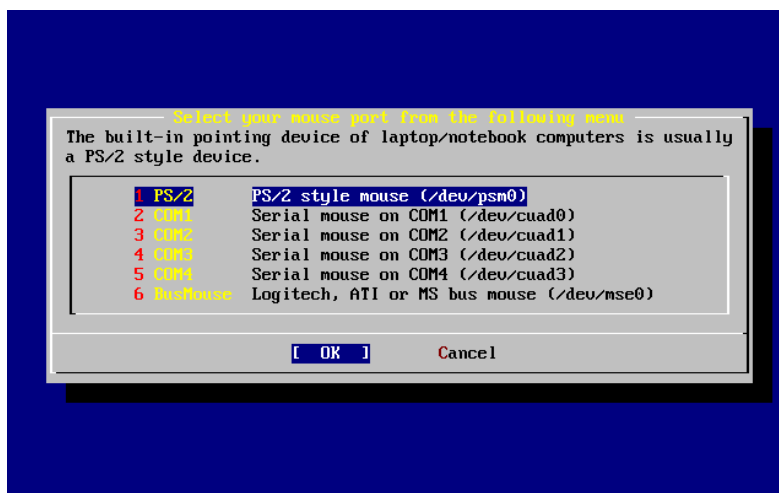
A példában használt egér típusa PS/2, ezért itt a alapértelmezés szerint felkínált Auto megfelelő. A protokoll megváltoztatásához a nyilakkal válasszunk ki egy másikat. Ezután gondoskodjunk róla, hogy az [OK] gombot választottuk ki és a kilépéshez nyomjuk meg az **Enter** billentyűt.

Ábra 2-44. Az egér portjának beállítása



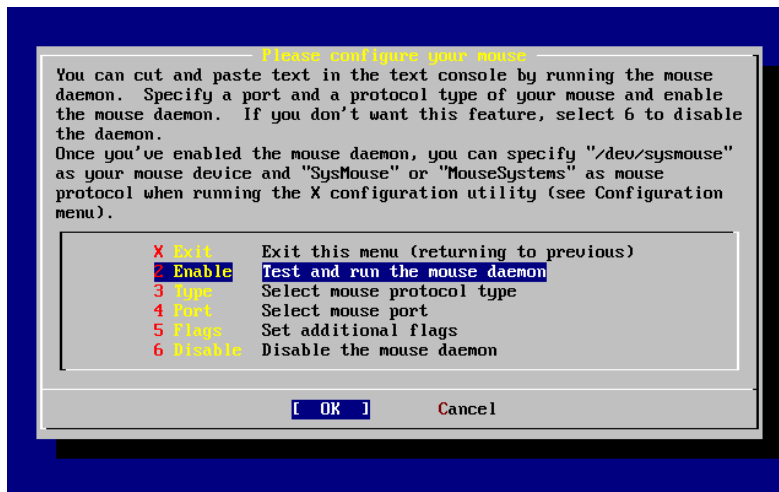
A nyílbillentyűkkel válasszuk ki a Port menüpontot és nyomjuk meg az **Enter** billentyűt.

Ábra 2-45. Az egér portjának kiválasztása



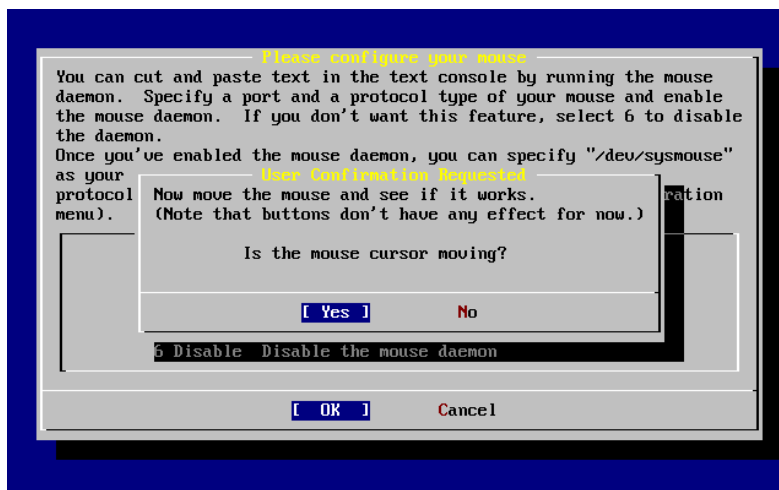
Mivel a példában szereplő rendszerhez egy PS/2 egér csatlakozik, ezért az alapértelmezett PS/2 menüpont megfelelőnek tűnik. A port megváltoztatásához használjuk a nyílakat, majd nyomjuk le az **Enter** billentyűt.

Ábra 2-46. Az egérdémon engedélyezése



Befejezésül a egerhez tartozó démon aktiválásához és kipróbálásához válasszuk ki a nyilakkal az **Enable** (Engedélyezés) menüpontot.

Ábra 2-47. Az egérdémon kipróbálása



Próbáljuk mozgatni a képernyőn megjelenő egérkurzort, és ellenőrizzük, hogy a kurzor a mozdulatainknak megfelelően reagál-e. Ha mindent rendben találunk, akkor válasszuk a [**Yes**] gombot és nyomjuk le az **Enter** billentyűt. Ellenkező esetben az egeret nem jól állítottuk be — válasszuk a [**No**] gombot és kísérletezzünk tovább más beállításokkal.

Az utólagos beállítások folytatásához válasszuk először az **Exit** (Kilépés) menüpontot, majd nyomjuk meg az **Enter** billentyűt.

2.10.11. Csomagok telepítése

A csomagok előre lefordított binárisokat tartalmaznak, és használatukkal igen kényelmesen tudunk szoftvereket telepíteni.

Szemléltetés céljából most bemutatjuk az egyik ilyen csomag telepítését. Természetesen igény szerint más csomagokat is hozzávehetünk. A telepítés után a `sysinstall` parancs használható további csomagok telepítésére.

```

User Confirmation Requested

The FreeBSD package collection is a collection of hundreds of
ready-to-run applications, from text editors to games to WEB servers
and more. Would you like to browse the collection now?

[ Yes ]   No

```

Az üzenet fordítása:

```

Felhasználói megerősítés szükséges

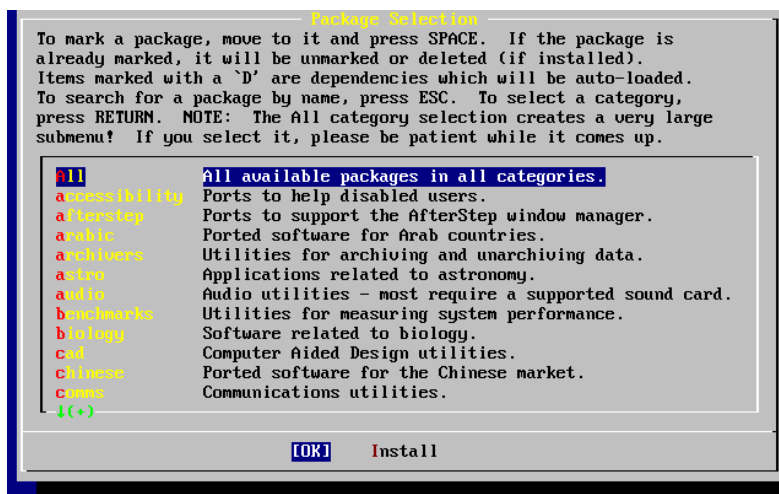
A FreeBSD csomaggyűjteménye többezernyi azonnal használható
alkalmazást tartalmaz, a szövegszerkesztőktől a játékokon keresztül a
WEBSzervereken át szinte mindent. Át kívánja lapozni most ezt a
gyűjteményt?

[ Igen ]   Nem

```

A [Yes] kiválasztása és az **Enter** lenyomása után a csomagválasztó képernyő következik:

Ábra 2-48. A csomagok kategóriájának kiválasztása

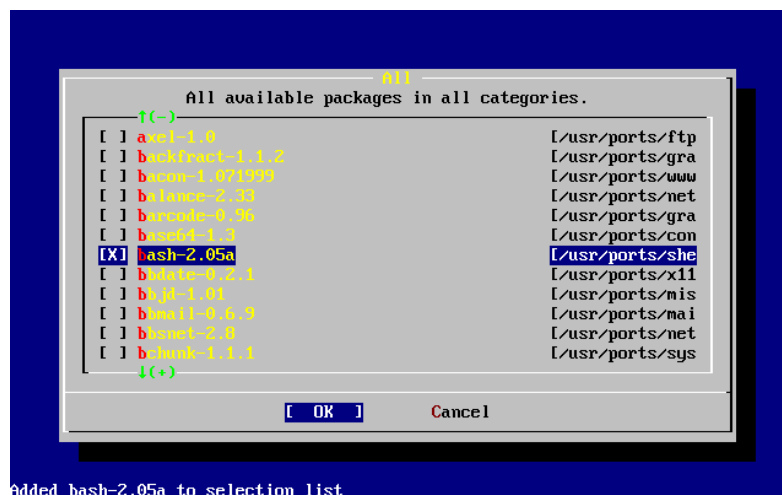


Ekkor csak az adott telepítőeszközön elérhető csomagok fognak megjelenni.

Az összes csomagot az All (Mind) menüpont kiválasztásával láthatjuk, vagy leszűkíthetjük ezt egy adott kategóriára is. Álljunk a kiválasztott kategóriához tartozó menüpontra és nyomjuk meg az **Enter** billentyűt.

Ezután egy menü fogja felsorolni az adott kategórián belül telepíthető csomagokat:

Ábra 2-49. Csomag kiválasztása



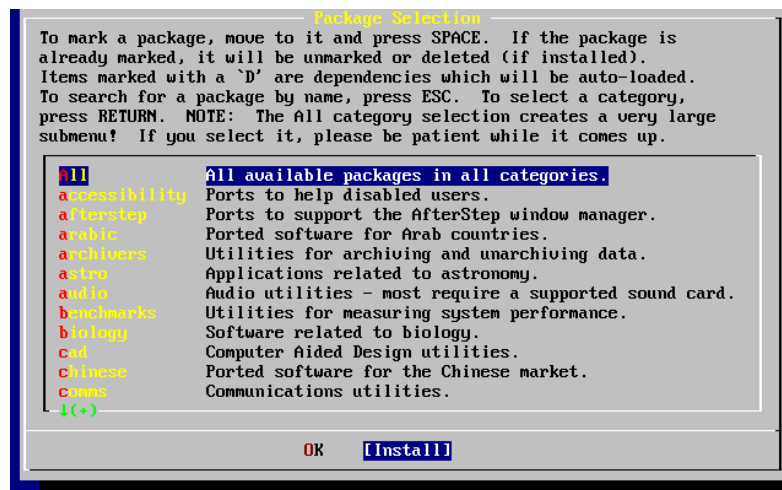
A példában a **bash** parancsértelmezőt választottuk ki. Válogassunk kedvünkre a csomagok között, és álljunk a telepíteni kívántakra, majd a **Szökőz** billentyű lenyomásával jelöljük be ezeket. Minden egyes csomag rövid leírása a képernyő bal alsó sarkában olvasható.

A **Tab** billentyű segítségével mozoghatunk az utoljára kiválasztott csomag, az [OK] és [Cancel] gombok között.

Miután bejelöltük az összes telepítésre szánt csomagot, a csomagválasztó menübe úgy tudunk visszatérni, ha a **Tab** billentyűvel átváltunk az [OK] gombra és nyomjuk meg az **Enter** billentyűt.

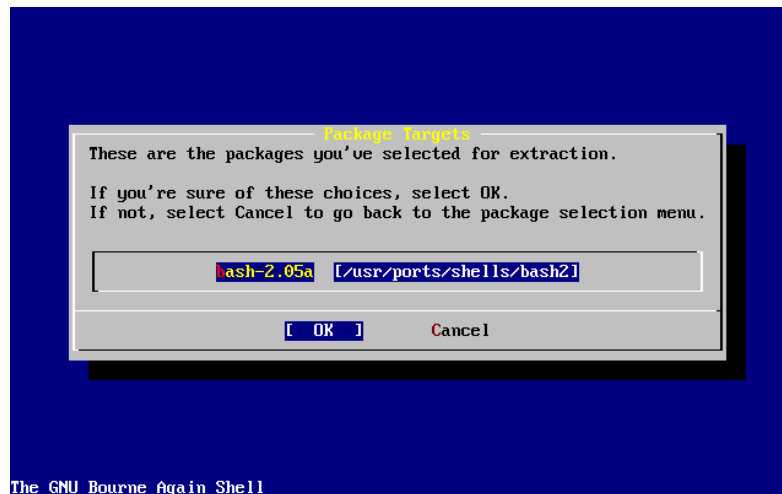
Ezeket felül a bal és jobb nyilak használhatóak az [OK] és [Cancel] gombok közti váltásra. Ugyanezzel a módszerrel választható ki az [OK] gomb is, ami után az **Enter** billentyű megnyomásával visszajutunk a csomagválasztó menübe.

Ábra 2-50. Csomagok telepítése



A nyilakkal és a **Tab** billentyűvel válasszuk ki az [Install] (Telepítés) gombot és nyomjuk meg az **Enter** billentyűt. Ekkor meg kell erősítenünk a csomagok telepítését:

Ábra 2-51. Csomagok telepítésének megerősítése



Az [OK] kiválasztása majd az **Enter** billentyű lenyomása indítja el a csomagok telepítését. A telepítés befejezéséig különböző üzenetek fognak megjelenni. Figyeljünk az ilyenkor felbukkanó hibaiüzenetekre!

A beállítások véglegesítése a csomagok telepítése után folytatódik. Amennyiben egyetlen csomagot sem választottunk és szeretnénk továbblépni, akkor is az Install (Telepítés) gombot válasszuk.

2.10.12. Felhasználók és csoportok felvétele

A telepítés során legalább egy felhasználót érdemes hozzáadnunk a rendszerhez, mivel a rendszer használatához így nem kell root felhasználóként bejelentkezni. Általánosságban véve ahhoz egyébként is kicsi a gyökérpartíció, hogy root felhasználóként (rendszeradminisztrátorként) futtassunk rajta programokat, és gyorsan be is telik. A nagyobb veszélyt azonban itt olvashatjuk:

```
User Confirmation Requested
Would you like to add any initial user accounts to the system? Adding
at least one account for yourself at this stage is suggested since
working as the "root" user is dangerous (it is easy to do things which
adversely affect the entire system).
```

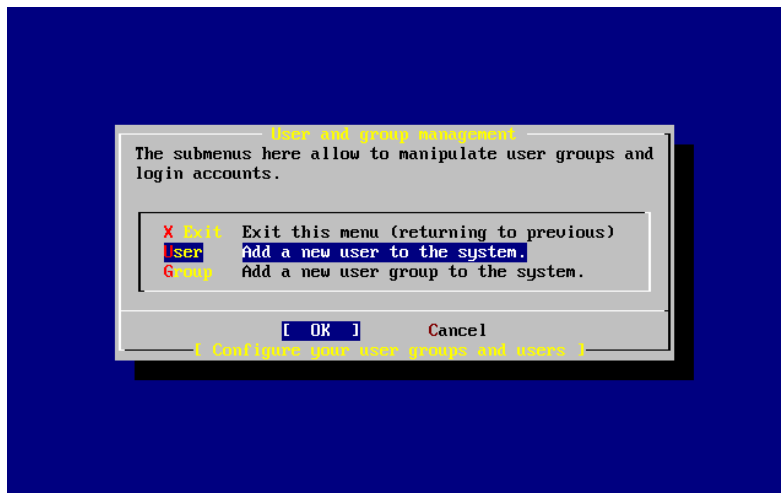
```
[ Yes ]   No
```

```
Felhasználói megerősítés szükséges
Szeretnénk mosta rendszerbe felvenni felhasználói fiókokat? Ebben a
lépésben legalább egy felhasználó felvétele javasolt, hiszen "root"
felhasználóként veszélyes dolgozni (mivel így könnyen tehetünk olyan
dolgot, amelyek káros hatással lehetnek rendszerünkre).
```

```
[ Igen ]   Nem
```

Ezért válasszuk a [Yes] gombot és az **Enter** billentyű lenyomásával lépünk tovább a felhasználók felvételéhez.

Ábra 2-52. Felhasználók kiválasztása



A nyílbillentyűkkel válasszuk ki a User (Felhasználó) menüpontot és nyomjuk meg az **Enter** billentyűt.

Ábra 2-53. A felhasználó adatainak megadása

Amikor a **Tab** billentyűvel lépkedünk a kitöltendő mezők között, a képernyő alsó részén az alábbi leírások magyarázzák az egyes mezők tartalmát:

Login ID (Bejelentkezési azonosító)

Az új felhasználó bejelentkezési neve (kötelező).

UID (Felhasználói azonosító)

A felhasználó számszerű azonosítója (automatikusan létrejön, ha üresen hagyjuk).

Group (Csoport)

A felhasználó bejelentkezési csoportjának neve (automatikusan létrejön, ha üresen hagyjuk).

Password (Jelszó)

A felhasználó jelszava (óvatosan bánjunk ezzel a mezővel!)

Full name (Teljes név)

A felhasználó teljes neve (megjegyzés).

Member groups (További csoportok)

A felhasználó ezen csoportoknak is tagja (tehát rendelkezik az engedélyeikkel).

Home directory (Felhasználói könyvtár)

A felhasználó saját könyvtára (ha üresen hagyjuk, az alapértelmezés szerint töltődik ki).

Login shell (Parancsértelmező)

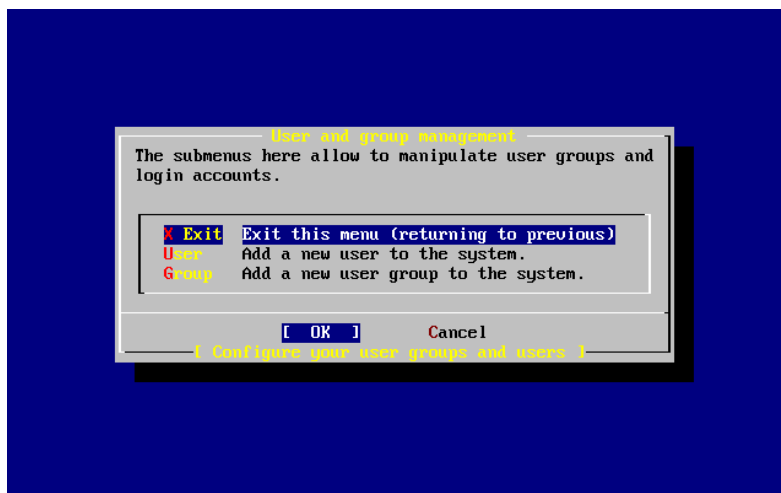
A felhasználó által használt parancsértelmező (ha üresen hagyjuk, az alapértelmezés szerint töltődik, mint például `/bin/sh`).

Az ábrán a bejelentkezés után használt parancsértelmezőt a `/bin/sh` parancsértelmezőről a `/usr/local/bin/bash` parancsértelmezőre változtattuk, így most a korábban telepített **bash** parancsértelmezőt fogjuk használni. Itt ne is próbáljunk nem létező parancsértelmezőt kiválasztani, hiszen ekkor nem tudunk majd bejelentkezni. A BSD világban egyébként a C shell a leggyakrabban használt, amelyet a `/bin/tcsh` megadásával választhatjuk ki.

Az ábrán szereplő felhasználót ezenkívül még a `wheel` csoportba is felvettük, aminek köszönhetően képes lesz a rendszerünkben a `root` felhasználói jogaival rendelkező rendszeradminisztrátorrá válni.

Amikor mindent megfelelőnek találunk, nyomjunk az [OK] gombra és ekkor ismét a felhasználók és csoportok karbantartását tartalmazó menü jelenik meg:

Ábra 2-54. Kilépés a felhasználók és csoportok menüjéből



Csoportokat is létre tudunk hozni, amennyiben erre szükségünk lenne. Ez a rész a telepítés befejezése után továbbra is elérhető a `sysinstall` parancs segítségével.

Amikor befejeztük a felhasználók hozzáadását, a nyilakkal válasszuk ki az **Exit** (Kilépés) menüpontot és a telepítés folytatásához nyomjuk meg az **Enter** billentyűt.

2.10.13. A root felhasználó jelszavának megadása

```

Message
Now you must set the system manager's password.
This is the password you'll use to log in as "root".

```

```
[ OK ]
```

```
[ Press enter or space ]
```

Fordítása:

```

Üzenet
Most meg kell adnia a rendszergazda jelszavát. Ezt a jelszót
kell a "root" felhasználó bejelentkezésekor használni.

```

```
[ OK ]
```

```
[ Nyomja le az Enter vagy a Szóköz billentyűt ]
```

A root felhasználó jelszavának beállításához nyomjuk meg az **Enter** billentyűt.

A jelszót kétszer kell megadnunk. Felesleges megemlíteni, hogy gondoskodjunk arról az esetről is, ha véletlenül elfelejténénk ezt a jelszót. Megemlítjük, hogy az itt begépel jelszó nem lesz látható és a betűk helyett sem jelennek meg csillagok.

```

New password:
Retype new password :

```

A jelszó sikeres megadása után a telepítés folytatódik.

2.10.14. Kilépés a telepítőből

Ha be szeretnénk még állítani egyéb hálózati szolgáltatást vagy valamilyen más konfigurációs lépést kívánunk még elvégezni, ezen a ponton megtehetjük vagy a telepítés után a `sysinstall` parancs kiadásával.

```

User Confirmation Requested
Visit the general configuration menu for a chance to set any last
options?

```

```
Yes [ No ]
```

Fordítás:

```

Felhasználói megerősítés szükséges
Végignézi még utoljára a beállításokat arra az esetre, ha véletlenül

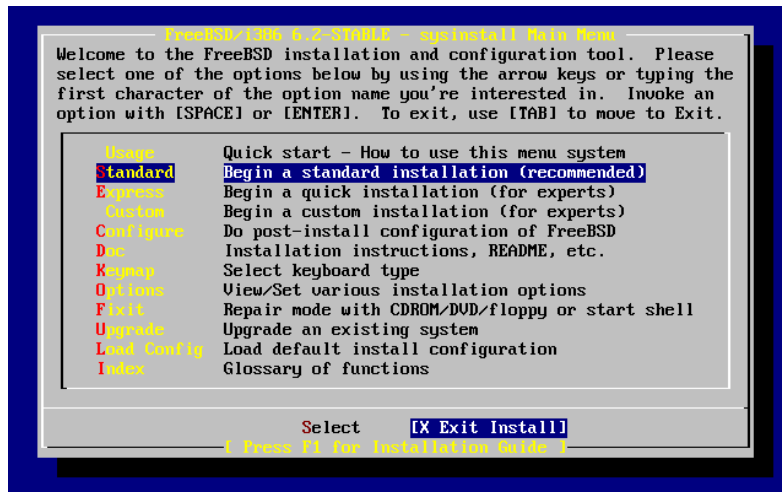
```

kihagytunk volna valamit?

Igen [Nem]

Ha a nyilakkal a [No] gombot választjuk, majd megnyomjuk rajta az **Enter** billentyűt, akkor visszatérünk a telepítő főmenüjébe.

Ábra 2-55. Kilépés a telepítőből



Válasszuk ki a nyílbillentyűkkel a [X Exit Install] (Kilépés a telepítőből) gombot és nyomjuk meg az **Enter** billentyűt. Ezután meg kell erősítenünk kilépési szándékunkat:

User Confirmation Requested

Are you sure you wish to exit? The system will reboot.

[Yes] No

Fordítás:

Felhasználói megerősítés szükséges

Valóban ki akar lépni? A rendszer ezt követően újra fog indulni!

[Igen] Nem

Válasszuk a [Yes] gombot. Ha CD-meghajtóról indítottuk a telepítést, akkor a következő üzenet fog figyelmeztetni minket a lemez kivételére:

Message

Be sure to remove the media from the drive.

[OK]

[Press enter or space]

Fordítás:

Üzenet

Ne felejtsük el kivenni a CD-lemezt a meghajtóból.

[OK]

[Nyomjunk Entert vagy szöközt]

A CD-meghajtó egészen az újraindítás megkezdéséig zárolt lesz, ezért csak ekkor tudjuk (gyorsan) kivenni a meghajtóból a lemezt. Nyomjuk meg az [OK] gombot az újraindításhoz.

A rendszer újraindul, legyünk résen és figyeljük a megjelenő hibaüzeneteket, erről bővebben lásd a 2.10.16 Szakaszban.

2.10.15. További hálózati szolgáltatások beállítása

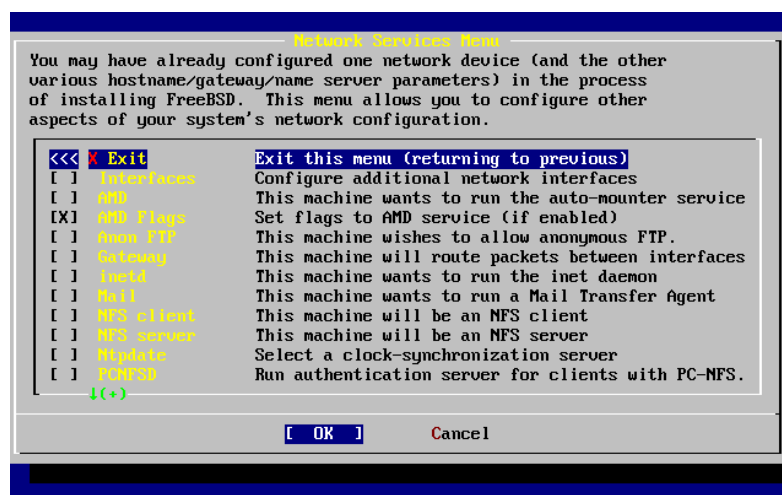
Írta: Tom Rhodes.

A hálózati szolgáltatások terén csekély tapasztalattal rendelkező kezdő felhasználók számára ijesztő lehet ezek beállítása. A hálózatok és többek közt az internet kezelése napjaink modern operációs rendszereink, így a FreeBSD-nek is az egyik fontos területe. Ezért nagyon hasznos ismernünk valamennyire a FreeBSD által felkínált hálózati lehetőségeket. A telepítés közben ezért a felhasználónak tisztában kell lennie a rendelkezésre álló szolgáltatásokkal.

A hálózati szolgáltatások olyan programok, amelyek a hálózat minden részéről fogadnak adatokat. Mindent el kell követnünk annak érdekében, hogy ezek a programok ne tehessenek semmilyen “kárt”. Sajnos a programozók sem tökéletesek, és az idők során már előfordult párszor, hogy a hálózati szolgáltatásokban maradtak hibák, amelyek kihasználásával a támadók rossz dolgokat tudtak csinálni. Ezért fontos, hogy csak is azokat a szolgáltatásokat engedélyezzük, amelyekre ténylegesen szükségünk van. Ha nem tudjuk eldönteni, akkor az a legjobb, ha egészen addig egyiket sem engedélyezzük, amíg valóban szükségünk nem lesz rájuk. A **sysinstall** újbóli elindításával vagy az `/etc/rc.conf` megfelelő beállításával mindig tudunk új szolgáltatásokat aktiválni.

A Networking (Hálózatok) menüpont kiválasztása után valami ilyesmit láthatunk:

Ábra 2-56. A hálózati beállítások menüjének felső szintje



Ezek közül a **Interfaces** (Csatolók), vagyis az első menüpontról korábban már szó esett a 2.10.1 Szakaszban, ezért ez most nyugodtan kihagyható.

Az **AMD** menüpont kiválasztásával engedélyezzük a BSD automatikus csatlakoztatásokért felelős segédeszközt (AMD, az AutoMounter Daemon). Ezt általában az NFS protokollal (lásd lentebb) együtt szokás használni a távoli állományrendszerek automatikus csatlakoztatásához. Itt nincs szükség semmilyen különleges beállításra.

A következő sorban az **AMD Flags** (Az AMD beállításai) menüpont szerepel. Kiválasztása után az AMD beállításait bekérő ablak fog felbukkani. Ez már számos alapértelmezett beállítást tartalmaz:

```
-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map
```

A `-a` kapcsolóval adjuk meg a csatlakozási pontok alapértelmezett helyét, amely ebben az esetben az `/.amd_mnt`. A `-l` kapcsolóval adjuk meg az alapértelmezett `log` (napló) állományt, habár a `syslogd` használata során az összes naplózási tevékenység a rendszer naplózó démonján fut majd keresztül. A `/host` könyvtárba fognak csatlakozni a távoli gépek exportált állományrendszerei, míg a `/net` könyvtárba a különböző IP-címekről exportált állományrendszerek kerülnek csatlakoztatásra. Az `/etc/amd.map` állomány tartalmazza az AMD exportjainak alapértelmezett beállításait.

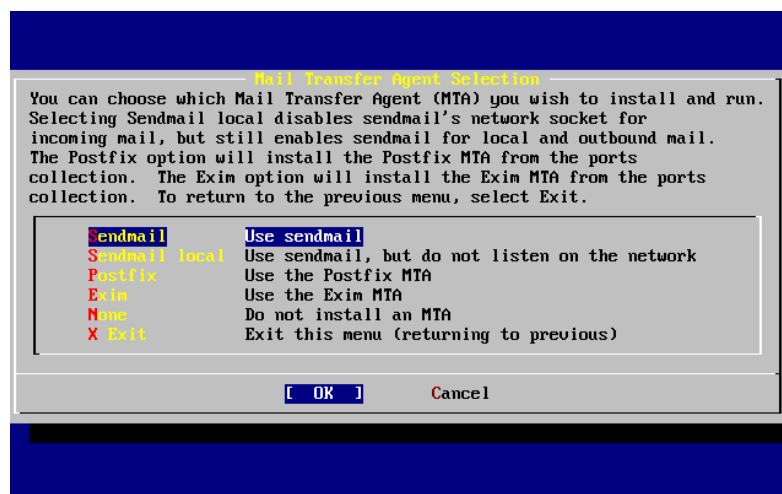
Az **Anon FTP** menüponton keresztül engedélyezhetjük az anonim FTP kapcsolatokat. A menüpont kiválasztásával számítógépünket egy anonim FTP szerverré tehetjük, azonban legyünk tekintettel a beállításhoz tartozó biztonsági veszélyekre! A kiválasztásakor egy ablak tájékoztat minket a beállítás részleteiről és felmerülő biztonsági kockázatokról.

A **Gateway** (Átjáró) menüpont használatával a korábbiakban tárgyaltak szerint állíthatjuk be számítógépünket hálózati átjárónak. Ugyanekkor a **Gateway** menüben nyílik lehetőségük kikapcsolni ezt a beállítást, amennyiben a telepítési folyamat korábbi lépései során véletlenül engedélyeztük volna.

Az **Inetd** menüpont segítségével beállíthatjuk, vagy akár teljesen ki is kapcsolhatjuk a korábban tárgyalt `inetd(8)` démon.

A **Mail** (Levelezés) menüpontban beállíthatjuk a rendszer alapértelmezett MTA avagy levéltovábbító ügynökét (Mail Transfer Agent). Ennek hatására a következő menü jelenik meg:

Ábra 2-57. Az alapértelmezett MTA kiválasztása



Itt választhatunk, hogy a különböző levélküldő rendszerek közül melyiket telepítsük alapértelmezettként. Egy ilyen alkalmazás lényegében nem több, mint egy levélküldésre használt szerver, amely továbbítja a rendszerben vagy az interneten található felhasználók számára a leveleket.

A Sendmail választásával a FreeBSD alpból felkínált megoldását, a népszerű **sendmail** szervert telepíthetjük. A Sendmail local (Helyi Sendmail) menüpont kiválasztásával szintén a **sendmail** lesz a telepítendő levélküldő szerver, azonban nem lesz képes az internetről érkező leveleket fogadni. Az itt felsorolt többi beállítás, tehát a Postfix és Exim, a Sendmail beállításához hasonlóan zajlik. Mind a kettő elektronikus levelek kézbesítésére használható, azonban bizonyos felhasználók a **sendmail** helyett inkább ezek valamelyikét használják.

Valamelyik vagy éppen semelyik levéltovábbító szerver kiválasztása után az NFS client (NFS kliens) beállítására vonatkozó menü jelentkezik.

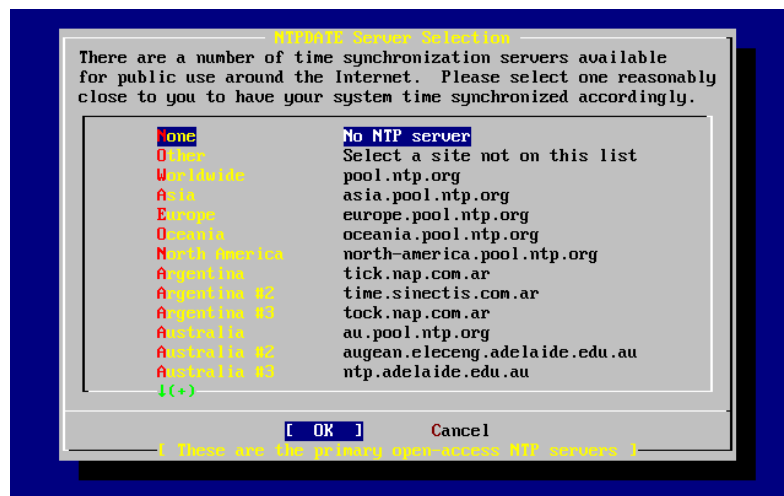
Az NFS client beállításával a rendszerünk NFS szerverekkel lesz képes kapcsolatba lépni. Egy ilyen NFS szerver az NFS protokoll segítségével a hálózaton keresztül elérhetővé tesz állományrendszereket. Ha gépünk független, akkor nem fontos kiválasztanunk ezt a menüpontot. A rendszernek később további beállításokra is szüksége lehet, amelyekről az 29.3 Szakaszban olvashatunk részletesebben.

Az NFS server (NFS szerver) menüpont kiválasztásával hozzájárulunk, hogy rendszerünk NFS szerverként üzemeljen. Ehhez meg kell adnunk az RPC, vagyis a távoli eljáráshívások kiszolgálásának elindításához szükséges adatokat is. Az RPC használatával a különböző kiszolgálók és programok között tudjuk vezérelni a kapcsolatot.

A sorban az Ntpdate beállítása következik, ahol az időszinkronizációhoz kapcsolódó opciókat találjuk.

Kiválasztásakor az ábrán szereplőhöz hasonló menü fog megjelenni:

Ábra 2-58. Az Ntpdate beállítása

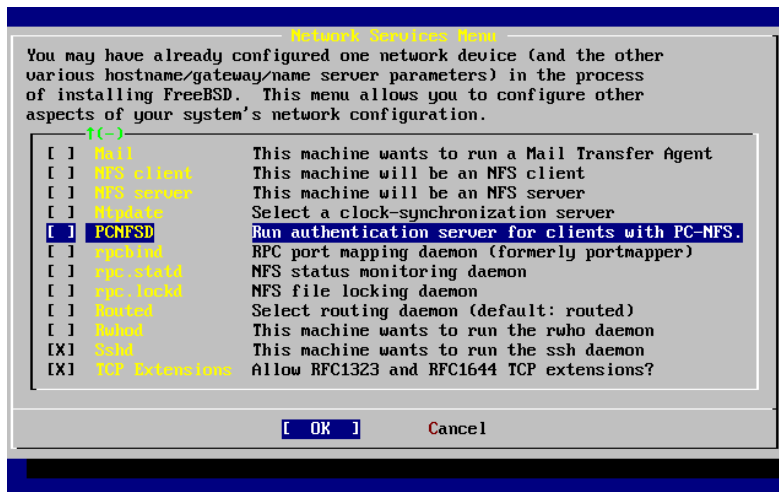


Ebből a menüből válasszuk ki a hozzánk legközelebb levő szervert. Egy közeli szerver megadásával az időszinkronizáció sokkalta pontosabbá válik, mivel a tőlünk távolabbi szerverek kapcsolatának késleltetése nagyobb lehet.

A következő beállítás az PCNFSD. Ennek kiválasztása során a Portgyűjteményből telepítésre kerül a `net/pcnfsd` csomag. Ez lényegében egy hasznos segédprogram, amellyel olyan operációs rendszerek számára tudunk hitelesítést szolgáltatni az NFS használata során, amelyek maguktól erre nem képesek, mint például a Microsoft MS-DOS rendszere.

A többi beállítás megtekintéséhez egy kicsit lejjebb kell haladnunk a listában:

Ábra 2-59. A hálózati beállítások menüjének alsó szintje



Az `rpcbind(8)` és `rpc.statd(8)`, valamint az `rpc.lockd(8)` segédprogramok mind a távoli eljáráshívásokhoz (Remote Procedure Call, RPC) használhatóak. Az `rpcbind` segédprogram az NFS szerverei és kliensei között felügyeli a kapcsolatot, ezért a használata az NFS szerverek és kliensek működéséhez elengedhetetlen. Az állapot figyeléséhez az **rpc.statd** démon felveszi a kapcsolatot a többi gépen futó **rpc.statd** démonokkal. A jelentett állapotok általában a `/var/db/statd.status` állományban találhatók. Itt a következőként felsorolt elem az `rpc.lockd`, amelynek kiválasztásával állományzárolási szolgáltatásokat érhetünk el. Ezt többnyire az **rpc.statd** démonnal együtt alkalmazzák a zárolásokat kérő gépek és a kérések gyakoriságának nyilvántartására. Míg ezekkel a beállításokkal gyönyörűen nyomon lehet követni a működést, az NFS szerverek és kliensek megfelelő működéséhez nem kötelező a használatuk.

Ahogy haladunk tovább a listában, a következő elem a **Routed**, vagyis az útválasztásért felelős démon lesz. A `routed(8)` segédprogram a hálózati útválasztó táblázatokat tartja karban, felderíti az elérhető útválasztókat és kérésre bármelyik hozzá fizikailag csatlakozó gép számára átadja az általa nyilvántartott útválasztási adatokat. Ezt leginkább a helyi hálózat átjárójaként működő számítógépek használják. Kiválasztásakor egy ablak fog rákérdezni a segédprogram helyére. Az itt alaphőz felkínált érték általában megfelelő, ezért nyugtázhatjuk az **Enter** billentyű lenyomásával. Ezt követően egy másik menü jelenik meg, ahol a **routed** beállításait adhatjuk meg. Itt alapértelmezés szerint a `-q` kapcsoló szerepel.

A következő sor az **Rwhod** beállítása, aminek kiválasztásával el tudjuk indíttatni az `rwhod(8)` démont a rendszer elindítása során. Az `rwhod` segédprogram a rendszerüzeneteket a hálózaton időközönként szétküldi vagy "figyelő" (consumer) módban összegyűjti ezeket. Ennek pontosabb részleteit az `ruptime(1)` és `rwho(1)` man oldalakon találhatjuk meg.

Az `sshd(8)` démoné az utolsó előtti beállítás. Ez az **OpenSSH** biztonságos shell szervere, melyet a szabványos **telnet** és FTP szerverek helyett ajánlanak. Az **sshd** szerver tehát két gép közti biztonságos, titkosított kapcsolatok létrehozására használható.

A lista végén a **TCP Extensions** (TCP kiterjesztések) menüpontot találhatjuk. Segítségével a TCP RFC 1323 és RFC 1644 dokumentumokban leírt kiterjesztéseinek használatát engedélyezhetjük. Ezzel egyes gépek esetén felgyorsulhat a kapcsolat, azonban más esetekben pedig eldobódhat. Ez szerverek használatánál nem ajánlott, viszont független gépeknél kifizetődő lehet.

Most, miután beállítottuk a hálózati szolgáltatásokat, lépünk vissza a lista elején található **X Exit** (Kilépés) menüpontra és folytassuk a beállítást a következő opcióval, vagy egyszerűen az **X Exit** kétszeri kiválasztásával, majd

a [X Exit Install] (Kilépés a telepítőből) gomb lenyomásával lépünk ki a **sysinstall** programból.

2.10.16. A FreeBSD indulása

2.10.16.1. A FreeBSD/i386 indulása

Ha minden remekült ment, a képernyőn lentől felfelé gördülő üzeneteket fogunk látni, majd a rendszer várni fog tőlünk egy bejelentkezési nevet. A kiírt üzeneteket között a **Scroll Lock** lenyomása után a **PgUp** és **PgDn** billentyűk használatával tudunk lapozni. A **Scroll Lock** ismételt lenyomásával visszatérünk a bejelentkezéshez.

Nem minden esetben lesz látható az összes üzenet (a puffer végessége miatt), de miután bejelentkeztünk, ezeket a `dmesg` parancs kiadásával is megnézhetjük.

Bejelentkezni a telepítéskor megadott felhasználói név/jelszó párossal tudunk (a példában ez most `rpratt`). Lehetőleg ne jelentkezzünk be `root` felhasználóként!

A rendszer indításakor jellemzően előforduló üzenetek (a verzióra vonatkozó adatokat kihagytuk):

```
Copyright (c) 1992-2002 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
    The Regents of the University of California. All rights reserved.
```

```
Timecounter "i8254" frequency 1193182 Hz
CPU: AMD-K6(tm) 3D processor (300.68-MHz 586-class CPU)
  Origin = "AuthenticAMD" Id = 0x580 Stepping = 0
  Features=0x8001bf<FPU,VME,DE,PSE,TSC,MSR,MCE,CX8,MMX>
  AMD Features=0x80000800<SYSCALL,3DNow!>
real memory = 268435456 (262144K bytes)
config> di sn0
config> di lnc0
config> di le0
config> di ie0
config> di fe0
config> di cs0
config> di bt0
config> di aic0
config> di aha0
config> di adv0
config> q
avail memory = 256311296 (250304K bytes)
Preloaded elf kernel "kernel" at 0xc0491000.
Preloaded userconfig_script "/boot/kernel.conf" at 0xc049109c.
md0: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1: <VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
```

```
isa0: <ISA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
uhci0: <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci0
usb0: <VIA 83C572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr 1
uhub0: 2 ports with 2 removable, self powered
chip1: <VIA 82C586B ACPI interface> at device 7.3 on pci0
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xe800-0xe81f irq 9 at
device 10.0 on pci0
ed0: address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq 2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbd0: <keyboard controller (i8042)> at port 0x60-0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq 1 on atkbd0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbd0
psm0: model Generic PS/2 mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x1 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
ppbus0: IEEE1284 device found /NIBBLE
Probing for PnP devices on ppbus0:
plip0: <PLIP network interface> on ppbus0
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
ppi0: <Parallel I/O> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master using UDMA33
ad2: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata1-master using UDMA33
acd0: CDROM <DELTA OTC-H101/ST3 F/W by OIPD> at ata0-slave using PIO4
Mounting root from ufs:/dev/ad0sla
swapon: adding /dev/ad0slb as swap device
Automatic boot in progress...
/dev/ad0sla: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0sla: clean, 48752 free (552 frags, 6025 blocks, 0.9% fragmentation)
/dev/ad0slf: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0slf: clean, 128997 free (21 frags, 16122 blocks, 0.0% fragmentation)
/dev/ad0slg: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0slg: clean, 3036299 free (43175 frags, 374073 blocks, 1.3% fragmentation)
/dev/ad0sle: filesystem CLEAN; SKIPPING CHECKS
/dev/ad0sle: clean, 128193 free (17 frags, 16022 blocks, 0.0% fragmentation)
```

```

Doing initial network setup: hostname.
ed0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 192.168.0.1 netmask 0xffffffff broadcast 192.168.0.255
    inet6 fe80::5054::5ff::fede:731b%ed0 prefixlen 64 tentative scopeid 0x1
    ether 52:54:05:de:73:1b
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x8
    inet6 ::1 prefixlen 128
    inet 127.0.0.1 netmask 0xff000000
Additional routing options: IP gateway=YES TCP keepalive=YES
routing daemons:.
additional daemons: syslogd.
Doing additional network setup:.
Starting final network daemons: creating ssh RSA host key
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
cd:76:89:16:69:0e:d0:6e:f8:66:d0:07:26:3c:7e:2d root@k6-2.example.com
creating ssh DSA host key
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
f9:a1:a9:47:c4:ad:f9:8d:52:b8:b8:ff:8c:ad:2d:e6 root@k6-2.example.com.
setting ELF ldconfig path: /usr/lib /usr/lib/compat /usr/X11R6/lib
/usr/local/lib
a.out ldconfig path: /usr/lib/aout /usr/lib/compat/aout /usr/X11R6/lib/aout
starting standard daemons: inetd cron sshd usbd sendmail.
Initial rc.i386 initialization:.
rc.i386 configuring syscons: blank_time screensaver moused.
Additional ABI support: linux.
Local package initialization:.
Additional TCP options:.

FreeBSD/i386 (k6-2.example.com) (ttyv0)

login: rpratt
Password:

```

Az RSA és DSA kulcsok generálása a lassabb gépeken sokáig is eltarthat, habár ez mindig csak a friss telepítések utáni első indításkor történik meg. A rendszer későbbi indulásai ettől már gyorsabbak lesznek.

Ha X szervert is beállítottunk és választottunk hozzá egy alapértelmezett munkakörnyezetet, akkor ezt a parancssorból a `startx` kiadásával elindíthatjuk el.

2.10.17. A FreeBSD leállítása

Fontos, hogy mindig szabályosan állítsuk le az operációs rendszert, ne kapcsoljuk ki csak úgy egyszerűen a számítógépünket! A leállításhoz először a `su` parancs kiadásával, majd itt a `root` jelszavának megadásával vegyünk fel az ehhez szükséges rendszeradminisztrátori jogosultságokat. Ez viszont csak abban az esetben fog működni, ha a felhasználónk tagja a `wheel` csoportnak. Minden más esetben egyszerűen jelentkezünk be `root` felhasználóként és használjuk a `shutdown -h now` parancsot.

```
The operating system has halted.
Please press any key to reboot.
```

A fenti üzenet jelzi, hogy a leállító parancs kiadása után már kikapcsolhatjuk a számítógépet, vagy ha ehelyett egy billentyűt nyomunk le, akkor a gép újraindul.

A **Ctrl+Alt+Del** billentyűkombináció használatával is újra tudjuk indítani a rendszert, azonban ez normál működés közben nem ajánlott.

2.11. Hibakeresés

A most következő szakaszban azokra a telepítés során felmerülő problémákra próbálunk meg megoldásokat adni, amelyeket eddig már sokan jeleztek nekünk. Ezek mellett szerepel néhány kérdés és válasz is a FreeBSD és az MS-DOS vagy Windows közös használatáról.

2.11.1. Mit tegyünk ha valami nem működik

A PC architektúra különféle korlátozásai miatt szinte lehetetlen 100%-ban megbízhatóvá tenni az eszközök felderítését, azonban ennek hibája kapcsán néhány dolgot még tenni tudunk.

Ellenőrizzük a Hardware Notes (<http://www.FreeBSD.org/releases/index.html>) (Hardverjegyzék) című dokumentumban, hogy az adott hardvert a FreeBSD valóban ismeri.

Amennyiben a hardvereszközünket a rendszer ismeri, azonban még mindig jelentkeznek fagyások vagy egyéb gondok, készítenünk kell egy saját rendszermagot. Ezzel olyan eszközök támogatását is beépíthetjük a rendszermagba, amelyek eredetileg nem szerepelnek a `GENERIC` rendszermagban. A telepítéshez készített rendszerindító lemezekon található rendszermag a legtöbb eszközt a gyári `IRQ`, `IO-cím` és `DMA` csatorna beállításai mentén próbálja felkutatni. Ha viszont a hardverünket átállítottuk, ennek megfelelően módosítanunk kell a rendszermag beállításait és újra kell fordítanunk, hogy a FreeBSD tudja, hol is keresse az eszközt.

Olyan is adódhat, hogy egy nem létező eszköz keresése egy utána keresendő másik, jelenlevő eszköz felkutatását akadályozza meg. Ilyenkor az ütköző meghajtókat le kell tiltani.

Megjegyzés: Egyes problémák elkerülhetőek vagy csillapíthatóak a különböző hardverösszetevők, különösen az alaplapi firmware frissítésével. Az alaplapi firmware-jére sokszor csak BIOS-ként hivatkoznak, és a legtöbb alaplapi- vagy számítógépgyártó honlapján találhatjuk meg ezeket, valamint a rájuk vonatkozó utasításokat.

A legtöbb gyártó azonban erősen tiltakozik az alaplapi BIOS-frissítések ellen, és csak indokolt esetekben, például kritikus javításoknál javasolják. A frissítés kimenetele *lehet* rossz is, aminek következménye a BIOS tartós károsodása.

2.11.2. Az MS-DOS® és Windows® állományrendszereinek használata

A FreeBSD jelenleg nem támogatja a **Double Space™** alkalmazással tömörített állományrendszereket, ezért a FreeBSD csak úgy tud az adataihoz hozzáférni, ha előtte kitömörítjük ezeket. Ezt a **Start** menü **Programs** (Programok) > **System Tools** (Rendszerezszközök) menüjében található **Compression Agent** (Lemeztömörítés) elindításával tehetjük meg.

A FreeBSD támogatja az MS-DOS alapú (gyakran csak FAT típusúnak nevezett) állományrendszereket. A `mount_msdosfs(8)` parancs segítségével az ilyen rendszerek könnyedén becsatlakoztathatók a már létező könyvtárszerkezetbe, amivel így el tudjuk érni a tartalmát. A `mount_msdosfs(8)` programot általában nem közvetlenül hívjuk meg, hanem az `/etc/fstab` vagy a `mount(8)` segédprogram megfelelő paraméterezésével.

Az `/etc/fstab` állományban általában így néz ki egy ilyen sor:

```
/dev/ad0sN /dos msdosfs rw 0 0
```

Megjegyzés: A művelet végrehajtásához a `/dos` könyvtárnak már léteznie kell. Az `/etc/fstab` pontos formátumával kapcsolatban a `fstab(5)` man oldalt olvassuk el.

Az MS-DOS állományrendszerek esetében a `mount(8)` parancsot többnyire így adjuk ki:

```
# mount -t msdosfs /dev/ad0s1 /mnt
```

Ebben a példában a MS-DOS állományrendszer az elsődleges merevlemez első partícióján helyezkedik el. A mi helyzetünk ettől eltérő lehet, ezért ehhez vizsgáljuk meg a `dmesg` és `mount` parancsok kimeneteit. Segítségükkel elegendő információt tudunk összeszedni a gépünkön található partíciók kiosztásáról.

Megjegyzés: Előfordulhat, hogy a FreeBSD a többi operációs rendszertől eltérő módon számozza a slice-okat (vagyis az MS-DOS partíciókat). Konkrétan: a kiterjesztett MS-DOS partíciók általában nagyobb sorszámot kapnak, mint az elsődleges MS-DOS partíciók. Az `fdisk(8)` segédprogram segíthet megállapítani, hogy mely slice-ok tartoznak a FreeBSD-hez és melyek más operációs rendszerekhez.

A `mount_ntfs(8)` parancs használatával az NTFS partíciók hasonló módon csatlakoztathatóak.

2.11.3. Kérdések és válaszok

1. A rendszerem teljesen leáll amikor az indítás során eszközöket próbál megtalálni, vagy furcsán viselkedik a telepítés során, esetleg a floppy meghajtót nem is keresi.

A FreeBSD az i386, amd64 és ia64 platformokon az indítás közben az eszközök felderítésében erősen építkezik a rendszeren elérhető ACPI szolgáltatásra. Sajnos még mindig vannak hibák az ACPI meghajtóban, az alaplapokban és a BIOS-okban. A rendszerbetöltő harmadik fokozatában viszont az `hint.acpi.0.disabled` megadásával kikapcsolható az ACPI használata:

```
set hint.acpi.0.disabled="1"
```

Ez a beállítás a rendszer minden egyes indításakor törlődik, ezért a `hint.acpi.0.disabled="1"` bejegyzést fel kell vennünk a `/boot/loader.conf` állományba. A rendszerbetöltő működéséről részletesebben a 12.1 Szakaszban olvashatunk.

2. A FreeBSD telepítése után először indítom el a merevlemezről a rendszert, a rendszermag betöltődik és nekilát felkutatni a hardvereszközöket, azonban megáll a következő üzenettel:

```
changing root device to ad1s1a panic: cannot mount root
```

Mi lehet a gond? Mit tegyek?

Mit jelent a `bios_drive:interface(unit,partition)kernel_name` a rendszerindítás során megjelenő sűgőban?

Ez egy régóta fennálló probléma olyan rendszerek esetén, ahol a rendszerindításhoz használt lemez nem az első. A BIOS a FreeBSD-től eltérő sorszámozást használ, és az általa alkalmazott megfeleltetések megfejtése nehézkes.

Amikor a rendszer indítására használt lemez nem az első lemez a rendszerünkben, segítenünk kell a FreeBSD-nek a megtalálásában. Két gyakori helyzet alakulhat ki, és mind a kettőben el kell árulnunk a FreeBSD-nek, hogy hol található a rendszer indításához használható gyökér állományrendszer. Ezt a lemez BIOS-ban nyilvántartott sorszámanak, típusának és a neki megfelelő FreeBSD szerinti lemezszám megadásával tehetjük meg.

Az első szituációban két IDE-lemezünk van, mind a kettőt masterként állítottuk be a hozzájuk tartozó IDE-buszokon, és a közülük a másodikról akarjuk indítani a FreeBSD-t. A BIOS ezeket 0. és 1. lemezként látja, miközben a FreeBSD pedig `ad0` és `ad2` eszközként.

A FreeBSD 1. BIOS-számozású lemezen van, amelynek a típusa `ad` és a FreeBSD szerinti a 2 sorszámot viseli. Ezért ezt kell használnunk:

```
1:ad(2,a)kernel
```

Ha az elsődleges buszon van egy slave meghajtónk, akkor mindez nem szükséges (és valószínűleg rossz is).

A második szituációban egy SCSI-lemezeztől akarjuk indítani a rendszert, miközben egy vagy több IDE-lemez is található a gépünkben. Ebben az esetben a FreeBSD szerinti sorszám kisebb lesz, mint a BIOS szerinti. Ha tehát a két IDE-lemezünk mellett van még egy SCSI-lemez is, akkor annak a BIOS szerinti sorszáma 2, a típusa `da` és a FreeBSD szerinti sorszáma pedig 0. Ennek megfelelően a

```
2:da(0,a)kernel
```

sorral tudjuk elárulni a FreeBSD-nek, hogy a BIOS szerint 2. lemezeztől akarjuk indítani, amely a rendszerben található első SCSI-lemeznek felel meg. Ha csak egy IDE-lemezünk van, akkor a sort kezdjük az `1:` beírásával.

Miután megtaláltuk a megfelelő értékeket, a hozzá tartozó sort egy szövegszerkesztő segítségével tegyük közvetlenül a `/boot.config` állományba. A FreeBSD ezen állomány tartalmát fogja alaphól felhasználni a `boot:` bekérésénél, hacsak másképpen nem utasítjuk.

3. A telepítés után először próbálom meg elindítani a merevlemezről a FreeBSD-t, azonban a rendszerválasztó mindig csak F? opciókat kínál fel, és a rendszer indítása sem halad tovább.

A FreeBSD telepítése során rosszul adtuk meg a partíciószerkesztőben a merevlemezhez tartozó geometriát. Menjünk vissza a partíciószerkesztőhöz és adjuk meg újra a merevlemezünk helyes geometriáját. Ennek használatához pedig a FreeBSD-t is újra kell telepítenünk.

Ha egyáltalán képtelenek vagyunk megállapítani a merevlemezhez tartozó geometriát, akkor próbáljuk meg ezt: a lemez elején hozzunk létre egy kis méretű DOS partíciót és rakjuk utána a FreeBSD-t. Amikor a telepítőprogram észreveszi a DOS partíciót, megpróbálja magától kikövetkeztetni belőle a helyes geometriát, ami általában működik is.

Ez a tanács ugyan már nem érvényes, de álljon itt felvilágosításként:

Ha teljesen egy FreeBSD alapú szerver vagy munkaállomás kialakítására szánjuk a számítógépünket, és nem törődünk a DOS-szal, Linuxszal és a többi operációs rendszerrel történő (jövőbeli) kompatibilitással, használhatjuk akár az egész lemezt is (a partíciószerkesztőben ez az A opció). Ezzel egy olyan nem szabványos beállítást engedélyezünk, amivel a FreeBSD elfoglalja a lemezt annak legelső szektorától a legutolsó szektoráig. Ilyenkor ugyan el tudunk tekinteni a geometriával kapcsolatos beállításoktól, azonban így a FreeBSD-n kívül semmilyen más operációs rendszert nem tudunk majd futtatni a gépen.

4. A rendszer megtalálja a ed(4) hálózati kártyát, azonban folyamatosan hibát ad időtúllépésre hivatkozva.

Az említett kártya valószínűleg a `/boot/device.hints` állományban beállítottaktól eltérő IRQ-t használ. A ed(4) meghajtó alapértelmezés szerint nem használ "szoftveres" beállításokat (amiket DOS-ban az EZSETUP használatával adunk meg), viszont engedélyezhetjük, ha a kártyánál megadjuk az -1 beállítást.

Hardveresen ezt a kártyán levő jumperek segítségével állíthatjuk be (ehhez változtassuk meg a rendszermag beállításait is, amennyiben szükséges), vagy a -1 kapcsolón keresztül a `hint.ed.0.irq="-1"` megadásával utasíthatjuk a rendszermagot az IRQ szoftveres beállítására.

Másik lehetőség, amikor a kártyánk a 9-es IRQ-t használja, amelyet általában megosztanak a 2-es IRQ-val, ami gyakori problémák forrása (különösen abban az esetben, amikor a VGA kártya a 2-es IRQ-t használja!) lehet. Lehetőleg ne használjuk a 2-es és 9-es IRQ-kat.

5. Amikor a **sysinstall** programot egy X11 terminálban futtatom, a sárga színű betűket viszonylag nehéz olvasni a világosszürke háttérrel. Esetleg lehet valahogy növelni a kontrasztot az alkalmazás használatakor?

Ha az X11 telepítése után a **sysinstall** által választott színekkel nem olvasható a szöveg xterm(1) vagy rxvt(1) terminálokban, akkor vegyük fel a következő sort a felhasználói könyvtárunkban levő `.xdefaults` konfigurációs állományunkba: `XTerm*color7:#c0c0c0`. Ezzel majd egy sötétebb szürke hátteret kapunk.

2.12. Telepítési útmutató haladóknak

Írta: Frissítette: Valentino Vaschetto és Marc Fonvieille.

Ebben a szakaszban megtudhatjuk, hogyan telepítsük a FreeBSD-t speciális esetekben.

2.12.1. A FreeBSD telepítése billentyűzet vagy monitor nélkül

A telepítés ezen fajtáját “fej nélküli telepítésnek” (headless install) hívják, mivel a gép, amire a FreeBSD-t telepíteni akarjuk, nem rendelkezik monitorral vagy éppen még VGA kimenettel sem. Felmerülhet a kérdés: hogyan lehetséges mindez? A soros vonali konzol használatával! A soros konzol segítségével lényegében egy másik számítógép monitorját és billentyűzetét használjuk. Ennek megvalósításához elsőként kövessük a rendszerindító pendrive készítésének 2.3.7 Szakaszban leírt lépéseit, vagy töltsük le a megfelelő ISO image-et a telepítéshez, lásd 2.13.1 Szakasz.

A következő lépésekkel tehetjük képessé a soros konzolon keresztüli rendszerindítást: (CD-lemez használata esetén az első lépésre nincs szükség)

1. A rendszerindító pendrive átállítása soros konzolra

Ha a korábban előkészített pendrive-val most csak egyszerűen elindítanánk a FreeBSD-t, akkor a megszokott telepítési módban indulna el. Mi viszont azt akarjuk, hogy a telepítéshez a FreeBSD a soros konzolon keresztül induljon el. Ehhez csatlakoztassuk az eszközt a számítógéphez, valamint a mount(8) paranccsal FreeBSD rendszerünkhöz pedig a hozzátartozó állományrendszert.

```
# mount /dev/da0a /mnt
```

Megjegyzés: A konkrét eszköznevet és csatlakozási pontot módosítsuk a saját környezetünknek megfelelően.

Most, miután már fizikailag és logikailag is csatlakoztattuk a pendrive-ot, be kell állítanunk a soros konzol használatára rendszerindítás közben. Ehhez egy loader.conf nevű állományt kell elhelyeznünk a pendrive állományrendszerén a soros konzolra (mint rendszerkonzolra) vonatkozó beállítással:

```
# echo 'console="comconsole"' >> /mnt//boot/loader.conf
```

Miután a pendrive-on sikeresen elvégeztük a szükséges beállítást, válasszuk le a umount(8) parancs kiadásával:

```
# umount /mnt
```

Most már leválaszthatjuk a pendrive-ot, és ugorjunk közvetlenül a harmadik lépésre.

2. A null-modem kábel csatlakoztatása

Össze kell kötnünk a két számítógépet egy null-modem kábelrel. Nincs más teendőnk, mit összekapcsolni a két gép soros portjait. *Itt a szokásos soros kábel nem működik*, konkrétan null-modem kábelre van szükség, mivel benne néhány vezetékét máshogy kötötték be.

3. A telepítő CD beállítása soros konzolra

Ha a telepítésre szánt ISO image-ből készített lemezzel (lásd 2.13.1 Szakasz) a FreeBSD normál módban indul el. A soros konzol használatához viszont kibontani, módosítani és újragenerálni kell az adott image-et mielőtt lemezre íránk.

A korábban, például a FreeBSD-8.1-RELEASE-i386-disc1.iso néven letöltött image-ből a tar(1) segédprogrammal tudjuk kinyerni a benne tárolt összes állományt:

```
# mkdir /a/hasznalt/iso/helye
# tar -C /a/hasznalt/iso/helye -pxvf FreeBSD-8.1-RELEASE-i386-disc1.iso
```

Ezt követően módosítanunk kell a telepítőlemez a soros konzol használatára. Ehhez egy `loader.conf` állományt kell hozzáadnunk a kibontott ISO image tartalmához. Ebben állítjuk be a soros konzolt rendszerkonzolnak:

```
# echo 'console="comconsole"' >> /a/hasznalt/iso/helye/boot/loader.conf
```

Ezután készítsünk egy új ISO image-et a módosított tartalom alapján. Ehhez a `sysutils/cdrtools` port részeként elérhető `mkisofs(8)` segédprogramot használjuk:

```
# mkisofs -v -b boot/cdboot -no-emul-boot -r -J -V "soroskonzolos" -o soroskonzolos-FreeBSD-8.1-RELEASE-i386
```

Most már van egy megfelelően összeállított ISO image-ünk, amelyet CD-lemezre tudunk írni a kedvenc CD-író alkalmazásunkkal.

4. A telepítés indítása

Most már ideje elkezdenni a telepítést. Tegyük a `boot.flp` image-et tartalmazó lemezt a fej nélkül telepítendő gép meghajtójába és kapcsoljuk be.

5. Kapcsolódás a fej nélküli gépre

Ezután a `cu(1)` parancs felhasználásával kapcsolódjunk rá a gépre:

```
# cu -l /dev/cuau0
```

Ezt FreeBSD 7.x esetén így kell használnunk:

```
# cu -l /dev/cuad0
```

Ezzel készen is vagyunk! Innentől a `cu` által megnyitott kapcsolaton keresztül tudjuk vezérelni a fej nélküli számítógépet. Hamarosan betölti a rendszermagot, majd megkérdezi a használt terminál típusát. Itt válasszuk ki a színes FreeBSD konzolt (FreeBSD color console) és folytassuk a telepítést a megszokott módon.

2.13. Saját telepítőeszköz elkészítése

Megjegyzés: Az ismételések elkerülése végett a továbbiakban a "FreeBSD lemez" a megvásárolható vagy a magunk által készített FreeBSD CD-re vagy DVD-re vonatkozik.

Adódhatnak olyan esetek, amikor létre kell hoznunk a FreeBSD telepítésére használt saját eszközünket és/vagy forrásunkat. Ez lehet egy tetszőleges fizikai eszköz, például szalag, vagy bármilyen olyan forrás, ahonnan a **sysinstall** képes állományokat elérni, például egy FTP oldal vagy egy MS-DOS partíció.

Például:

- Egy FreeBSD lemezünk van és több hálózaton kapcsolódó számítógépünk. Készíteni akarunk egy helyi FTP oldalt a FreeBSD lemez felhasználásával, és így a hálózaton levő gépre az internet helyett innen telepítjük a rendszert.
- Van egy FreeBSD lemezünk, azonban a FreeBSD-nek nem sikerült felismernie a CD/DVD-meghajtónkat, viszont az MS-DOS/Windows-nak igen. Felmásoljuk a FreeBSD telepítéséhez használt állományokat ugyanazon a számítógépen található egyik DOS partícióra, majd a FreeBSD-t ezekkel telepítjük.
- A gépben, amelyre telepíteni akarunk, nincs CD/DVD-meghajtó vagy hálózati kártya, viszont "Laplink stílusú" soros vagy párhuzamos kábellel hozzá tudunk kapcsolódni egy olyan számítógépről, amelyben viszont van.

- Készíteni akarunk a FreeBSD telepítésére használható szalagot.

2.13.1. Telepítő CD készítése

A FreeBSD Projekt minden kiadás részeként architektúráként elérhetővé tesz legalább két CD image-et (“ISO image-et”). Ha rendelkezünk CD-íróval, ezeket az image-eket fel-, illetve ki tudjuk írni (“égetni”) CD-re, és a FreeBSD telepítésére tudjuk használni. Tehát ha van a kezünk ügyében CD-író és olcsón jutunk nagyobb sebességű interneteléréshez, akkor a FreeBSD telepítésének ez a legkönnyebb módja.

1. A megfelelő ISO image-ek letöltése

Az egyes kiadások ISO image-ei letölthetőek a

`ftp://ftp.FreeBSD.org/pub/FreeBSD/ISO-IMAGES-architektúra/változat` címről vagy annak legközelebbi tükrözéséről. Az *architektúra* és *változat* részeket igényeinknek megfelelően helyettesítsük.

Az említett könyvtár általában a következő lemezek image-eit tartalmazza:

Táblázat 2-4. FreeBSD 7.x és 8.x ISO image-ek nevei és jelentései

Állománynév	Tartalom
<code>FreeBSD-változat-RELEASE-architektúra-bootonly1</code>	Ez az ISO image-dzsel tudjuk a FreeBSD CD-meghajtóról indításával elkezdni a telepítést. Fontos tudnunk azonban, hogy ez az image nem tartalmazza a FreeBSD telepítéséhez szükséges komponenseket. Ezt a rendszer indítása után hálózaton keresztül (például egy FTP szerver segítségével) tudjuk megtenni.
<code>FreeBSD-változat-RELEASE-architektúra-dvd1</code>	Ez a DVD image minden, az alap FreeBSD rendszer telepítéséhez szükséges komponenst tartalmaz, bináris csomagokkal és dokumentációval együtt. Ezenkívül még “élő” rendszert is tudunk indítani vele, közvetlenül a lemezről.
<code>FreeBSD-változat-RELEASE-architektúra-memstick1</code>	Ez az image egy USB pendrive-ra írható, és minden olyan számítógépen használható, amely képes ilyen eszköztől elindulni. Támogatja az “élő” módot is, amellyel rendszerünket állíthatjuk helyre. Ez az image nem érhető el FreeBSD 7.3 vagy korábbi rendszerek esetén.
<code>FreeBSD-változat-RELEASE-architektúra-disc1</code>	Ez az image tartalmazza az alap FreeBSD operációs rendszert és a hozzá tartozó dokumentációt, de semmilyen más további csomagot nem.
<code>FreeBSD-változat-RELEASE-architektúra-disc2</code>	Ezen az image-en bináris csomagok találhatók. Ilyen a FreeBSD 8.0 és az utána következő változatoknál már nincs.
<code>FreeBSD-változat-RELEASE-architektúra-disc3</code>	Ez egy másik image, amelyen szintén bináris csomagok találhatók. Ilyen a FreeBSD 8.0 és az utána következő változatoknál már nincs.

Állománynév**Tartalom**

FreeBSD-változat-RELEASE-architektúra-docs.1 A FreeBSD dokumentációja.

FreeBSD-változat-RELEASE-architektúra-livefs.1 Ez az image a rendszerhelyreállításához használt "élő" indítási módot támogatja, telepítést alapvetően nem lehet vele végezni.

Megjegyzés: A FreeBSD 7.3 és a FreeBSD 8.1 előtti 7.x, illetve 8.x kiadások egy ettől eltérő elnevezési sémát követnek: a hozzájuk tartozó ISO image-ek neveiben nem szerepel a FreeBSD- előtag.

Le kell töltenünk az első lemez vagy (ha elérhető) a bootonly lemez ISO image-einek egyikét. A kettőt egyszerre viszont ne töltsük le, mivel a disc1 image tartalmaz mindent, ami a bootonly image-en megtalálható.

Akkor használjuk a bootonly jelzésű image-et, ha szélessávú interneteléréssel rendelkezünk. Segítségével el tudjuk kezdeni a FreeBSD telepítést, és szükség szerint a port/csomagrendszer (lásd 4 fejezet) használatával csomagokat tudunk letölteni és telepíteni.

A DVD image-ét (dvd1) akkor érdemes használni, ha a FreeBSD adott kiadásának telepítése mellett igényt tartunk valamennyi csomagra is.

A további lemezek image-ei is hasznosak lehetnek, de nem feltétlenül kellenek a telepítéshez, főleg abban az esetben, amikor gyors interneteléréssel rendelkezünk.

2. A CD-k írása

Ezután lemezekre kell írunk a letöltött image-eket. Amennyiben ezt egy másik FreeBSD rendszeren végezzük, ennek részleteiről a 18.6 Szakasz számol be (különösen a 18.6.3 Szakasz és a 18.6.4 Szakasz leírása).

Ha másik platformon végezzük ezt a műveletet, akkor az adott platformon felkínált CD-író szoftverekkel kell dolgoznunk. Az image-ek szabványos ISO formátumúak, amelyet szinte az összes CD-író alkalmazás ismer.

Megjegyzés: Ha kíváncsiak vagyunk egy saját FreeBSD kiadás elkészítésére, olvassuk el a kiadások szervezéséről szóló cikket (angolul) (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/releeng).

2.13.2. Helyi FTP oldal létrehozása FreeBSD lemezzel

A FreeBSD lemezeken az FTP oldalakéhoz hasonló elrendezést találunk. Ez megkönnyíti a hálózatunkban található számítógépekhez a FreeBSD telepítésére használható helyi FTP oldal létrehozását.

1. Az FTP oldalnak otthont adó FreeBSD számítógépen tegyük a CD-t a meghajtóba, majd csatlakoztassuk a /cdrom könyvtárba.

```
# mount /cdrom
```

2. Hozzunk létre egy anonim FTP hozzáférést az /etc/passwd állományban. A vipw(8) segítségével tehát illesszük be a következő sort az /etc/passwd állományba:

```
ftp:*:99:99::0:0:FTP:/cdrom:/nonexistent
```

3. Gondoskodjunk róla, hogy az FTP szolgáltatás engedélyezve legyen az `/etc/inetd.conf` állományban.

Most már bárki, aki képes csatlakozni ehhez a számítógéphez, a telepítés típusának ki tudja választani az FTP-t. Az FTP oldalak menüjében válassza az “Other” (Egyéb) pontot, majd adja meg az `ftp://gépnev` címet.

Megjegyzés: Ha az FTP-n csatlakozó kliensek rendszerindításhoz használt eszköze (általában a floppy) verziója nem egyezik meg tökéletesen a helyi FTP oldalon találhatóval, akkor a **sysinstall** nem engedi a telepítést. Ha a változatok nem hasonlóak és ezt felül akarjuk bírálni, akkor be kell lépünk az Options (Beállítások) menübe, ahol át kell állítanunk a terjesztés nevét (distribution name) any (bármelyik)-re.

FigyelemA fenti megközelítés kizárólag csak egy tűzfallal védett helyi hálózaton javasolt. FTP szolgáltatás létrehozása az interneten (és nem a helyi hálózatunkban) levő számítógépek számára különböző támadásoknak és egyéb kellemetlenségeknek teszi ki a számítógépünket. Határozottan javasoljuk, hogy ebben az esetben különösen ügyeljünk a biztonságra.

2.13.3. Telepítőfloppyk létrehozása

Ha floppylemezről kellene telepítenünk (amit viszont *semmiképpen sem* ajánlanánk) egy nem támogatott hardvereszköz miatt, vagy mert egyszerűen szeretjük a dolgok nehezebbik oldalát megfogni, akkor ehhez először elő kell készítenünk pár lemezt.

Legalább annyi 1,44 MB-os lemezre van szükségünk, mint amennyire ráférnek a `base` (alapterjesztés) könyvtárban található állományok. Ha DOS-ban hozzuk létre ezeket a lemezeket, akkor a használatukhoz meg *kell* formázni ezeket az MS-DOS `FORMAT` parancsával. Windows használata esetén az Windows Explorerben (Intézőben) tudjuk megformázni a lemezeket (kattintsunk a jobb gombbal az `A:` meghajtóra, majd válasszuk a “Format” (Formázás) menüpontot).

Ne bízzunk a gyárilag formázott (“pre-formatted” jelzésű) lemezekben! Menjünk biztosra és formázzuk meg mi magunk is lemezeket. A felhasználóinktól régebben számtalan olyan panasz érkezett, amely a helytelenül megformázott lemezből fakadt, ezért erre most kiemelten felhívjuk a figyelmet.

A formázás abban az esetben sem bizonyul rossz ötletnek, ha egy másik FreeBSD gépen gyártjuk le a lemezeket, habár nem kell mindegyik lemezre DOS állományrendszert tennünk. Helyette a `bsdlabel` és `newfs` parancsok használatával UFS állományrendszert is tehetünk rájuk, ahogy (1,44 MB méretű lemezek esetén) ezt az alábbi parancsok mutatják:

```
# fdformat -f 1440 fd0.1440
# bsdlabel -w fd0.1440 floppy3
# newfs -t 2 -u 18 -l 1 -i 65536 /dev/fd0
```

Ezután a többi állományrendszerhez hasonlóan a lemezeket tudjuk csatlakoztatni és írni.

Miután megformáztuk a lemezeket, rájuk kell másolnunk az állományokat. A terjesztésekhez tartozó állományokat adott méretű darabokra szeleteltük, így kényelmesen ráférnek egy hagyományos 1,44 MB méretű floppyra. Menjünk végig az összes floppyn és mindegyikre pakoljuk fel a lehető legtöbb állományt egészen addig, amíg így az összes szükséges terjesztést össze nem szedtük. A floppykon minden terjesztés kerüljön egy hozzá tartozó alkönyvtárba, például: `a:\base\base.aa`, `a:\base\base.ab` és így tovább.

Fontos: Az első lemezre rá kell másolnunk a `base.inf` nevű állományt is, mivel ennek beolvasásával lesz képes kitalálni a telepítő, hogy a terjesztések összeszedése és összefűzése során mennyi darabot keressen.

Ahogy elérkezünk a telepítőeszköz kiválasztásához a telepítés folyamatában, ott válasszuk a Floppy menüpontot, majd utána kövessük a felbukkanó üzeneteket.

2.13.4. Telepítés MS-DOS partícióról

Amikor egy MS-DOS partícióról akarunk telepíteni, előkészítés gyanánt másoljuk a terjesztésekhez tartozó állományokat a partícióra egy `freebsd` könyvtárba. Ez lesz például a `c:\freebsd`. Ebben a könyvtárban igyekezzük minél jobban megtartani a CD vagy az FTP oldal könyvtárszerkezetét, ezért erre a CD-ről történő átmásolásra a DOS `xcopy` parancsát javasoljuk. Például így tudjuk előkészíteni a FreeBSD legegyszerűbb változatának telepítését:

```
C:\> md c:\freebsd
C:\> xcopy e:\bin c:\freebsd\bin\ /s
C:\> xcopy e:\manpages c:\freebsd\manpages\ /s
```

A fentiekben feltételeztük, hogy ehhez a `C:` meghajtón elég szabad helyünk van, valamint az `E:` meghajtón érjük el a CD-t.

Ha nincs CD-meghajtónk, az [ftp.FreeBSD.org](ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/8.1-RELEASE/) (<ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/8.1-RELEASE/>) címről letölthetjük a terjesztést. Minden egyes terjesztés külön könyvtárban található, tehát például a *base* (alap) terjesztés az `8.1/base/` (<ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/8.1-RELEASE/base/>) könyvtárban található.

Mindegyik telepítendő terjesztést (ami még elfér) másoljuk át az MS-DOS partíció `c:\freebsd` könyvtárba — a telepítéshez egyébként egyedül a `BIN` terjesztés szükséges.

2.13.5. Telepítőszalag létrehozása

Valószínűleg a szalagos módszer a legegyszerűbb, egyfajta élő FTP-s vagy CD-s telepítés. A telepítőprogram arra számít, hogy a szalagon az állományok egymás után helyezkednek el. Tehát miután beszereztük a nekünk kellő terjesztésekhez tartozó összes állományt, egyszerűen vegyük fel ezeket a szalagra:

```
# cd /freebsd/distdir
# tar cvf /dev/rwt0 dist1 ... dist2
```

Mielőtt telepítenénk, ellenőrizzük, hogy legyen elég helyünk valamelyik (a telepítés során majd kiválasztható átmeneti) könyvtárban ahhoz, hogy az itt létrehozott szalag *teljes* tartalma elférjen benne. Mivel a szalagok csak szekvenciálisan érhetők el, ezért ennél a módszernél jó sok ideiglenes tárhelyre lesz szükségünk.

Megjegyzés: A telepítés megkezdése után a szalagnak már *azelőtt* a meghajtóban kell lennie, hogy rendszerindító floppyról elindítanánk a rendszert, máskülönben nem találja meg.

2.13.6. Mielőtt hálózatról telepítenénk

Háromféle hálózati telepítési mód létezik: Ethernet (szabványos Ethernet-vezérlővel), soros port (PPP) vagy párhuzamos port (PLIP (laplink kábel)).

Valószínűleg az Ethernet-csatlakozó választásával érjük el a leggyorsabb hálózati telepítést. A FreeBSD ismeri a legtöbb PC-s Ethernet kártyát. Az ismert kártyák (és a hozzájuk tartozó beállítások) a FreeBSD egyes kiadásának hardverjegyzékében (Hardware Notes) találhatóak meg. Amennyiben egy támogatott PCMCIA Ethernet kártyát használunk, mindig a laptop bekapcsolása *előtt* helyezzük be! A FreeBSD telepítés közben sajnos nem támogatja a PCMCIA kártyák menetközbeni behelyezését.

Ezenkívül még ismernünk kell a hálózaton kapott IP-címünket, az általa használt címosztály hálózati maszkját, a gépünk nevét. Ha PPP kapcsolaton keresztül telepítünk és nincs statikus IP-címünk, akkor minden bizonnyal az internet-szolgáltatónktól kaptunk egyet dinamikusan. A konkrét hálózati beállításokat a hálózatunk rendszergazdájától is érdemes megkérdezni. Ha a hálózaton levő többi gépre névvel és nem IP-címmel hivatkozunk, akkor szükségünk lesz még egy név(feloldó) szerverre és az internet eléréséhez egy átjáró címére is (ha PPP-t használunk, ez a szolgáltatónk IP-címe lesz). Ha FTP-ről HTTP proxy használatával telepítünk, akkor a proxy címe is kelleni fog. Ha magunktól nem vagyunk képesek ezekre a kérdésekre válaszolni, akkor az ilyen típusú telepítés megkezdése *előtt* tényleg segítséget kell kérnünk egy rendszergazdától vagy az internet-szolgáltatónktól.

Ha modemet használunk, akkor a PPP szinte biztosan megfelel nekünk. Gondoskodjunk róla, hogy már a telepítés korai szakaszában rendelkezésünkre áll az internet-szolgáltatónkkal kapcsolatosan minden hasznos információ.

Ha PAP vagy CHAP használatával kapcsolódunk a szolgáltatónkhoz (másképp szólva Windows-ban így tudunk szkriptek nélkül csatlakozni), mindössze a `dial` parancsot kell kiadnunk a **ppp** parancssorában. Minden más esetben tudnunk kell a modemünk saját "AT parancsaival" tárcsázni az internet-szolgáltatónkat, hiszen ehhez a PPP tárcsázó csak egy nagyon kezdetleges terminálemulációt nyújt. Ezzel kapcsolatban olvassuk el a kézikönyv és a GYIK (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/ppp.html) idevágó részeit. Ha gondjaink akadnának, a naplózás a `set log local ...` parancs kiadásával átirányítható közvetlenül a képernyőre.

Ha kötött módon tudunk csatlakozni egy másik (2.0-R vagy későbbi verziójú) FreeBSD géphez, akkor megpróbálkozhatunk a párhuzamos "laplink" kábellel. A párhuzamos porton keresztüli adatátvitel sebessége a soros vonalénál jóval nagyobb (egészen 50 kbyte/mp), ezért vele a telepítés is gyorsabb.

2.13.6.1. Mielőtt NFS-ről telepítenénk

A telepítés NFS-en keresztül szinte magától értetődik. Egyszerűen csak másoljuk a FreeBSD terjesztéseihez tartozó állományokat az NFS szerverre és állítsuk be rá az NFS telepítőeszközt.

Ha a szerver csak "privilegizált portokat" ismer (ami általában alapértelmezett a Sun munkaállomásoknál), a telepítés megkezdése előtt az **Options** (Beállítások) menüben be kell állítani az **NFS Secure** (Biztonságos NFS) opciót.

Ha egy gyenge minőségű és kis adatátviteli sebességű Ethernet kártyánk van, akkor emellett még hasznos lehet beállítani az **NFS Slow** (Lassú NFS) opciót is.

Az NFS-en keresztüli telepítés működéséhez a szervernek támogatnia kell az alkönyvtárak csatlakoztatását is, tehát például ha a FreeBSD 8.1 terjesztésünk a `ziggy:/usr/archive/stuff/FreeBSD` könyvtárban található, akkor `ziggy` nevű gépnek lehetővé kell tennie a `/usr/archive/stuff/FreeBSD` könyvtár közvetlen csatlakoztatását is, nem csak a `/usr` vagy `/usr/archive/stuff` könyvtárakét.

A FreeBSD `/etc/exports` állományában ezt az `-alldirs` beállítással vezérelhetjük. Más NFS szervereken esetleg más megszokásokat kell követnünk. Amennyiben a szervertől `permission denied` (hozzáférés megtagadva) üzeneteket kapjuk, valószínű, hogy ezt nem állítottuk be megfelelően.

Fejezet 3. A UNIX alapjai

Átdolgozta: Chris Shumway.

3.1. Áttekintés

Ez a fejezet a FreeBSD operációs rendszer alapvető funkcióit és parancsait mutatja be. Az itt tárgyalásra kerülő anyag nagy része érvényes bármelyik más UNIX-szerű operációs rendszer esetén is. Ezért, ha már ismerjük az említésre kerülő ismereteket, minden további gond nélkül átugorhatjuk ezt a fejezetet. Azonban ha még teljesen ismeretlen számunkra a FreeBSD, minden bizonnyal ez lesz az, amit alaposan át kell majd olvasnunk.

A fejezet elolvasása során megismerjük:

- az ún. “virtuális konzolok” használatát FreeBSD alatt;
- hogyan működnek együtt a UNIX állományokra vonatkozó engedélyei a FreeBSD saját kiegészítéseivel;
- egy FreeBSD állományrendszer alapértelmezett kialakítását;
- a FreeBSD lemezszervezését;
- hogyan csatlakoztassunk és válasszunk le állományrendszereket;
- mik azok a folyamatok, démonok és jelzések;
- mik azok a parancsértelmezők, és miként tudjuk megváltoztatni az alapértelmezett bejelentkezési környezetünket;
- hogyan használjuk az alapvető szövegszerkesztőket;
- mik az eszközök és az eszközeírók;
- FreeBSD alatt milyen bináris formátumokat használhatunk;
- szükség esetén hogyan olvassuk el a megfelelő man oldalakat.

3.2. Virtuális konzolok és terminálok

A FreeBSD számos módon használható. Ezek közül az egyik az, ha parancsokat gépelünk be a szöveges terminálon. Így érhető el egyszerűen a UNIX operációs rendszer rugalmasságának és erejének jelentős része. Ebben a szakaszban megtudhatjuk, mik azok a “terminálok” és “konzolok” és miként tudjuk ezeket FreeBSD alatt használni.

3.2.1. A konzol

Ha nem állítottuk volna be, hogy a FreeBSD indulása során automatikusan induljon el a grafikus felület is, akkor a rendszer egy bejelentkező képernyőt fog mutatni közvetlenül a rendszerindítás befejeződése után. Ekkor valami ilyesmit kell majd látnunk:

```
Additional ABI support:.  
Local package initialization:.  
Additional TCP options:.
```

```
Fri Sep 20 13:01:06 EEST 2002
```

```
FreeBSD/i386 (pc3.example.org) (ttyv0)
```

```
login:
```

Egyes rendszereken ugyan némileg eltérhetnek az üzenetek, de hasonlót kell látnunk. Minket most az utolsó két sor érdekel. Az utolsó előtti sorban ez olvasható:

```
FreeBSD/i386 (pc3.example.org) (ttyv0)
```

Ez a sor arról értesít minket, hogy a rendszerünk éppen most indult el: egy “FreeBSD” konzolt látunk, amely egy Intel x86 architektúrájú processzoron fut ¹. A gépünk neve (mivel minden UNIX-os gép rendelkezik egy névvel) `pc3.example.org`, és ennek a rendszerkonzolját látjuk most éppen — a `ttyv0` terminált.

Végezetül az utolsó sor mindig:

```
login:
```

Ez az a rész, ahova a FreeBSD-be történő bejelentkezéshez meg kell adnunk a “felhasználói nevünket” (user name). A következő szakaszban erről olvashatunk.

3.2.2. Bejelentkezés a FreeBSD-be

A FreeBSD egy többfelhasználós, többfeladatos rendszer. Így hívják hivatalosan azokat a rendszereket, amelyeket többen tudnak használni és egyetlen számítógépen egyszerre rengeteg programot képesek futtatni.

Minden többfelhasználós rendszernek valamilyen módon meg kell tudnia különböztetnie egy “felhasználóját” a többitől. A FreeBSD-ben (és minden más UNIX-szerű operációs rendszerben) ezt úgy érik el, hogy a programok futtatása előtt minden felhasználónak “be kell jelentkeznie” a rendszerbe. Minden felhasználó rendelkezik egy egyedi névvel (ez a “felhasználói név”) és ehhez egy titkos kulccsal (ez a “jelszó”). A FreeBSD a programok futtatásához ezt a kettőt fogja elkérni a felhasználótól.

Egyből miután a FreeBSD elindult és befejezte a rendszerindításhoz használt szkriptjeinek lefuttatását ², ez a kijelzés (vagy más néven “prompt”) fog megjelenni és kér egy érvényes felhasználói nevet:

```
login:
```

A példa kedvéért most tegyük fel, hogy a felhasználói nevünk `pgj`. Az iménti prompthoz írjuk be, hogy `pgj` és nyomjuk le az **Enter** billentyűt. Ezt követően meg kell jelennie egy másik promptnak is, amely egy “jelszót” (password) kér:

```
login: pgj
Password:
```

Most pedig gépeljük be `pgj` jelszavát és nyomjunk utána egy **Enter** billentyűt. Vigyázzunk, hogy a jelszót *nem látjuk* a beírás során! Emiatt most ne aggódjunk. Ezzel kapcsolatban elegendő csak annyit tudni, hogy mindez biztonsági megfontolásokból történik.

Amennyiben jól adtuk meg a jelszavunkat, sikeresen bejelentkeztünk a FreeBSD rendszerébe és készen állunk az összes elérhető parancs kipróbálására.

Bejelentkezés után a MOTD (message of the day) vagy más néven “a nap üzenete” jelenik meg, amelyet a parancssor követ (egy `#`, `$` vagy `%` jel). Innen tudhatjuk meg, hogy sikerült bejelentkeznünk.

3.2.3. Több konzol használata

A UNIX parancsokat egy konzolon is szépen ki tudjuk adni, de a FreeBSD egyszerre ugyebár több programot is tud futtatni. A parancsok megadásához viszont egyetlen konzol használata elég nagy pazarlás lenne, hiszen egy olyan operációs rendszer mint a FreeBSD, tucatnyi programot képes futtatni egy időben. Ebben az esetben jelenthetnek számunkra segítséget a “virtuális konzolok”.

A FreeBSD beállítható úgy, hogy sok-sok különféle virtuális konzolt ajánljon fel számunkra. A virtuális konzolok között a billentyűzeten a megfelelő gombok lenyomásával tudunk váltani. Mindegyik konzolnak megvan a saját kimeneti csatornája, és a virtuális konzolok közti váltás folyamán a FreeBSD gondoskodik a billentyűzetről érkező bemenet, valamint a monitorra irányított kimenet megfelelő kezeléséről.

A konzolok közti váltásra a FreeBSD külön billentyűkombinációkat tart fenn³. A FreeBSD-ben a különböző virtuális konzolok közti váltásra az **Alt-F1**, **Alt-F2** billentyűket, az **Alt-F8** billentyűkombinációval bezárólag használhatjuk.

A konzolok közti váltogatás során a FreeBSD ügyel a képernyő tartalmának elmentésére és visszaállítására. Ennek eredményeképpen “úgy látszik”, mintha több “virtuális” képernyőn és billentyűzeten adnánk parancsokat a FreeBSD-nek.

3.2.4. Az /etc/ttys állomány

A FreeBSD alapértelmezés szerint nyolc virtuális konzollal indul. Ez azonban nem egy előre rögzített érték, hiszen könnyedén testreszabhatjuk úgy a telepített rendszerünket, hogy több vagy esetleg kevesebb virtuális konzollal induljon el. A virtuális konzolok száma és azok pontos beállítása az /etc/ttys állományon keresztül adható meg.

A FreeBSD virtuális konzoljait tehát az /etc/ttys állomány megfelelő módosításával tudjuk behangolni. Itt minden egyes olyan sor, amely nem megjegyzés (vagyis azok a sorok, amelyek nem a # karakterrel kezdődnek), tartalmazza az egyes terminálok vagy virtuális konzolok beállításait. Az állomány a FreeBSD telepítésében szereplő, alapértelmezett változata kilenc virtuális konzol konfigurációját tartalmazza, amelyek közül nyolc aktív. Ezek a tttyv résszel kezdődő sorok:

#	name	getty	type	status	comments
#	tttyv0	"/usr/libexec/getty Pc"	cons25	on	secure
#	Virtual terminals				
	tttyv1	"/usr/libexec/getty Pc"	cons25	on	secure
	tttyv2	"/usr/libexec/getty Pc"	cons25	on	secure
	tttyv3	"/usr/libexec/getty Pc"	cons25	on	secure
	tttyv4	"/usr/libexec/getty Pc"	cons25	on	secure
	tttyv5	"/usr/libexec/getty Pc"	cons25	on	secure
	tttyv6	"/usr/libexec/getty Pc"	cons25	on	secure
	tttyv7	"/usr/libexec/getty Pc"	cons25	on	secure
	tttyv8	"/usr/X11R6/bin/xdm -nodaemon"	xterm	off	secure

Az állományban található oszlopok kimerítő magyarázatát, illetve a virtuális konzolok beállításához használható kapcsolatokat a ttys(5) man oldalon olvashatjuk.

3.2.5. Az egyfelhasználós mód konzolja

Az “egyfelhasználós mód” részletes leírása a 12.6.2 Szakaszban található. Fontos tudni, hogy amikor a FreeBSD-t egyfelhasználós módban futtatjuk, csupán egyetlen konzolunk van, és a virtuális konzolok nem érhetők el.

Egyébként az egyfelhasználós mód erre vonatkozó beállításai is megtalálhatóak az `/etc/ttys` állományban. Ehhez keressük meg a `console` kezdetű sort:

```
# name  getty                                type    status    comments
#
# Ha a konzolt "insecure" (nem biztonságos) típusúnak választjuk meg,
# akkor a használatához az egyfelhasználós mód aktiválásá előtt a rendszer
# kérni fogja a rendszeradminisztrátori jelszót.
console none                                unknown off secure
```

Megjegyzés: A `console` felett látható megjegyzés jelzi, hogy át tudjuk írni ebben a sorban a `secure` (biztonságos) értékű paramétert `insecure` (nem biztonságos) értékűre. Ilyenkor, hogy ha a FreeBSD egyfelhasználós módban indul, kérni fogja a `root` felhasználó (a rendszeradminisztrátor) jelszavát.

Vigyázzunk, amikor ezt az értéket `insecure`-ra állítjuk! Ha ugyanis véletlenül elfeledkeznénk a `root` jelszaváról, akkor azzal az egyfelhasználós mód használata is veszélybe kerülhet. Habár ettől függetlenül is lehetséges, azokra számára mégis nehéz helyzetnek bizonyulhat, akik nem mozognak elég otthonosan a FreeBSD rendszerindítási folyamatának és a hozzákapcsolódó programok ismeretében.

3.2.6. A videomód váltása konzolban

A FreeBSD konzol alapértelmezett videomódja átállítható 1024x768-ra, 1280x1024-re, vagy bármilyen olyan más méretre, amit a videokártyánk és monitorunk képes megjeleníteni. Az eltérő videomódok használatához először újra kell fordítanunk a rendszermagunkat az alábbi két beállítás hozzáadásával:

```
options VESA
options SC_PIXEL_MODE
```

Miután a rendszermagot sikeresen újrafordítottuk a fenti beállításokkal, a `vidcontrol(1)` segédprogrammal tudjuk megállapítani, hogy a hardverünk milyen videomódokat enged használni. Az összes támogatott videomódot a következőképpen tudjuk lekérdezni:

```
# vidcontrol -i mode
```

A parancs eredményeképpen tehát megkapjuk a hardverünk által ismert videomódokat. Ezek közül tudjuk kiválasztani valamelyikőjüket és `root` felhasználóként a `vidcontrol(1)` segítségével beállítani:

```
# vidcontrol MODE_279
```

Ha az új videomód megfelel számunkra, akkor ezt a beállítást az `/etc/rc.conf` állományon keresztül véglegesíthetjük is:

```
allscreens_flags="MODE_279"
```

3.3. Engedélyek

A FreeBSD, mivel a BSD UNIX egyik közvetlen leszármazottja, számos UNIX-os alapötletre épül. Ezek közül az első és talán a leginkább kihangsúlyozott, hogy a FreeBSD egy többfelhasználós operációs rendszer. Egy olyan rendszer, amely egyszerre több, egymástól független feladattal foglalkozó felhasználót képes kiszolgálni. A rendszer felelős a hardveres eszközök, a különféle perifériák, a memória és a processzor idejének minden egyes felhasználó számára szabályos és pártatlan megosztásáért és a feljük irányuló kérések szervezéséért.

Mivel a rendszer több felhasználót is képes támogatni, az általa kezelt erőforrások rendelkeznek engedélyek egy adott halmazával, amelyek eldöntik, ki tudja ezeket olvasni, írni és végrehajtani. Az engedélyek háromszor három bit formájában jelennek meg, amelyek közül az első bitszórt az állomány tulajdonosára, a második az állomány csoportjára, végül az utolsó pedig a mindenki másra vonatkozó engedélyeket tárolja.

Érték	Engedély	Könyvtárlistában
0	Nem olvasható, nem írható, nem hajtható végre	---
1	Nem olvasható, nem írható, végrehajtható	--x
2	Nem olvasható, írható, nem hajtható végre	-w-
3	Nem olvasható, írható, végrehajtható	-wx
4	Olvasható, nem írható, nem hajtható végre	r--
5	Olvasható, nem írható, végrehajtható	r-x
6	Olvasható, írható, nem hajtható végre	rw-
7	Olvasható, írható, végrehajtható	rwX

A `ls(1)` `-l` kapcsolójának segítségével megnézhetjük a könyvtárak tartalmának részletes listáját, amiben megjelennek az állományok tulajdonosaira, csoportjára és a mindenki másra vonatkozó engedélyek is. Például ezt láthatjuk, ha kiadjuk az `ls -l` parancsot egy tetszőleges könyvtárban:

```
% ls -l
total 530
-rw-r--r-- 1 root wheel 512 Sep 5 12:31 egyik
-rw-r--r-- 1 root wheel 512 Sep 5 12:31 másik
-rw-r--r-- 1 root wheel 7680 Sep 5 12:31 e-mail.txt
...
```

A példabeli `ls -l` parancs kimenetének első oszlopa így bomlik fel:

```
-rw-r--r--
```

Az első (bal szélső) karakter mondja meg, hogy ez egy hagyományos állomány, könyvtár, speciális karakteres eszköz, socket vagy bármilyen más különleges pszeudoállomány. Ebben az esetben a `-` jelzi, hogy egy hagyományos állományról van szó. A következő három karakter, ami ebben a példában az `rw-`, adja meg az állomány tulajdonosának engedélyeit. Az ezután következő három karakter, az `r--` mutatja az állomány csoportjának engedélyeit. Az utolsó három karakter, vagyis itt az `r--` adja meg a többiek engedélyeit. A kötőjel arra utal, hogy az adott engedélyű tevékenység nem engedélyezett. Tehát ennél az állománynál az engedélyek a következők: a

tulajdonosa tudja olvasni és írni, a csoportja csak olvasni tudja, ugyanígy bárki más. A fenti táblázatnak megfelelően az állomány engedélyének kódja 644 lesz, ahol az egyes számjegyek jelentik az állomány engedélyeinek három elemét.

Ez mind szép és jó, de vajon a rendszer milyen módon kezeli az állományok engedélyeit? A FreeBSD a legtöbb hardveres eszközt állománynak tekinti, amelyeket a programok meg tudnak nyitni, tudnak róluk olvasni és adatokat tudnak kiírni rájuk pontosan úgy, mint bármilyen más állomány esetén. Ezeket a speciális állományokat a `/dev` könyvtárban találjuk.

A könyvtárakat is állományokként kezeli, ezért azok is rendelkeznek olvasási, írási és végrehajtási engedélyekkel. Azonban a könyvtárak végrehajtását engedélyező bit némileg más jelentéssel bír, mint az állományok esetén. Amikor ugyanis egy könyvtárat végrehajthatónak jelölünk meg, az arra fog utalni, hogy bele tudunk lépni, vagyis hogy ki tudjuk rá adni a “könyvtárváltás” (`cd`, `change directory`) parancsát. Ez továbbá arra is utal, hogy az ismert nevű állományokhoz hozzá tudunk férni (természetesen az egyes állományok engedélyeinek megfelelően).

A könyvtárak tartalmát ennek megfelelően viszont csak úgy láthatjuk, ha olvasási engedéllyel rendelkezünk a könyvtárra, míg egy általunk ismert állomány törléséhez a tartalmazó könyvtárhoz kell írási és végrehajtási engedélyekkel rendelkezünk.

Ezeket kívül még léteznek további engedélyek is, de ezeket csak olyan különleges esetekben használják, mint például a felhasználóváltó programok (`setuid` program) vagy a ragadós könyvtárak (`sticky directory`) létrehozása. Az állományok engedélyeinek behatóbb megismeréséhez és beállításához mindenképpen nézzük át a `chmod(1)` man oldalt.

3.3.1. Szimbolikus engedélyek

Írta: Tom Rhodes.

A szimbolikus engedélyek (gyakran csak szimbolikus kifejezések) az állományok és könyvtárak engedélyeinek megadása során a számok helyett karaktereket használnak. A szimbolikus kifejezések (ki) (hogyan) (milyen engedélyt) alakúak, ahol az alábbi értékek adhatóak meg:

Elem	Betű	Jelentése
(ki)	u	tulajdonos
(ki)	g	csoport tulajdonos
(ki)	o	egyéb
(ki)	a	mindenki (a “világ”)
(hogyan)	+	engedély megadása
(hogyan)	-	engedély visszavonása
(hogyan)	=	engedély explicit beállítása
(milyen engedély)	r	olvasás
(milyen engedély)	w	írás
(milyen engedély)	x	végrehajtás
(milyen engedély)	t	ragadós (sticky bit)
(milyen engedély)	s	UID vagy GID állítása

Ezek az értékek a `chmod(1)` paranccsal az eddigiekhez hasonló módon használhatóak, csak itt betűket kell megadnunk. Például az alábbi paranccsal akadályozhatjuk meg, hogy a tulajdonosán kívül bárki hozzáférhessen az `ÁLLOMÁNY` nevű állományhoz:

```
% chmod go= ÁLLOMÁNY
```

Amennyiben egy állománnyal kapcsolatban több változtatást is el kívánunk végezni, össze tudjuk ezeket fűzni egy vesszőkkel elhatárolt felsorolásban:

```
% chmod go-w,a+x ÁLLOMÁNY
```

3.3.2. A FreeBSD állományjelzői

Írta: Tom Rhodes.

A korábban tárgyalt engedélyek mellett még a FreeBSD ismeri az ún. “állományjelzők” (file flags) beállítását is. Ezek a jelzőbitek egy további biztonsági és irányítási szintet nyújtanak az állományok felett, viszont a könyvtárakra nem vonatkoznak.

Ezek az állományjelzők az állományok felett további vezérlést adnak a kezünkbe, aminek révén gondoskodhatunk róla, hogy akár még a `root` felhasználó (a rendszer adminisztrátora) se legyen képes állományokat eltávolítani vagy módosítani.

Az állományjelzők értékei egy egyszerű felületen keresztül, a `chflags(1)` segédprogrammal változtathatóak meg. Például a következő paranccsal állíthatjuk a rendszer törölhetetlen (`undeletable`) jelzését az `allomany1` állományon:

```
# chflags sunlink allomany1
```

A törölhetetlen jelzés eltávolításához egyszerűen csak írjuk be az előző parancsot úgy, hogy a “`sunlink`” paraméter elejére még beszúrunk egy “`no`” szövegrészt. Így:

```
# chflags nosunlink allomany1
```

Az állományokra éppen érvényes jelzéseket az `ls(1)` parancs `-lo` kapcsolójának segítségével jeleníthetjük meg:

```
# ls -lo file1
```

Ennek megfelelően az eredménynek valahogy így kellene kinéznie:

```
-rw-r--r--  1 trhodes  trhodes  sunlnk 0 Mar  1 05:54 allomany1
```

Sok jelzés csak a `root` felhasználón keresztül vehető fel vagy távolítható el. Más esetekben viszont az állomány tulajdonosa állíthatja ezeket. A rendszergazdáknak javasoljuk, hogy ezzel kapcsolatban a `chflags(1)` és `chflags(2)` man oldalakat tanulmányozzák át.

3.3.3. A `setuid`, `setgid` és `sticky` engedélyek

Készítette: Tom Rhodes.

A korábban említett engedélyeken kívül létezik még további három, amelyekkel minden rendszergazdának illik tisztában lennie. Ezek név szerint a `setuid`, `setgid` és `sticky` típusú engedélyek.

Ezek a beállítások bizonyos UNIX műveletek esetén nagyon fontosak, mivel az átlagos felhasználók számára általában el nem érhető funkciók használatát támogatják. A megértésükhöz elsőként a felhasználók valódi és effektív azonosítója közti különbségeket kell tisztáznunk.

A valódi azonosító tulajdonképpen az a felhasználói azonosító, amellyel a programot indítjuk el vagy futás előtt birtokoljuk. A program futása közben azonban az effektív felhasználói azonosítóval fut. Például a `passwd(1)` segédprogram a jelszavát megváltoztatni kívánó felhasználó valódi azonosítójával indul, miközben a jelszavakat tároló adatbázis elérésékor már a `root` felhasználó effektív azonosítójával fut. Ezáltal a privilégiumokkal nem rendelkező felhasználók is meg tudják anélkül változtatni a jelszavaikat, hogy a `Permission Denied` hibaüzenettel találkozának.

Megjegyzés: A `mount(8)` `nosuid` beállításával azonban az ilyen típusú binárisok minden különösebb jel nélkül csődöt fognak mondani. Mellesleg a `mount(8)` man oldala szerint ez az opció nem is teljesen megbízható, mivel `nosuid` wrapperek segítségével meg lehet kerülni.

Ahogy azt az alábbi példa is szemlélteti, a `setuid` engedélyt a többi elé egy négyes (4) beszúrásával tudjuk beállítani:

```
# chmod 4755 suidexample.sh
```

A `suidexample.sh` állomány engedélyei ezt követően már így fognak megjelenni:

```
-rwsr-xr-x  1 trhodes  trhodes   63 Aug 29 06:36 suidexample.sh
```

Most már jól látható, hogy az állomány tulajdonosához tartozó engedélyek között a végrehajthatóságot szabályozó bit lecserélődött egy `s` bitre. Ennek köszönhetően a `passwd` parancshoz hasonló módon kibővített engedélyekkel leszünk képesek futtatni programokat.

Két terminál megnyitásával mindezt valós időben is megvizsgálhatjuk. Az egyiket indítsuk el normál felhasználóként a `passwd` programot. Miközben a program várakozik az új jelszó megadására, a másik terminálon kérdezzük le a programhoz tartozó felhasználói információkat.

Tehát az egyik terminálon a következőt látjuk:

```
% passwd
Changing local password for trhodes
Old Password:
```

Eközben pedig a másikon:

```
# ps aux | grep passwd
trhodes  5232  0.0  0.2  3420  1608   0  R+   2:10AM  0:00.00 grep passwd
root      5211  0.0  0.2  3620  1724   2  I+   2:09AM  0:00.01 passwd
```

A `passwd` parancsot egyszerű felhasználóként adtunk ki, azonban jól látható, hogy valójában a `root` felhasználó azonosítójával fut.

A `setgid` a `setuid` engedélyhez hasonlóan működik, egyedül annyiban tér el, hogy a csoportra vonatkozó beállításokat módosítja. Amikor egy alkalmazást vagy segédprogramot ilyen engedéllyel futtatunk, akkor az adott programot birtokló csoport engedélyeit kapjuk meg.

Úgy tudjuk állományokon beállítani a `setgid` típusú engedélyt, ha az iménti példához hasonlóan a `chmod` parancs hívásakor még egy kettest (2) írunk az engedélyek elé:

```
# chmod 2755 sgidexample.sh
```

Az így beállított engedélyek az előbbihez hasonló módon szemléltethetők meg, azonban ebben az esetben a csoporthoz tartozó engedélyeknél jelenik meg az `s` bit:

```
-rwxr-sr-x  1 trhodes  trhodes    44 Aug 31 01:49 sgidexample.sh
```

Megjegyzés: Az előbb tárgyalt példákkal kapcsolatban fontos megemlítenünk, hogy habár a szkriptek is végrehajtható állományok, nem fognak a valóditól eltérő effektív felhasználói azonosítóval futni. Ennek oka abban keresendő, hogy a parancssori szkriptek nem hívhatják a `setuid(2)` rendszerhívást.

Ez a két speciális engedély (a `setuid` és a `setgid`) a programhoz tartozó engedélyek kiterjesztésével csökkentheti rendszerünk biztonságát. Ezzel szemben viszont a harmadik bemutatandó speciális engedély rendszerünk védelmének erősítésére szolgál: ez az ún. `sticky` bit.

Ha a `sticky` típusú engedélyt könyvtárra adjuk meg, akkor a benne levő állományok törlését kizárólag azok tulajdonosainak engedi. Ezzel az engedéllyel lényegében a `/tmp` könyvtárhoz hasonló nyilvános, bárki által elérhető könyvtárakban akadályozhatjuk meg az állományok idegen felhasználók általi törlését. Az engedély beállításához egy egységet (1) kell a többi elé fűznünk, mint például:

```
# chmod 1777 /tmp
```

Most már az `ls` parancs segítségével láthatjuk ennek a hatását:

```
# ls -al / | grep tmp
drwxrwxrwt  10 root  wheel          512 Aug 31 01:49 tmp
```

A `sticky` bit a beállítások végén felbukkanó `t` révén azonosítható be.

3.4. A könyvtárak elrendezése

A FreeBSD könyvtárszerkezetének ismerete alapvető jelentőségű a rendszer egészének megértése szempontjából. Ezen belül is a legfontosabb a gyökérkönyvtár, a `/`. Ez az első könyvtár, amelyet a rendszer a rendszerindítás során csatlakoztat és a többfelhasználós mód előkészítéséhez elegendhetlenül szükséges alaprendszert tartalmazza. A gyökérkönyvtár emellett csatlakozási pontokat szolgáltat a többfelhasználós működésre váltás során csatlakoztatandó további állományrendszerek számára.

A csatlakozási pont egy olyan könyvtár, ahová a szülő állományrendszeren (ami gyakran maga a gyökér-állományrendszer) belül további állományrendszereket tudunk beoltani. Erről bővebben a 3.5 Szakaszban olvashatunk. A szabványos csatlakozási pontok: `/usr`, `/var`, `/tmp`, `/mnt` és `/cdrom`. Ezekre a könyvtárakra általában az `/etc/fstab` állományban találunk hivatkozásokat. Az `/etc/fstab` állomány a rendszer számára a különböző állományrendszerek és a hozzájuk tartozó csatlakozási pontok táblázatát tartalmazza. Az `/etc/fstab` állományban szereplő legtöbb állományrendszer a rendszerindítás során automatikusan csatlakoztatásra kerül az `rc(8)` szkriptből, hacsak nem tartalmazzák a `noauto` beállítást. Ennek részleteit a 3.6.1 Szakaszban találhatjuk meg.

Az állományrendszerek hierarchiájának teljes leírását a `hier(7)` man oldalon olvashatjuk. Mi egyelőre most megelégszünk a leggyakrabban megjelenő könyvtárak rövid áttekintésével.

Könyvtár

Mi található itt

Könyvtár	Mi található itt
/	Az állományrendszer gyökere.
/bin/	Az egy- és többfelhasználós környezetekben is egyaránt alapvető felhasználói segédprogramok.
/boot/	Az operációs rendszer indítása során használt programok és konfigurációs állományok.
/boot/defaults/	A rendszerindítás alapértelmezett konfigurációs állományai. Lásd loader.conf(5)
/dev/	Eszközeleírók, lásd intro(4).
/etc/	Rendszerkonfigurációs állományok és szkriptek.
/etc/defaults/	Az alapértelmezett rendszerkonfigurációs állományok, lásd rc(8).
/etc/mail/	A sendmail(8) programhoz hasonló levélküldő rendszerek konfigurációs állományai.
/etc/namedb/	A named program konfigurációs állományai, lásd named(8).
/etc/periodic/	A cron(8) által naponta, hetente és havonta lefuttatandó szkriptek, lásd periodic(8).
/etc/ppp/	A ppp program konfigurációs állományai, lásd ppp(8).
/mnt/	Egy üres könyvtár, amelyet a rendszergazdák általában ideiglenes csatlakozási pontként használnak.
/proc/	A futó programokat tartalmazó állományrendszer, lásd procfs(5), illetve mount_procfs(8).
/rescue/	Statikusan linkelt programok vészhelyzet esetére, lásd rescue(8).
/root/	A root felhasználó könyvtára.
/sbin/	Az egy- és többfelhasználós környezetekben fontos rendszerprogramok és rendszerfelügyeleti eszközök.
/tmp/	Átmeneti állományok. A /tmp könyvtár tartalma általában NEM marad meg az újraindítás után. Erre a célra gyakran memóriában létrehozott állományrendszert szoktak csatlakoztatni a /tmp könyvtárba. Ez utóbbit az rc.conf(5) tmpmfs-re vonatkozó változóinak beállításával lehet automatikussá tenni (vagy a /etc/fstab megfelelő módosításával, lásd mdmfs(8)).
/usr/	A felhasználói programok és alkalmazások többsége.
/usr/bin/	Általános segédprogramok, programozási eszközök és alkalmazások.
/usr/include/	Szabványos C include-állományok.
/usr/lib/	Függvénykönyvtárak.
/usr/libdata/	Egyéb hasznos adatállományok.
/usr/libexec/	(Más programok által használt) Rendszerdémonok és rendszereszközök.

Könyvtár`/usr/local/``/usr/obj/``/usr/ports/``/usr/sbin/``/usr/share/``/usr/src/``/usr/X11R6/``/var/``/var/log/``/var/mail/``/var/spool/``/var/tmp/``/var/yp`**Mi található itt**

A helyi rendszeren telepített programok, függvénykönyvtárak stb. A FreeBSD portrendszere is ezt használja alapértelmezés szerint. A `/usr/local` könyvtárban belül a hier(7) man oldalon található `/usr` könyvtár általános felépítése használatos. Ez alól kivételt képez a man alkönyvtár, amely közvetlenül a `/usr/local` alatt található, nem pedig a `/usr/local/share` könyvtárban belül, valamint a portok dokumentációja a `share/doc/port` könyvtárban található.

A `/usr/src` könyvtárban található források fordítása során keletkező architektúrafüggő objektumok.

A FreeBSD Portgyűjtemény (választható).

(A felhasználók által használt) Rendszerdémonok és rendszereszközök.

Architektúrafüggő állományok.

BSD és/vagy helyi források.

Az X11R6 rendszer programjai, függvénykönyvtárai stb. (választható)

Különbféle napló, átmeneti, ideiglenes és pufferben tárolt állományok. A memóriában létrehozott állományrendszereket is olykor a `/var` könyvtárban találjuk. Ezt az `rc.conf(5)` állományban található `varmfs-változók` beállításával tehetjük automatikussá (vagy a `/etc/fstab` megfelelő módosításával, lásd `mdmfs(8)`).

Mindenféle rendszernaplók.

A felhasználók postafiókjait tároló állományok.

A nyomtatók és a levelezés puffereléséhez használt könyvtárak.

Átmeneti állományok. Az itt található állományok általában megmaradnak a következő rendszerindítás alkalmával is, hacsak a `/var` nem egy memóriában létező állományrendszer.

A NIS állományai.

3.5. A lemezek szervezése

Az állománynév a legkisebb szervezési egység, amin keresztül a FreeBSD képes megtalálni az állományokat. Az állományok neveiben a kis- és nagybetűt megkülönböztetjük, tehát a `readme.txt` és a `README.TXT` elnevezés két különböző állományra utal. A FreeBSD nem az állományok kiterjesztése (ami a konkrét példánkban a `.txt` volt) alapján dönti el, hogy az adott állomány vajon program, dokumentum vagy valamilyen más fajtájú adat.

Az állományok könyvtárakban tárolódnak. Egy könyvtár lehet akár üres (nincs benne egyetlen állomány sem), vagy többszáz állományt is tartalmazhat. Egy könyvtár ráadásul további könyvtárakat is tárolhat, és így az egymásban elhelyezkedő könyvtárak segítségével könyvtárak egy hierarchiáját tudjuk felépíteni. Ezzel sokkalta könnyebben szervezhetővé válnak az adataink.

Az állományokat és könyvtárakat úgy tudjuk elérni, ha megadjuk az állomány vagy a könyvtárt tároló könyvtár nevét, amit egy perjel, a / követ, valamint így összefűzve az eléréshez szükséges további könyvtárak felsorolása. Tehát, ha van egy *ize* nevű könyvtárunk, amelyben található egy *mize* könyvtár, amelyen belül pedig egy *readme.txt*, akkor ennek az állománynak a teljes neve, vagy másképpen szólva az *elérési útja* *ize/mize/readme.txt* lesz.

A könyvtárak és az állományok egy állományrendszerben tárolódnak. Minden állományrendszer pontosan egy könyvtárat tartalmaz a legfelső szintjén, amelyet az adott állományrendszer *gyökérkönyvtárának* nevezünk. Ez a gyökérkönyvtár tartalmazhat aztán további könyvtárakat.

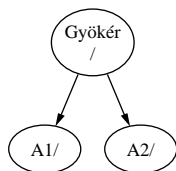
Eddig még valószínűleg minden nagyon hasonló a más operációs rendszerekben tapasztalható fogalmakhoz. Azonban adódnak különbségek: például az MS-DOS a \ jellel választja el az állományok és könyvtárak neveit, miközben a Mac OS® erre a : jelet használja.

A FreeBSD az elérési utakban sem betűkkel, sem pedig semmilyen más névvel nem jelöli meg a meghajtókat. Tehát a FreeBSD-ben nem írhatjuk, hogy a *c:/ize/mize/readme.txt*.

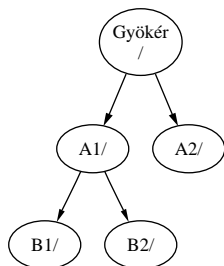
Helyette az egyik állományrendszert kijelölik *gyökér-állományrendszernek*. A gyökér-állományrendszer gyökérkönyvtárára hivatkoznak később / könyvtárként. Ezután minden más állományrendszert a gyökér-állományrendszerhez *csatlakoztatunk*. Ennek értelmében nem számít, hogy mennyi lemezünk is van a FreeBSD rendszerünkben, hiszen minden könyvtár egyazon lemez részeként jelenik meg.

Tegyük fel, hogy van három állományrendszerünk, hívjuk ezeket A-nak, B-nek és C-nek. Minden állományrendszer rendelkezik egy gyökérkönyvtárral, amely két további könyvtárat tartalmaz: A1-et és A2-t (és ennek megfelelően a többi B1-et és B2-t, valamint C1 és C2-t).

Nevezzük A-t a gyökér-állományrendszernek. Ha a könyvtár tartalmának megjelenítéséhez most kiadnánk az *ls* parancsot, két alkönyvtárat látnánk, az A1-et és A2-t. A létrejött könyvtárfa valahogy így nézne ki:

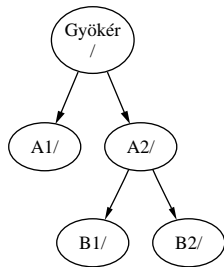


Egy állományrendszert csak egy másik állományrendszer valamelyik könyvtárába tudunk csatlakoztatni. Ezért most tételezzük fel, hogy a B állományrendszert az A1 könyvtárba csatlakoztatjuk. Ezután a B gyökérkönyvtára átveszi a A1 helyét az állományrendszerben, és ennek megfelelően megjelennek a B könyvtárai is:



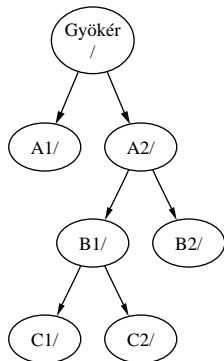
A B1 vagy B2 könyvtárakban található állományok bármelyike innentől kezdve a /A1/B1, illetve a /A1/B2 elérési utakon érhetőek el. Az A1 könyvtárban található állományok erre az időre rejtve maradnak. Akkor fognak újra felbukkanni, ha a B állományrendszert *leválasztjuk* az A állományrendszerről.

Ha a B állományrendszert az A2 könyvtárba csatlakoztatnánk, az iménti ábra nagyjából így nézne ki:

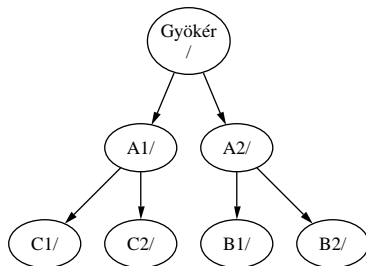


és ennek megfelelően az előbb tárgyalt elérési utak /A2/B1 és /A2/B2 lennének.

Az állományrendszerek egymáshoz is csatlakoztathatóak. A példát ennek megfelelően úgy is folytathatjuk, hogy a C állományrendszert csatlakoztatjuk B állományrendszerben található B1 könyvtárhoz. Ennek eredménye a következő elrendezés lesz:



Vagy a C állományrendszer az A1 könyvtáron keresztül csatlakoztatható akár közvetlenül az A állományrendszerhez is:



Az MS-DOS operációs rendszert ismerők számára ez hasonló lehet a `join` parancshoz (habár teljesen nem egyezik meg vele).

Általában azonban ezzel nem kell törődnünk, hiszen többnyire csak a FreeBSD telepítése során hozunk létre állományrendszereket és választjuk meg a csatlakozási pontjukat. A későbbiekben ez legfeljebb akkor kerül elő ismét, amikor újabb lemezeket adunk hozzá a rendszerhez.

Teljességgel megengedhető, hogy elhagyjuk a többbit és csak egyetlen óriási gyökér-állományrendszert használjunk. Ennek viszont megvannak a maga hátrányai és az egyetlen előnye.

Több állományrendszer használatának előnyei

- A különböző állományrendszereknek különböző *csatlakoztatási beállításai* (mount options) lehetnek. Például, ha kellően elővigyázatosak akarunk lenni, a gyökér-állományrendszer írásvédett módon is csatlakoztatható, aminek köszönhetően lehetetlenné válik a rendszer számára fontos állományok véletlen törlése vagy felülírása. Ha elkülönítjük a felhasználók számára írható állományrendszereket (például a /home könyvtárakat) a többi állományrendszertől, lehetővé válik számunkra, hogy *nosuid* beállítással csatlakoztassuk ezeket. Ez a beállítás megakadályozza, hogy ezekben a *suid/guid* bitekkel rendelkező végrehajtható állományok használhatóak legyenek, ezáltal növeli a rendszer biztonságosságát.
- A FreeBSD az állományrendszer használatától függően magától határozza meg a benne található állományok optimális kiosztását illetően. Így tehát a gyakorta módosított, kisebb állományokat tartalmazó állományrendszerek esetén teljesen más technikákat alkalmaz, mint például a nagyobb, kevésbé változó állományok esetén. Azonban egyetlen állományrendszer használatával ez a gyorsítási módszer odavész.
- Noha a FreeBSD állományrendszerei nagyon jól tűrik a hirtelen áramkimaradásokat, egy döntő ponton bekövetkező váratlan leállás mégis kárt okozhat a szerkezetükben. Ha azonban több állományrendszerre osztjuk a tárolandó adatainkat, sokkal valószínűbbé válik, hogy egy ilyen eset után a rendszerünk talpra tud állni, és szükség esetén nekünk is könnyebb lesz a biztonsági mentéseinkből helyreállítani a sérült állományokat.

Egyetlen állományrendszer használatának előnyei

- Az állományrendszerek mérete rögzített. Miután a FreeBSD telepítése során létrehoztunk egy adott méretű állományrendszert, előfordulhat, hogy később szükségünk lesz a méretének növelésére. Ilyenkor nehezen kerülhetjük el az ilyenkor szokásos teendőket: biztonsági mentés készítése, az új méretnek megfelelő állományrendszer létrehozása, majd ezután a lementett adataink visszaállítása.

Fontos: A FreeBSD-ben azonban megtalálható a *growfs(8)* parancs, amelynek segítségével az állományrendszerek mérete használat közben növelhető, és ezzel megszűnik a méretre vonatkozó korlátozás.

Az állományrendszerek partíciókban tárolódnak. A FreeBSD UNIX-os eredete miatt azonban ez a kifejezés nem a hétköznapi "partíció" jelentését takarja (mint például egy MS-DOS partíció). Minden partíciót egy betű azonosít a-tól h-ig. Mindegyik partíció csak egyetlen állományrendszert tartalmazhat, aminek révén az állományrendszereket vagy az állományrendszerek hierarchiájában található csatlakozási pontjukkal vagy pedig az ezeket tartalmazó partíció betűjével azonosíthatjuk.

A FreeBSD ezeken felül külön lemezterületen tárolja a *lapozóállományt* (swap space). A lapozóállományt használja a FreeBSD *virtuális memória* (virtual memory) megvalósításához. Ennek köszönhetően a számítógép képes úgy viselkedni, mintha jóval több memóriával rendelkezne, mint valójában. Így, amikor a FreeBSD kifogy a memóriából, egyszerűen kirakja a memóriából a lapozóállományba az éppen nem használt adatokat, majd amikor ismét szüksége lesz rájuk, visszatölti ezeket (és ilyenkor megint kirak valami mást).

Némely partícióhoz kötődnek bizonyos megszokások.

Partíció	Megszokás
a	Általában ez tartalmazza a gyökér-állományrendszert.

Partíció	Megszokás
b	Általában ez tartalmazza a lapozóállományt.
c	Mérete általában a tartalmazó slice méretével egyezik meg. Ennek köszönhetően a segédprogramok (például egy hibás szektorokat kereső program) a c partíción keresztül képesek akár az egész slice-szal dolgozni. Normális esetben ezen a partíción nem hozunk létre állományrendszert.
d	A d partícióhoz egykoron kapcsolódott különleges jelentés, azonban mostanra ez már megszűnt, és a d egy teljesen átlagos partíciónak tekinthető.

Minden állományrendszert tartalmazó partíciót a FreeBSD egy ún. *slice*-ban tárol. A FreeBSD számára a slice elnevezés utal mindarra, amit általában partíciónak neveznek, és ismét megemlíjtük, mindez a UNIX-os eredet miatt. A slice-okat 1-től 4-ig sorszámozzák.

A slice-ok sorszáma 1-től indulva az eszközök neve után egy s betűvel elválasztva következik. Így tehát a “da0s1” jelentése az első slice lesz az első SCSI-meghajtón. Lemezenként négy fizikai slice hozható létre, de ezeken belül tetszőleges típusú logikai slice-ok helyezhetők el. Ezen további slice-ok sorszámozása 5-től kezdődik, így ennek megfelelően a “ad0s5” lesz az első IDE-lemezen található első kiterjesztett slice. Ezeket az eszközöket foglalják el a különböző állományrendszerek.

A slice-ok, a “veszélyesen dedikált” (Dangerously Dedicated) fizikai meghajtók, és minden más olyan meghajtó, amely *partíciókat* tartalmaz, a-tól h-ig jelölődnek. Ez a betű az eszköz neve után következik, így ennek megfelelően a “da0a” lesz az első “da” meghajtó “a”, vagyis a “veszélyesen dedikált” partíciója. Az “ad1s3e” lesz a második IDE-lemezmeghajtón a harmadik slice-ban szereplő ötödik partíció.

Végezetül, a rendszerben minden lemezt azonosítunk. A lemez neve a típusára utaló kóddal kezdődik, amely után aztán egy sorszám jelzi, hogy melyik lemeze is van szó. Azonban eltérően a slice-okétól, a lemezek sorszámozása 0-tól indul. Az általánosan elterjedt kódolások a 3-1 Táblázatban találhatók.

Amikor hivatkozunk egy partícióra, a FreeBSD elvárja tőlünk, hogy nevezzük meg az adott partíciót tartalmazó slice-ot és lemezt is. Emiatt egy partícióra mindig úgy hivatkozunk, hogy először megadjuk a tartalmazó lemez nevét, ettől s-sel elválasztva a tartalmazó slice sorszámát, majd ezt a partíció betűjelével zárjuk. Erre példákat a 3-1 Példában láthatunk.

Az érthetőség kedvéért a 3-2 Példa bemutatja egy lemez kiosztásának fogalmi sablonját.

A FreeBSD telepítéséhez először be kell állítani a lemezen található slice-okat, majd létrehozni benne a FreeBSD-hez használni kívánt partíciókat, kialakítani rajtuk az állományrendszereket (vagy a lapozóállományt) és eldönteni, melyik állományrendszert kívánjuk csatlakoztatni.

Táblázat 3-1. Lemezes eszközök kódjai

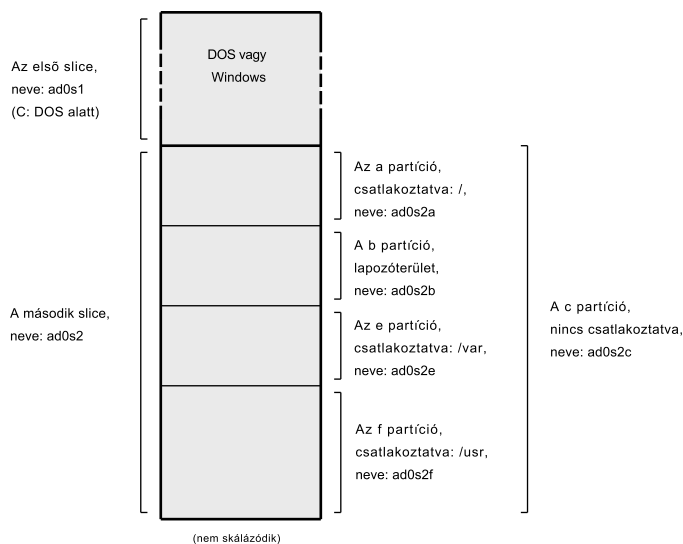
Kód	Jelentés
ad	ATAPI (IDE) lemez
da	közvetlen hozzáférésű SCSI lemez
acd	ATAPI (IDE) CDROM
cd	SCSI CDROM
fd	Floppylemez

Példa 3-1. Példák lemezek, slice-ok és partíciók neveire

Név	Jelentés
ad0s1a	Az első IDE lemezen (ad0) levő első slice (s1) első partíciója (a).) ötödik partíciója (e).

szemszögéből. Tegyük fel, hogy ez a (S partíciót) tartalmaz. Az első slice egy odik slice egy telepített FreeBSD-t nyt tároló partícióval rendelkezik.

a partíció lesz a
; a /usr könyvtár.



3.6. Állományrendszerek csatlakoztatása és leválasztása

Az állományrendszereket legkönnyebben egy-egy faként tudjuk magunk előtt elképzelni, amelyek a / könyvtárból nőnek ki. A /dev, /usr és mellettük szereplő, hozzájuk hasonló összes többi könyvtár csupán egy-egy ág, amelyeknek saját ágaik is lehetnek, mint például a /usr/local és így tovább.

Különféle okai vannak annak, hogy egyes könyvtárakat különálló állományrendszereken tárolunk. A /var könyvtár tartalmazza a log/, spool/ könyvtárakat és különféle átmeneti állományokat, azonban az ilyen állományok könnyen megszaporodhatnak és megtölthetik az állományrendszert. Mivel a gyöker-állományrendszert nem tanácsos

elárasztani mindenféle állománnyal, ezért gyakran a hasznunkra válhat, ha a `/var` könyvtárat leválasztjuk a `/` könyvtárból.

A másik gyakori ok, ami az imént említett fá egyes ágainak különböző állományrendszereken történő tárolását indokolja, hogy ezek gyakran más fizikai vagy virtuális lemezekon, például a rendszerhez csatlakoztatott Hálózati állományrendszereken vagy éppen CD-meghajtókon találhatók.

3.6.1. Az `fstab` állomány

A rendszerindítás folyamata során az `/etc/fstab` állományban felsorolt állományrendszerek maguktól kerülnek csatlakoztatásra (kivéve amikor a `noauto` beállítással szerepelnek).

Az `/etc/fstab` állományban található sorok az alábbi szerkezetűek:

```
eszköz           /csatlakozási-pont típus      beállítások      mentésigyak      ellszám
```

`eszköz`

A 18.2 Szakaszban leírtak szerint megnevezett (létező) eszköz.

`csatlakozási-pont`

Egy (létező) könyvtár, ahova az állományrendszer csatlakozik.

`típus`

Az állományrendszer `mount(8)` parancs szerint ismert típusa. A FreeBSD alapértelmezett állományrendszere az `ufs`.

`beállítások`

Az írható-olvasható állományrendszerek esetén `rw`, az írásvédettek esetén pedig `ro`, amelyet igény szerint további beállítások követhetnek. A rendszerindítás során automatikusan nem csatlakoztatandó állományrendszerek esetén gyakran alkalmazott beállítás itt még a `noauto`. Egyéb lehetőségeket a `mount(8)` man oldalon láthatunk.

`mentésigyak`

Ezt általában a `dump(8)` parancs használja a menteni szükséges állományrendszerek megállapításához. Amennyiben hiányzik ez a mező, az automatikusan a nulla értéket jelöli.

`ellszám`

Megadja, hogy mely állományrendszereket kell ellenőrizni. A nullás `pass` értékkel rendelkező állományrendszerek nem kerülnek ellenőrzésre. A gyökér-állományrendszer (melyet minden más előtt kell ellenőrizni) `passno` értéke egy, míg az összes többi állományrendszer `passno` értéke általában egytől különböző. Ha egynél több állományrendszer is ugyanazt a `passno` értéket kapta, akkor az `fsck(8)` a lehetőségei szerint megpróbálja ezeket egyszerre ellenőrizni.

Az `/etc/fstab` felépítéséről és a benne használható beállításokról bővebben a `fstab(5)` man oldalon olvashatunk.

3.6.2. A `mount` parancs

Az állományrendszerek tényleges csatlakoztatására avagy “mountolására” a `mount(8)` parancs használható.

Legegyszerűbb formája:

```
# mount eszköz csatlakozási-pont
```

Ahogy a `mount(8)` man oldalán is olvashatjuk, itt rengeteg opció is megadható, de ezek közül a leggyakoribbak:

Csatlakoztatási opciók

-a

Csatlakoztatja az `/etc/fstab` állományban felsorolt összes állományrendszert, kivéve azokat, amelyek a “noauto” beállítást tartalmazzák, vagy kizártuk a `-t` kapcsolóval, esetleg korábban már csatlakoztattuk.

-d

A tényleges csatlakoztatás elvégzése nélkül végrehajt minden mást. Ez az opció leginkább `-v` opcióval együtt használható annak megállapítására, hogy a `mount(8)` valójában mit is akar csinálni.

-f

Egy nem tiszta állományrendszer csatlakoztatásának kényszerítése (veszélyes!) vagy egy korábban már csatlakoztatott állományrendszer írható állapotának felfüggesztése.

-r

Az állományrendszer írásvédett csatlakoztatása. Megegyezik a `-o` opciónál megadható `ro` (vagy a FreeBSD 5.2-nél régebbi verziója esetén a `rdonly`) beállítás használatával.

-t *típus*

Az adott állományrendszert az adott típusnak megfelelően csatlakoztatja, vagy az `-a` használata esetén csak az adott típusú állományrendszereket.

Az “ufs” az állományrendszerek alapértelmezett típusa.

-u

Frissíti az állományrendszerre vonatkozó csatlakoztatási beállításokat.

-v

Részletesebb kijelzés.

-w

Az állományrendszer csatlakoztatása írásra és olvasásra.

Az `-o` opció után vesszővel elválasztott beállításokat adhatunk meg, többek közt az alábbiakat:

`noexec`

Az állományrendszeren található állományok végrehajtásának tiltása. Ez egy nagyon hasznos biztonsági beállítás.

nosuid

Az állományrendszeren nem használhatóak a felhasználó- (setuid) vagy csoportváltásra (setgid) vonatkozó engedélyek. Nagyon hasznos biztonsági beállítás.

3.6.3. Az `umount` parancs

Az `umount(8)` parancs paraméterként egy csatlakozási pontot, egy eszköznevet vagy a `-a`, illetve az `-A` opciókat várja.

A leválasztás kényszerítéséhez mindegyik alakban szerepelhet az `-f` opció, valamint a részletesebb kijelzést a `-v` opcióval kapcsolhatjuk be. Azonban szeretnénk mindenkit figyelmeztetni, hogy a `-f` használata alapvetően nem ajánlott. Az erőszakkal leválasztott állományrendszerek összeomlaszthatják a számítógépet vagy kárt okozhatnak az állományrendszereken található adatokban.

Az `-a` és `-A` opciók használatosak az összes csatlakoztatott állományrendszer leválasztására, amelyek típusait a `-t` opció megadása után sorolhatjuk fel. Fontos különbség azonban, hogy az `-A` opció a gyöker állományrendszert nem próbálja meg leválasztani.

3.7. Folyamatok

A FreeBSD egy többfeladatos operációs rendszer. Ez azt jelenti, hogy képes látszólag egyszerre több programot is futtatni. Az így egyszerre futó programokat egyenként *folyamatoknak* (process) nevezzük. Minden kiadott parancsunk elindít legalább egy ilyen folyamatot, és a rendszerünk mozgásában tartásához bizonyos rendszerszintű folyamatok állandóan futnak a háttérben.

Minden folyamatot egy *folyamatazonosító*nak (process ID vagy *PID*) nevezett szám azonosít egyértelműen, és az állományokhoz hasonlóan, minden folyamatnak van tulajdonosa és csoportja is. A tulajdonos és a csoport ismeretében állapítja meg a rendszer, hogy az adott folyamat a korábban említett engedélyek szerint milyen állományokhoz és eszközökhöz férhet hozzá. Ezenkívül a legtöbb folyamatnak van még egy szülőfolyamata is. A szülőfolyamat az a folyamat, amely az adott folyamatot elindította. Például amikor parancsokat adunk egy parancsértelmezőn keresztül, akkor maga a parancsértelmező is egy ilyen folyamat lesz ugyanúgy, ahogy a benne kiadott parancsok által elindított programok. Ennek megfelelően az így létrehozott összes folyamat szülője maga a parancsértelmező folyamata lesz. Az említettek alól egyik kivétel az `init(8)` nevű speciális folyamat. Az `init` lesz a rendszerben mindig az első folyamat, ezért a *PID*-je is mindig 1. Az `init` programot a FreeBSD indulásakor a rendszermag fogja automatikusan elindítani.

A rendszerben futó programok vizsgálatához két, különösen hasznos parancsot találhatunk: ezek a `ps(1)` és a `top(1)`. A `ps` parancs használatos a pillanatnyilag futó programok statikus listájának megjelenítésére. Ebben olvashatjuk a futó programok azonosítóit, mennyi memóriát használnak éppen, milyen parancssal indították ezeket stb. A `top` parancs mutatja az összes aktívan futó programot, majd néhány másodpercenként automatikusan frissíti ezt a listát, aminek révén folyamatosan láthatjuk, miként viselkednek a futó programok.

A `ps` alapértelmezés szerint csupán az általunk futtatott programokat mutatja. Például:

```
% ps
  PID  TT  STAT      TIME COMMAND
   298  p0  Ss      0:01.10 tcsh
  7078  p0  S        2:40.88 xemacs mdoc.xml (xemacs-21.1.14)
37393  p0  I        0:03.11 xemacs freebsd.dsl (xemacs-21.1.14)
```

```

48630 p0 S      2:50.89 /usr/local/lib/netscape-linux/navigator-linux-4.77.bi
48730 p0 IW     0:00.00 (dns helper) (navigator-linux-)
72210 p0 R+    0:00.00 ps
   390 p1 Is    0:01.14 tcsh
   7059 p2 Is+  1:36.18 /usr/local/bin/mutt -y
   6688 p3 IWS   0:00.00 tcsh
  10735 p4 IWS   0:00.00 tcsh
  20256 p5 IWS   0:00.00 tcsh
   262 v0 IWS   0:00.00 -tcsh (tcsh)
   270 v0 IW+   0:00.00 /bin/sh /usr/X11R6/bin/startx -- -bpp 16
   280 v0 IW+   0:00.00 xinit /home/nik/.xinitrc -- -bpp 16
   284 v0 IW     0:00.00 /bin/sh /home/nik/.xinitrc
   285 v0 S      0:38.45 /usr/X11R6/bin/sawfish

```

Ahogy az a fenti példában is látszik, a `ps(1)` kimenete oszlopokra tagolható. Ezek közül a `PID` tartalmazza a korábban már ismertetett folyamatazonosítókat. Az azonosítók 1-től indulva egészen 99999-ig sorszámozódhatnak, illetve ha kifutnánk belőlük, akkor a számozás kezdődik előlről (azonban a használatban levő azonosítók sosem kerülnek újra kiosztásra). A `TT` oszlopban láthatjuk azt a terminált, amelyen az adott program éppen fut, de ezt pillanatnyilag akár nyugodtan figyelmen kívül is hagyhatjuk. A `STAT` oszlopban a program állapotát kapjuk meg, de szintén átugorható. A `TIME` a program processzoron eltöltött idejét mutatja — ez általában nem arra utal, hogy mennyi ideje fut maga a program, hiszen a legtöbb program sok időt tölt tétlenül, mielőtt egyáltalán szüksége lenne processzora. Végezetül a `COMMAND` oszlopban olvashatjuk azt a parancsot, amellyel a programot elindították.

A `ps(1)` számos különféle beállítást ismer az általa megjelenített információk megválasztásához. Az egyik ilyen leghasznosabb beállítás az `auxww`: az a segítségével az összes futó programot láthatjuk, nem csak a sajátjainkat; az `u` megadásával láthatóvá válik a folyamat tulajdonosának a felhasználói neve, valamint a memóriahasználat is; az `x` megmutatja a démon (avagy háttér)folyamatok adatait is és a `ww` hatására pedig a `ps(1)` az összes folyamathoz a teljes parancssort kiírja, még akkor is, ha nem férne ki a képernyőre.

A `top(1)` kimenete is hasonló. Ha elindítjuk, általában ezt láthatjuk:

```

% top
last pid: 72257;  load averages:  0.13,  0.09,  0.03    up 0+13:38:33  22:39:10
47 processes:  1 running, 46 sleeping
CPU states: 12.6% user,  0.0% nice,  7.8% system,  0.0% interrupt, 79.7% idle
Mem: 36M Active, 5256K Inact, 13M Wired, 6312K Cache, 15M Buf, 408K Free
Swap: 256M Total, 38M Used, 217M Free, 15% Inuse

  PID USERNAME PRI NICE  SIZE  RES STATE   TIME  WCPU   CPU COMMAND
72257 nik         28  0  1960K  1044K RUN      0:00 14.86%  1.42% top
  7078 nik         2  0 15280K 10960K select   2:54  0.88%  0.88% xemacs-21.1.14
   281 nik         2  0 18636K  7112K select   5:36  0.73%  0.73% XF86_SVGA
   296 nik         2  0  3240K  1644K select   0:12  0.05%  0.05% xterm
48630 nik         2  0 29816K  9148K select   3:18  0.00%  0.00% navigator-linu
   175 root         2  0   924K   252K select   1:41  0.00%  0.00% syslogd
   7059 nik         2  0   7260K 4644K poll    1:38  0.00%  0.00% mutt
...

```

A kimenet két részre osztható. A fejlécben (vagyis az első öt sorban) látható az utoljára futtatott program azonosítója (PID), a rendszer átlagos terhelése (load average, amellyel mérjük, hogy a rendszerünk mennyire lefoglalt), a rendszer indítása óta eltelt idő (up mint uptime) és a jelenlegi idő. A fejlécben még megtalálhatjuk azt is, mennyi

program fut (esetünkben ez most 47), mennyi memóriát és lapozóállományt használnak, és mennyi időt tölt a rendszer a processzor különböző állapotaiban.

A fejléc alatt a `ps(1)` kimenetéhez hasonló módon oszlopokba rendezve találhatjuk meg a folyamatok adatait: az azonosítóikat, a tulajdonosaik nevét, a felhasznált processzoridőt, a futtatott parancsot. A `top(1)` alapértelmezés szerint mutatja a futó programok által használt memória mennyiségét is: ez további két oszlopra oszlik, ahol az egyikben a teljes memórafoglalást (`SIZE`), a másikban pedig a jelen pillanatban aktívan használt memóriát (`RES`) láthatjuk. A példában látható is, hogy a **Netscape®** (navigator-linu) alkalmazásnak majdnem 30 MB-nyi memóriára van szüksége, de ebből aktívan csak 9 MB-ot használ.

A `top(1)` a kijelzést minden második másodpercben magától frissíti, de ez az `s` kapcsolóval állítható.

3.8. Démonok, jelzések és a futó programok leállítása

Amikor elindítunk egy szövegszerkesztőt, nem sok gondunk akad az irányításával, könnyen utasíthatjuk az állományok betöltésére és így tovább. Mindezt azért tehetjük meg, mert a szövegszerkesztő erre lehetőséget biztosít és mivel a szövegszerkesztő egy *terminál*hoz kapcsolódik. Egyes programok azonban nem úgy lettek kialakítva, hogy állandóan a felhasználó utasításaira támaszkodjanak, ezért az első adandó alkalommal lekapcsolódnak a terminálról. Például egy webszerver egész nap csak webes kéréseket válaszol meg, és általában semmi szüksége nincs a felhasználók utasításaira. A szerverek között leveleket közvetítő programok is ugyanezen osztályba tartoznak.

Ezeket a programokat *démon*oknak hívjuk. A démonok a görög mitológiában jelentek meg: sem a jót, sem pedig a gonoszt nem képviselték, egyszerű apró szellemecskék voltak, akik az emberiség javát szolgálták, pontosan úgy, ahogy ma teszik azt a különféle web- és levelező szerverek. Ezért is ábrázolták sokáig a BSD kabalafiguráját is egy tornacipős, vasvillás vidám démonként.

A démonként futó programok nevéhez a hagyományok szerint hozzá szokták fűzni a “d” betűt. A **BIND** a Berkeley Internet Name Domain (névfeloldó) szolgáltatása, azonban a hozzá tartozó program neve `named`, az **Apache** webszerver programját `httpd`-nek nevezzük, a sornymatató kezelésért felelős démon pedig az `lpd` és így tovább. Ez csupán egy hagyomány, megszokás, nem pedig egy kőbe véssett szabály: például a **Sendmail** levelező démonának neve `sendmail` és nem pedig `maild`.

Néha azért szükségünk lehet arra, hogy felvegyük valahogy a kapcsolatot a démonként futó programokkal is. Ennek egyik lehetséges módja a *jelzések* (signal) küldése (de alapvetően bármilyen futó programnak küldhetünk). Több különféle jelzés küldhető — egyeseknek közülük megkülönböztetett jelentése van, másokat magukat az alkalmazások értelmeznek, amelyről a dokumentációjukban tájékozódhatunk. A `kill(1)` vagy `kill(2)` paranccsal más tulajdonában levő futó programoknak nem tudunk jelzéseket küldeni, ami alól egyedüli kivétel a `root` felhasználó.

Bizonyos esetekben a FreeBSD maga is küld néha jelzéseket. Amikor egy alkalmazást rosszul programoznak le és megpróbál egy számára tiltott memóriaterülethez hozzáférni, a FreeBSD küld neki egy *Segmentation Violation* (`SIGSEGV`, szegmentálási hiba) jelzést. Ha egy alkalmazás az `alarm(3)` rendszerhíváson keresztül kér egy adott idő utáni bekövetkező értesítést, akkor kap erről egy `Alarm` (`SIGALRM`) jelzést és így tovább.

A folyamatok leállítására két jelzés használható: a `SIGTERM` (befejeztetés) és a `SIGKILL` (leállítás). A `SIGTERM` a folyamatok leállításának illedelmes módja, mivel ekkor a futó program képes *elkapni* ezt a jelzést és észrevenni, hogy le akarjuk állítani. Ilyenkor a leállítás előtt lehetősége van szabályosan lezárni a naplót és általánosságban véve befejezni mindent, amit éppen csinál. Előfordulhat azonban, hogy a folyamatok figyelmen kívül hagyják a `SIGTERM` jelzést, ha például éppen egy félbeszakíthatatlan feladat közepén tartanak.

A `SIGKILL` jelzést azonban egyetlen futó program sem hagyhatja figyelmen kívül. Ez lenne a “Nem érdekel, mivel foglalkozol, azonnal hagyj abba!” jelzés. Amikor `SIGKILL` jelzést küldünk egy folyamatnak, a FreeBSD leállítja a folyamatot ott és ahol tart ⁴.

További használható jelzések: `SIGHUP`, `SIGUSR1` és `SIGUSR2`. Ezek általános célú jelzések, amelyeket az alkalmazások eltérő módokon kezelnek.

Tegyük fel, hogy megváltoztattuk a webszerverünk beállításait tartalmazó állományt — valamilyen módon szeretnénk tudatni a szerverrel, hogy olvassa be újra a beállításait. Ezt megtehetjük úgy, hogy leállítjuk és újraindítjuk a `httpd` démonot, de ezzel kiesést okozhatunk a szerver működésében, amit viszont nem engedhetünk meg. A legtöbb démon úgy készítették el, hogy a `SIGHUP` jelzés hatására olvassa be újra a beállításait tartalmazó állományt. Így a `httpd` leállítása és újraindítása helyett egyszerűen elegendő egy `SIGHUP` jelzés küldése. Mivel azonban ez nem szabványosított, a különböző démonok ezt a jelzést többféleképpen is értelmezhetik. Ezért a használata előtt ennek mindenképpen járjunk utána a kérdéses démon dokumentációjában.

A jelzéseket a `kill(1)` paranccsal tudjuk elküldeni, ahogy ezt a következő példában is láthatjuk.

Jelzés küldése egy futó programnak

Ebben a példában megmutatjuk, hogyan lehet jelzést küldeni az `inetd(8)` démonnak. Az `inetd` a beállításait az `/etc/inetd.conf` állományban tárolja, és az `inetd` a `SIGHUP` jelzés hatására képes újraolvasni ezt.

1. Keressük meg annak a folyamatnak az azonosítóját, amelynek a jelzést kívánjuk küldeni. Ezt a `ps(1)` és a `grep(1)` használatával tehetjük meg. A `grep(1)` parancs segítségével más parancsok kimenetében tudunk megkeresni egy általunk megadott szöveget. Ezt a parancsot átlagos felhasználóként futtatjuk, azonban az `inetd(8)` démon a `root` birtokolja, ezért az `ps(1)` használata során meg kell adnunk az `ax` kapcsolókat is.

```
% ps -ax | grep inetd
198  ??  IWs      0:00.00 inetd -wW
```

Innen kiderül, hogy az `inetd(8)` azonosítója 198. Előfordulhat, hogy az eredményben maga a `grep inetd` parancs is megjelenik. Ez a `ps(1)` listázási módszere miatt következhet be.

2. A jelzés elküldésére használjuk a `kill(1)` parancsot. Mivel az `inetd(8)` démon a `root` felhasználó futtatja, ehhez először a `su(1)` parancs kiadásával nekünk is `root` felhasználóvá (rendszeradminisztrátorrá) kell válnunk.

```
% su
Password:
# /bin/kill -s HUP 198
```

Ahogy az a legtöbb UNIX esetén elfogadott, a sikeres végrehajtás esetén a `kill(1)` sem válaszol semmit. Amikor viszont nem egy saját programunknak akarunk jelzést küldeni, akkor a `kill: PID: Operation not permitted` (a művelet nem engedélyezett) hibaüzenetet látunk. Ha véletlenül elgépeltük volna a futó program azonosítóját, akkor a küldendő jelzés nem a megfelelő folyamatnál fog kikötni (ami nem éppen jó), vagy ha szerencsénk van, akkor a jelzést egy éppen használaton kívüli azonosítóra küldtük. Az utóbbi esetben a következő láthatjuk: `kill: PID: No such process` (nincs ilyen folyamat).

Miért `/bin/kill`? A legtöbb parancsértelmező beépítetten tartalmazza a saját `kill` parancsát, tehát ilyenkor közvetlenül maga a parancsértelmező küldi a jelzést, nem pedig a `/bin/kill` programon keresztül. Ez gyakran a javunkra válhat, azonban a küldhető jelzések megadása parancsértelmezőnként eltérhet. Így, ahelyett, hogy egyenként ismernünk kellene mindegyiket, sokkal egyszerűbb közvetlenül a `/bin/kill ...` parancsot használni.

A többi jelzés küldése is nagyon hasonló módon történik, hiszen elegendő csupán a `TERM` vagy a `KILL` behelyettesítése a parancs megfelelő helyére.

Fontos: A rendszerünkben óvatosan bányunk a futó programok leállításával, és legyünk különös tekintettel az 1-es azonosítóval rendelkező, speciális feladattal bíró `init(8)` folyamatra. A `/bin/kill -s KILL 1` parancs kiadásával ugyanis gyorsan le tudjuk állítani a rendszerünket. *Mielőtt* egy `kill(1)` parancsot lezárnánk az **Enter** billentyűvel, *mindig* győződjünk meg róla, hogy valóban tényleg a jó paramétereket adtuk meg.

3.9. Parancsértelmezők

A FreeBSD-ben hétköznapi munkánk legnagyobb részét a parancsértelmezőknek (shell) nevezett parancssoros felületen tudjuk elvégezni. A parancsértelmező fő feladata a beérkező parancsok elfogadása és végrehajtása. Sok parancsértelmező ezenfelül rendelkezik beépített funkciókkal is, amelyek olyan hétköznapi feladatokban igyekeznek segíteni, mint például az állományok kezelése és tömeges elérése reguláris kifejezések használatával, a parancssor szerkesztése, parancsok makrózása és a környezeti változók használata. A FreeBSD alapból tartalmaz néhány parancsértelmezőt, ilyen például az `sh`, a Bourne Shell, és a `tcsh`, a továbbfejlesztett C-shell. Sok más parancsértelmező, mint például a `zsh` és `bash` is elérhető a FreeBSD Portgyűjteményéből.

De melyik parancsértelmezőt is válasszuk? Ez igazából ízlés kérdése. Ha inkább C programozók vagyunk, akkor valószínűleg egy olyan C-szerű shelllel tudunk kényelmesen dolgozni, amilyen például a `tcsh`. Ha viszont egy linuxos rendszert használtunk korábban vagy éppen még soha nem használtunk volna a UNIX parancssorát, érdemes a `bash`-sel megpróbálkoznunk. A lényeg az, hogy minden parancsértelmezőnek vannak olyan egyedi jellemzői, amiért használatosak vagy éppen nem használatosak a munkánkban, ezért magunknak kell kiválasztani a nekünk megfelelőt.

A shellek egyik legáltalánosabb jellemzője az állományok neveinek kiegészítése. Miután begépeljük egy parancs vagy állománynév első néhány karakterét, a **Tab** billentyű lenyomásával megkérhetjük a parancsértelmezőt, hogy magától egészítse ki ("találja ki") a fennmaradó részt. Nézzük erre egy példát. Tegyük fel, hogy van két állományunk, `izemize` és `ize.mize`, és szeretnénk letörölni az `ize.mize` nevűt. Ehhez a következőt kell begépelnünk: `rm iz[Tab].[Tab]`.

Erre a parancsértelmező a következő parancsot írja ki: `rm ize[SIPOLÁS].mize`.

A [SIPOLÁS] itt a konzol sávjára vonatkozik, amellyel jelzi, hogy nem tudta teljesen kiegészíteni az állomány nevét, mivel egynél több is megfelel a megadott alaknak. Az `izemize` és az `ize.mize` is egyaránt az `iz` előtaggal kezdődik, azonban ebből a parancsértelmező csak az `ize` előtagot tudta kikövetkeztetni. Ha most begépelünk még egy `.` karaktert és újra megnyomjuk a **Tab** billentyűt, a parancsértelmező ezúttal képes lesz az állomány teljes nevét megállapítani.

A parancsértelmezők másik általános jellemzője a környezeti változók használata. A környezeti változók lényegében a parancsértelmező környezetéhez tárolt név-érték párok. Ezt a környezetet látja minden olyan program, amit a parancsértelmezőből meghívunk, és ezért tartalmazni is szokott sok ilyen beállítást. Íme a leggyakoribb környezeti változók felsorolása és rövid leírása:

Változó	Leírás
USER	A bejelentkezett felhasználó neve.
PATH	Vesszővel elválasztott könyvtárak, ahol a parancsértelmező a végrehajtható állományokat keresi.
DISPLAY	Az aktuálisan használt X11 megjelenítő hálózati neve, amennyiben létezik ilyen.
SHELL	A használt parancsértelmező.

Változó

TERM

TERMCAP

OSTYPE

MACHTYPE

EDITOR

PAGER

MANPATH

Leírás

A felhasználó által használt terminál típusa. Ebből a terminál képességeit lehet megállapítani.

A terminálok adatbázisából származó, különböző terminálfunkciókhoz tartozó helyettesítő (escape) kódok.

Az operációs rendszer típusa, például FreeBSD.

A rendszer alatt futó gép architektúrája.

A felhasználó által használt szövegszerkesztő.

A felhasználó által lapozásra használt program.

Vesszővel elválasztott könyvtárak, ahol a parancsértelmező a man oldalakat keresi.

A környezeti változók beállítása parancsértelmezőnként valamennyire eltér. Például egy C stílusú parancsértelmező, mint például a `tcsh` vagy a `csh`, a `setenv` paranccsal állítja a környezeti változókat. A Bourne-féle parancsértelmezők, mint például az `sh` vagy a `bash`, az `export` parancsot használják a környezeti változók beállítására. Például a `csh` vagy a `tcsh` használata során a következőképpen tudjuk be- vagy átállítani az `EDITOR` környezeti változó értékét `/usr/local/bin/emacs-re`:

```
% setenv EDITOR /usr/local/bin/emacs
```

Ugyanez a Bourne-féle parancsértelmezőkben:

```
% export EDITOR="/usr/local/bin/emacs"
```

A legtöbb parancsértelmezőben a nevük előtt szerepeltetett `$` jel segítségével kérhetjük a környezeti változók értékének behelyettesítését a parancssorba. Ennek megfelelően az `echo $TERM` parancs kiírja a `TERM` változó aktuális értékét, mivel ebbe a parancsértelmező már az `echo` meghívása előtt behelyettesíti a `TERM` értékét.

A parancsértelmezők számos speciális karaktert, ún. metakaraktert az adatok különleges reprezentációjaként kezelnek. Köztük a leggyakrabban használt a `*`, amely tetszőleges számú karaktert helyettesít egy állomány nevében. Az ilyen metakarakterek segítségével tudunk egyszerre több állományt is megnevezni. Például ha begépeljük az `echo *` parancsot, akkor majdnem ugyanazt kapjuk eredményül, mintha az `ls` parancsot adtuk volna ki, hiszen a parancsértelmező ilyenkor veszi az összes `*` metakarakterre illeszkedő állományt, és a kiíratásukhoz pedig rendre behelyettesíti ezeket a parancssorba az `echo` paramétereként.

Ha nem szeretnénk, hogy a parancsértelmező értelmezze a speciális karaktereket, akkor egy “backslash” (visszaper) (`\`) karaktert eléjük téve mindezt megakadályozhatjuk. Az `echo $TERM` parancs ugyebár kiírja a terminálra vonatkozó környezeti változó beállítását, azonban a `echo \$TERM` változatlanul kiírja a `$TERM` szöveget.

3.9.1. A parancsértelmezőnk megváltoztatása

A parancsértelmezőnk legegyszerűbben a `chsh` parancs használatával változtatható meg. A `chsh` kiadása után elindítja az `EDITOR` környezeti változónak megfelelő szövegszerkesztőt, ha nem lenne ilyen, akkor alapértelmezés szerint a `vi` hívódik meg. Az így megnyitott állományban változtassuk meg kedvünk szerint a “Shell: ” kezdetű sort.

A `chsh` parancsnak megadhatjuk az `-s` opciót is, amin keresztül szövegszerkesztő használata nélkül be tudjuk állítani a parancsértelmezőt. Például ha a parancsértelmezőnket a `bash`-re akarjuk lecserélni, akkor ezt írjuk be:

```
% chsh -s /usr/local/bin/bash
```

Megjegyzés: A használni kívánt parancsértelmezőnek szerepelnie *kell* az `/etc/shells` állományban. Ha a kiválasztott parancsértelmezőt a Portgyűjteményből telepítettük fel, akkor az már minden bizonnyal bekerült oda. Ha viszont saját magunk raktuk volna fel, akkor ide is fel kell vennünk.

Például ha a `bash`-t manuálisan telepítettük és másoltuk a `/usr/local/bin` könyvtárba, akkor így kell eljárunk:

```
# echo "/usr/local/bin/bash" >> /etc/shells
```

Majd próbálkozzunk újra a `chsh` paranccsal.

3.10. Szövegszerkesztők

A FreeBSD beállításának nagy része szöveges állományok szerkesztésével történik. Emiatt sosem árt legalább egy szövegszerkesztőt ismernünk. A FreeBSD alaprendszerében, valamint a Portgyűjteményben is találhatunk néhányat belőlük.

A legegyszerűbben megtanulható és legkönnyedebb szövegszerkesztőt **ee**-nek, avagy “easy editornak” hívják. Az **ee** indításához írjuk be az `ee állománynév` parancsot, ahol az *állománynév* lesz a szerkesztendő állomány neve. Így például az `/etc/rc.conf` állomány szerkesztéséhez gépeljük be az `ee /etc/rc.conf` parancsot. Miután elindult az **ee**, az összes szerkesztéshez használható parancsa megjelenik a képernyő felső részében. Itt a “kalap” (^) karakter a **Ctrl** billentyű lenyomására utal, így tehát a ^e jelölés a **Ctrl+e** billentyűkombinációt jelenti. Ha ki akarunk lépni az **ee**-ből, nyomjuk le az **Esc** billentyűt, majd a felbukkanó menüből válasszuk a szerkesztő elhagyását (leave editor). Ha az állományt módosítottuk, kilépés előtt még a szövegszerkesztő rákérdez, hogy mentse-e a változtatásainkat.

A FreeBSD nagyobb tudású szövegszerkesztőket, mint például a **vi**-t, is tartalmaz az alaprendszer részeként, miközben a többi, mint például az **Emacs** vagy a **vim** a Portgyűjtemény részeként (`editors/emacs` és `editors/vim`) érhető el. Ezek a szerkesztők sokkal több lehetőséget és erőt képviselnek, amiért cserébe viszont valamivel nehezebb megtanulni a használatukat. Ha viszont rengeteg szöveget akarunk majd szerkeszteni, akkor egy **vim** vagy **Emacs** használatának megismerésével sok időt megspórolhatunk.

Számos alkalmazás, amely állományokat akar módosítani vagy szöveges bemenetre van szüksége, automatikusan szövegszerkesztőt nyit meg. Ezt az `EDITOR` környezeti változó beállításával tudjuk meghatározni. Erről részletesebben a parancsértelmezőknél olvashatunk.

3.11. Eszközök és eszközeírók

Az eszköz elnevezést leginkább a rendszerben folyó, hardverrel kapcsolatos tevékenységek kapcsán használják lemezekre, nyomtatókra, grafikus kártyákra és billentyűzetekre. A FreeBSD indulása során többnyire azt láthatjuk, hogy milyen eszközöket sikerült felismernie. Ezeket a rendszerindításkor megjelenő üzeneteket a `/var/run/dmesg.boot` állományban nézhetjük meg újra.

Például az `acd0` az első IDE CD-meghajtót, míg a `kbd0` a billentyűzetet képviseli.

A UNIX operációs rendszerben a legtöbb eszközt a `/dev` könyvtárban található, eszközeíróknak (device node) nevezett speciális állományokon keresztül érhetjük el.

3.11.1. Eszközleírók létrehozása

Amikor egy újfajta eszközt adunk hozzá a rendszerhez vagy csak annak egy új példányát, mindig létre kell hoznunk hozzá egy új eszközleírót.

3.11.1.1. DEVFS (DEVICE File System, Eszköz-állományrendszer)

Az eszközöket tartalmazó állományrendszer, avagy DEVFS, ad hozzáférést a rendszermag által ismert eszközök neveihez a globális állományrendszer nevein keresztül. Így ahelyett, hogy magunknak kellene létrehozniuk és módosítaniuk az eszközleírókat, a DEVFS erre a célra fenntart egy külön állományrendszert.

A devfs(5) man oldalon olvashatunk bővebben erről.

3.12. Bináris formátumok

Annak megértéséhez, hogy a FreeBSD miért az elf(5) formátumot használja, először is tisztában kell lennünk a UNIX típusú rendszerekben használt végrehajtható állományok három “uralkodó” formátumával:

- a.out(5)

A legősibb és egyben a “klasszikus” UNIX-os tárgykódformátum. Egy tömör és rövidke fejlécet használ, aminek az elején a formátum leírására szolgáló “bűvös szám” található (erről bővebben lásd a.out(5)). Három betöltött szegmenst tartalmaz: .text, .data és .bss, valamint egy szimbólumokat és karakterláncokat tároló táblát.

- COFF

Az SVR3 tárgykódformátuma. A fejléc itt már tartalmaz egy table nevű szegmenst is, tehát a .text, .data és .bss szegmensekhez hasonlóan ebből is többet tud tárolni.

- elf(5)

A COFF után következő formátum, amelyben több szegmens is megtalálható, valamint létezik 32 bites és 64 bites változatban is. Egyetlen hátránya van: az ELF tervezése során rendszerarchitektúránként csupán egyetlen ABI-t (bináris alkalmazói felületet) feltételeztek. Ez azonban meglehetősen helytelen, mivel még a kereskedelmi SYSV világában (ahol már legalább három ABI található: SVR4, Solaris és SCO) sem állja meg a helyét.

A FreeBSD ezt a problémát a *megbélyegzés* (branding) segítségével próbálja megoldani, aminek révén el tudunk látni egy ismert ELF állományt a futtatásához megfelelő ABI-ra vonatkozó információkkal. Erről részletesebben a brandelf(1) oldalán tájékozódhatunk.

A FreeBSD a “klasszikusok” táborából indult, ezért kezdetben az a.out(5) formátumot használta, mivel ez a technológia a BSD kiadások számos generációjában megmérettetett és bevált, egészen a 3.X ág elindulásáig. Habár már jóval előtte lehetett fordítani és futtatni natív ELF binárisokat (és rendszermagokat) a FreeBSD rendszereken, a FreeBSD kezdetben ódzkodott váltani az alapértelmezés szerinti ELF formátumra. De vajon miért? Nos, amikor a Linux-tábor megtette a maga fájdalmas váltását az ELF-re, az nem annyira azért volt, hogy megszabaduljanak az a.out végrehajtható formátumtól, hanem mert a rugalmatlan, ugrótáblákon alapuló osztottkönyvtár-kezelési mechanizmusaik nagyon megnehezítették a gyártók és fejlesztők számára az osztott függvénykönyvtárak létrehozását. Mivel az ELF formátumhoz rendelkezésre álló eszközök megoldást kínáltak az osztott könyvtárak gondjaira, és mivel általánosan elfogadták “a jövőbe vezető útként”, a FreeBSD is felvállalta az átállással kapcsolatos költségeket és végrehajtotta azt. A FreeBSD az osztott könyvtárakat leginkább a Sun SunOS™ rendszeréhez hasonlóan kezeli, ami egy nagyon könnyen használható megoldás.

De miért van ilyen sok különböző formátum?

A ködös és sötét múltban egyszerűbb hardverek voltak. Ezek az egyszerű hardverek egyszerű, kicsi rendszereket támogattak. Az `a.out` tökéletesen megfelelő volt egy ilyen egyszerű rendszer (egy PDP-11) binárisainak tárolására. Ahogy az emberek nekiláttak átültetni erről az egyszerű rendszerről a UNIX-ot más rendszerekre, az `a.out` formátumot továbbra is megtartották, mivel a UNIX kezdeti, Motorola 68k-ra, VAXenre készített átíratához is elegendő volt.

Ezután néhány éles elméjű hardvermérnök kitalálta, ha rá tudnák kényszeríteni a programokat egy-két ügyetlen trükkre, akkor a terveken meg tudnának spórolni néhány logikai kaput és ezzel a processzor is gyorsabban tudna futni. Miközben az `a.out` formátumot ilyen hardverre (amit manapság RISC-nek hívnak) is szeretnék volna áthozni, kiderült, hogy ebben az esetben szinte használhatatlan. Ezért az `a.out` formátum által felkínálnál nagyobb teljesítmény elérése érdekében nekiláttak számos más formátumot is kidolgozni. Ekkor jöttek létre a COFF, ECOFF és más hasonló formátumok, amelyek előbb-utóbb korlátokba ütköztek, még mielőtt a történelem megállapodott volna az ELF formátumnál.

Ráadásul a programok méretei egyre inkább kezdtek nőni, miközben a lemezek (valamint a fizikai memória) továbbra is viszonylag kicsik maradtak, ezért megszületett az osztott könyvtár ötlete, és a virtuális memóriát kezelő alrendszer is sokat finomodott. Mivel ezek a különböző fejlesztések az `a.out` formátumra épültek, annak használhatósága a beletömött módosítások számával együtt romlott. Emellett az emberek még szerettek volna betölteni különféle dolgokat futási időben dinamikusan, vagy éppen a memória és a lapozóállomány megspórolásához kipucolni a programjaik egyes részeit az inicializáló kódrészletek lefutása után. A programozási nyelvek is fejlődtek, és az emberek a főprogram futása előtt is akartak kódot futtatni. Az `a.out` formátum rengeteg apró foltozáson esett keresztül, amelyek egy ideig még tudták is tartani magukat. Azonban egy idő után már az `a.out` formátum egyre növekvő teljesítménycsökkenés nélkül már nem volt képes állni a sarat. Habár az ELF megszüntette a fennálló problémák jelentős részét, egyúttal megnehezítette egy alapvetően működő rendszer leváltását. Ezért az ELF formátumnak meg kellett várnia azt a pillanatot, amikor az `a.out` használata már kényelmetlenné vált.

Azonban ahogy múlt az idő, az eszközökből, amelyekből a FreeBSD a fordításához szükséges eszközöket származtatta (különösen az assembler és a betöltő), létrejött két párhuzamos fejlesztési fa. A FreeBSD-fa kiegészült az osztott könyvtárak támogatásával és hibákat javított, miközben a GNU-fa alkotói, akik eredetileg készítették ezeket a programokat, újraírták az eszközeiket és a keresztfordításhoz egyszerűbb támogatást készítettek, cserélhetővé tették a különböző formátumokat és így tovább. Sokan akartak FreeBSD-re keresztfordítani, azonban nem volt szerencsájük, mert a FreeBSD régebbi forrásait az **as** és **ld** már nem emésztette meg. Az új GNU eszköztár (a **binutils**) viszont ismeri már a keresztfordítást, az ELF formátumot, az osztott könyvtárakat, a C++ kiterjesztéseit stb. Időközben egyre több gyártó ELF formátumú binárisokat adott ki, és jó érzés volt ezeket FreeBSD-n is futtatni.

Az ELF sokkal kifejezőbb az `a.out` formátumnál, és jóval több bővítési lehetőséget enged az alaprendszerben. Az ELF formátumhoz tartozó eszközöket jobban karbantartják és támogatja a keresztfordítást, ami viszont sokaknak fontos. Az ELF talán némileg lassabb, mint az `a.out`, azonban ez nehezen mérhető le. Számos részletben eltérnek ugyan, például hogyan képeznek le lapokat, hogyan kezelik az inicializáló kódot stb., de ezek egyike sem igazán fontos. Idővel az `a.out` támogatása ki fog kerülni a **GENERIC** rendszermagból, és végül majd teljesen eltávolításra kerül, ahogy a régi `a.out` formátumú programok szépen lassan kifutnak.

3.13. Bővebben olvashatunk...

3.13.1. Man oldalak

A FreeBSD legátfogóbb dokumentációja a benne található man oldalak összessége. A rendszerben található szinte majdnem mindegyik programhoz létezik egy rövid használati útmutató, amely bemutatja az adott program alapvető működését és a különböző beállításait. Ezek a leírások a `man` parancs segítségével jeleníthetők meg. A `man` parancs használata egyszerű:

```
% man parancs
```

ahol a `parancs` a megismerni kívánt parancsra utal. Például ha az `ls` parancsról szeretnénk többet megtudni, írjuk be:

```
% man ls
```

Az elérhető használati útmutatókat a következő számozott szakaszokra osztották:

1. Felhasználói parancsok
2. Rendszerhívások és hibakódok
3. A C függvénykönyvtár függvényei
4. Eszközmeghajtók
5. Állományformátumok
6. Játékok és egyéb szórakoztató alkalmazások
7. Egyéb információk
8. Rendszerkarbantartási és -működtetési parancsok
9. Rendszermagfejlesztők számára

Bizonyos esetekben ugyanaz a téma az útmutatók több szakaszában is elérhető. Például létezik `chmod` felhasználói parancs és a `chmod()` rendszerhívás. Ilyenkor a `man` parancsnak meg tudjuk adni pontosan, melyik szakaszra is vagyunk kíváncsiak:

```
% man 1 chmod
```

Ennek hatására a `chmod` felhasználói parancshoz tartozó oldal jelenik meg. Írott formában a használati útmutatók különböző szakaszaira hagyományosan a név után zárójelbe tett számmal hivatkoznak, így a `chmod(1)` a `chmod` felhasználói parancs és a `chmod(2)` a rendszerhívás.

Ez a módszer remekül működik abban az esetben, amikor ismerjük a parancs nevét, azonban mit tegyünk akkor, ha nem is emlékszünk a nevére? A `man` parancs a `-k` segítségével paraméterezhető úgy is, hogy a parancsok leírásai között keressen valamilyen kulcsszó mentén:

```
% man -k mail
```

Ezzel a paranccsal megkapjuk azon parancsok listáját, amelyek leírásában szerepel a “mail” kulcsszó. Ez egyébként működésében teljesen megegyezik a `apropos` paranccsal.

Szóval szeretnénk megtudni, hogy a `/usr/bin` könyvtárban levő parancsok pontosan mit is csinálnak? Ehhez írjuk be:

```
% cd /usr/bin
% man -f *
```

vagy

```
% cd /usr/bin
% whatis *
```

ami ugyanezt teszi.

3.13.2. A GNU info állományok

A FreeBSD-ben megtalálható a Szabad Szoftver Alapítvány (Free Software Foundation, FSF) által készített számos alkalmazás. Ezek a programok a szokványos man oldalakon kívül még általában tartalmaznak egy `info`-nak nevezett, sokkal részletesebb hipertext alapú leírást is, amelyeket az `info` paranccsal, vagy ha van fenn **emacs**, akkor annak az `info` módjában tudjuk megjeleníteni.

Az `info(1)` parancs használatához ennyit kell beírnunk:

```
% info
```

Itt a `h` lenyomásával kapunk egy rövid bemutatkozást. A parancsok rövid listáját a `?` billentyű hozza elő.

Megjegyzések

1. Erre utal pontosan az `i386` jelzés. Még abban az esetben is az `i386` kiírást fogjuk látni, ha a FreeBSD-t konkrétan nem is az Intel 386-os processzorán futtatjuk. Itt ugyanis nem a processzorunk típusát, hanem annak "architektúráját" láthatjuk.
2. A rendszerindító szkriptek olyan programok, amelyek a FreeBSD indulása során maguktól lefutnak. Legfontosabb feladatuk elvégezni a többi program futtatásához szükséges beállításokat, valamint elindítani a háttérben futtatandó, hasznos munkát végző szolgáltatásokat.
3. A FreeBSD konzol- és billentyűzetmeghajtóinak teljes, pusztán műszaki és precíz leírása a `syscons(4)`, `atkbd(4)`, `vidcontrol(1)` és `kbdcontrol(1)` man oldalakon olvasható. Itt most nem bocsátkozunk részletekbe, azonban a téma iránt érdeklődő olvasóknak mindig érdemes fellapozniuk a kapcsolódó man oldalakat, ahol megtalálhatják az említett eszközök részletesebb és bővebb leírását.
4. Ez azért nem teljesen igaz. Van néhány olyan tevékenység, ami nem szakítható meg. Ilyen például az, amikor a program egy másik számítógépen található állományt próbál olvasni, miközben valamilyen ok (kikapcsolás, hálózati hiba) folytán elveszti vele a kapcsolatot. Ekkor a program futása "megszakíthatatlan". Majd amikor a program feladja a próbálkozást (általában két perc után), akkor következik be a tényleges leállítása.

Fejezet 4. Alkalmazások telepítése: csomagok és portok

4.1. Áttekintés

A FreeBSD rendszereszközök gazdag gyűjteményével érkezik az alaprendszer részeként. Azonban a külső alkalmazások telepítéséhez rengeteg teendőt kell elvégeznünk. A feladat elvégzésére ezért a FreeBSD két, egymást kiegészítő technológiát kínál fel: a FreeBSD Portgyűjteményt (telepítés forráskódból) és a csomagokat (telepítés előre elkészített bináris csomagokból). Mind a két módszerrel fel tudjuk telepíteni a kedvenc alkalmazásunk legújabb verzióját lokálisan vagy egyenesen a hálózatról.

A fejezet elolvasása során megismerjük:

- hogyan telepítsünk külső fejlesztésű bináris szoftvercsomagokat;
- hogyan fordítsunk le a forrásukból külső fejlesztésű szoftvereket a Portgyűjtemény segítségével;
- hogyan távolítsunk el korábban már telepített csomagokat és portokat;
- hogyan bíráljuk felül a Portgyűjtemény által használt alapértelmezett értékeket;
- hogyan keressük meg a megfelelő szoftvercsomagokat;
- hogyan frissítsük a telepített alkalmazásokat.

4.2. Az alkalmazások telepítésének összefoglalása

Ha korábban már használtunk UNIX rendszereket, valószínűleg ismerjük a külső alkalmazások telepítésének jellemező menetét:

1. Töltsük le a szoftvert, amelyet vagy forráskód vagy pedig bináris formátumban érhetünk el.
2. Bontsuk ki az alkalmazás letöltött változatát (ez általában a `compress(1)`, `gzip(1)` vagy a `bzip2(1)` által tömörített tar állomány).
3. Keressük meg a dokumentációt (többnyire az `INSTALL` vagy a `README` állományban található, vagy a `doc/` alkönyvtárban) és olvassuk el benne, hogyan tudjuk telepíteni a szoftvert.
4. Ha a szoftver forrását töltöttük le, fordítsuk le. Elképzelhető, hogy ennek során szerkesztenünk kell a `Makefile` állományt vagy lefuttatnunk a `configure` szkriptet, illetve más lépéseket is el kell végeznünk.
5. Próbáljuk a ki szoftvert, majd telepítsük.

Ez annak a forgatókönyve, amikor minden hiba nélkül lezajlik. Megeshet azonban, ha olyan szoftvert telepítünk, amelyet nem kifejezetten a FreeBSD-hez terveztek, akkor javítanunk kell a forráskódban a szoftver megfelelő működéséhez.

Ha sikerül működésre bírni, folytathatjuk FreeBSD-n a szoftver telepítését a “megszokott” módon. Habár a FreeBSD erre a célra két lehetőséget is felkínál, amivel rengeteg erőfeszítéstől megkímélhet minket: ezek a csomagok és a portok. Az írás pillanatában közel 20 000 külső alkalmazás érhető el ilyen formában.

Egy adott alkalmazás esetén a hozzá tartozó FreeBSD-s csomag mindössze egyetlen letöltendő állományt takar. A csomag tartalmazza az alkalmazás telepítéséhez szükséges összes parancs előre lefordított változatát, ugyanígy magát a dokumentációt is. A letöltött csomagokat a FreeBSD csomagkezelő parancsaival vehetjük használatba: ezek a `pkg_add(1)`, `pkg_delete(1)`, `pkg_info(1)` és így tovább. Az új alkalmazások telepítése ennek köszönhetően egyetlen paranccsal elvégezhető.

Egy alkalmazás FreeBSD-s portja mögött lényegében állományok gyűjteménye áll, amelyek abban segítenek, hogy automatikusan tudjunk telepíteni a forráskód felhasználásával.

Ne felejtjük el, hogy normális esetben számos lépcsőt végig kell járnunk egy program sajátkezű lefordításához (letöltés, kitömörítés, javíthatás, fordítás, telepítés). A portot alkotó állományok tartalmazzák az összes olyan szükséges információt, amelyek átengedik ezt a feladatot a rendszernek. Kiadunk néhány egyszerű parancsot és az alkalmazás magától letöltődik, kitömörítődik, módosítja a forráskódját, lefordul és települ.

Valójában a portrendszer használható olyan csomagok létrehozására is, amelyeket később a `pkg_add` és többi hozzá hasonló, hamarosan részletesebben is bemutatandó csomagkezelő paranccsal is kezelni tudunk.

A csomagok és a portok egyaránt képesek *függőségeket* kezelni. Tegyük fel, hogy egy olyan alkalmazást akarunk telepíteni, amely egy adott függvénykönyvtár meglététől függ a rendszeren. Az alkalmazás és a könyvtár is elérhető FreeBSD portként és csomagként. Akár a `pkg_add` parancsot, akár a portrendszert használjuk az alkalmazás hozzáadására, mind a kettő észre fogja venni, hogy a szükséges könyvtárt még nem telepítettük, ezért először azt fogja automatikusan telepíteni.

Tudván, hogy a két említett megoldás szinte teljesen egyenértékű, felmerülhet a kérdés: a FreeBSD mégis miért rendelkezik mindkettővel? A csomagoknak és a portoknak is megvannak a maguk előnyei, és hogy a kettő közül melyiket használjuk, csak az egyéni ízlésünkön múlik.

A csomagok használatának előnyei

- Egy csomag általában kisebb, mint az alkalmazás forráskódját tartalmazó tömörített tar állomány.
- A csomagokat nem kell fordítani. Nagyobb alkalmazások, mint például a **Mozilla**, **KDE** vagy **GNOME** esetén ez kulcsfontosságú lehet, főleg abban az esetben, ha a rendszerünk ehhez nem eléggé gyors.
- A csomagok használata nem várja el tőlünk, hogy behatóbban ismerjük, miként is kell FreeBSD-n szoftvereket lefordítani.

A portok használatának előnyei

- A csomagokat általános esetben igen óvatos beállításokkal készítik el, hiszen a lehető legtöbb rendszeren működőképesnek kell lenniük. Ha viszont portból telepítünk, nyugodtan hangolhatjuk úgy a beállításokat, hogy (például) a Pentium 4 vagy az Athlon processzoroknak kedvező kódot hozzanak létre.
- Bizonyos alkalmazások fordítás idején állítandó beállításokkal rendelkeznek arról, hogy mire lesznek képesek és mire nem. Például az **Apache** beépített konfigurációs opciók széles kelléktárával rendelkezik. Amikor viszont portból hozzuk létre, nem kell elfogadnunk ezek alapértelmezett értékeit, hanem a saját igényeinknek megfelelően átállíthatjuk ezeket.

Egyes esetekben több különféle beállítást tükröző csomag is létezhet ugyanahhoz az alkalmazáshoz. Például a **Ghostscript** elérhető `ghostscript` és `ghostscript-nox11` csomagként is attól függően, hogy telepítettük-e az X11 szerveret. Ez természetesen egy meglehetősen durva kijátszása a csomagrendszernek, és gyorsan lehetetlenné is válik a használata, ha az adott alkalmazás egy-két fordítási idejű beállításnál többel rendelkezik.

- Néhány szoftver licencelése tiltja a bináris terjesztést. Ezért ezek a szoftverek kizárólag csak forráskód formájában továbbíthatóak.

- Néhányan nem bíznak meg a bináris verziókban. Ha látjuk a forráskódot is, akkor (elméletben) át tudjuk nézni, és mi magunk is megkereshetjük a benne lappangó hibákat.
- Ha vannak saját javításaink, csak a forráskód birtokában tudjuk ezeket felhasználni.
- Sokan szeretik, ha egyszerűen csak “ott van” a szoftverek forráskódja. Ha éppen unatkoznak, beléjük tudnak nézni, ötleteket és kódot tudnak belőlük meríteni (persze csak akkor, ha ezt a licenc megengedi), vagy tovább tudják ezeket fejleszteni, orvosolni tudják a hibákat stb.

A portok frissítéséről a FreeBSD ports levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports>) és a FreeBSD ports bugs levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs>) valamelyikéről szerezhetünk naprakész információkat.

FigyelemMielőtt bármelyik alkalmazást is telepítenénk, érdemes meglátogatnunk az <http://vuxml.freebsd.org> oldalt, ahol a hozzá tartozó ismert biztonsági problémákról olvashatunk.

Telepíthetjük a `ports-mgmt/portaudit` programot is, amely automatikusan ellenőrzi a telepített alkalmazások ismert sebezhetőségeit. Ez az ellenőrzés egyébként megejthető minden port lefordítása előtt is. Ezalatt a `portaudit -F -a` parancs kiadásával ellenőrizhetjük utólag a telepített csomagokat.

A fejezet fennmaradó részében megmutatjuk, hogyan használjuk FreeBSD-ben a csomagokat és portokat külső alkalmazások telepítésére és karbantartására.

4.3. A számunkra szükséges alkalmazások felkutatása

Mielőtt telepítenénk bármilyen alkalmazást, tudnunk kell, hogyan is nevezik.

A FreeBSD-hez elérhető alkalmazások listája folyamatosan növekszik. Szerencsére számos módja van annak, hogy utánajárjunk a keresett szoftvernek:

- A FreeBSD honlapján található egy rendszeresen frissülő listát az összes elérhető alkalmazásról, a <http://www.FreeBSD.org/ports/> (<http://www.FreeBSD.org/ports/index.html>) címen. Itt a portok különböző kategóriákba sorolva találhatóak meg, ahol név szerint megkereshetjük az alkalmazást (amennyiben ismerjük), vagy végigbongéshatjuk az adott kategóriában elérhető alkalmazásokat is.

•

Dan Langille a <http://www.FreshPorts.org/> címen karbantartja a FreshPorts nevű oldalt. Ezen az oldalon folyamatosan nyomon lehet követni a Portgyűjteményben megtalálható alkalmazások változásait, lehetővé téve, hogy egy vagy több portot is “figyeljünk”, vagy e-mailt küldjünk a frissítésükről.

•

Amennyiben nem ismerjük a keresett alkalmazás nevét, próbáljuk meg felkutatni a FreshMeaten (<http://www.freshmeat.net/>) vagy hozzá hasonló oldalakon, majd nézzük meg a FreeBSD honlapján, hogy az adott alkalmazást portolták-e már a rendszerre.

- Ha pontosan ismerjük a port nevét, és csak a kategóriáját kellene megkeresnünk, használjuk a `whereis(1)` parancsot. Egyszerűen csak adjuk ki a `whereis név` parancsot, ahol a *név* a telepítendő program neve. Ha sikerült megtalálni, részletes információt kapunk arról, hogy hol található, valahogy így:

```
# whereis lsof
lsof: /usr/ports/sysutils/lsof
```

A fenti példában megtudhatjuk, hogy az `lsof` parancs a `/usr/ports/sysutils/lsof` könyvtárban található.

- Vagy egy egyszerű `echo(1)` parancssal is megkereshetjük a portfában a portokat. Mint például:

```
# echo /usr/ports/*/*lsof*
/usr/ports/sysutils/lsof
```

Ez a módszer a `/usr/ports/distfiles` könyvtárba letöltött összes illeszkedő állományt is kilistázza.

- Egy másik lehetőség egy adott port megtalálására, ha a Portgyűjtemény beépített keresési mechanizmusát használjuk. Ennek használatához a `/usr/ports` könyvtárban kell lennünk. Miután beléptünk ide, futtassuk le a `make search name=programnév` parancsot, ahol a *programnév* a keresendő program neve. Például, ha az `lsof` programot keressük:

```
# cd /usr/ports
# make search name=lsof
Port:    lsof-4.56.4
Path:    /usr/ports/sysutils/lsof
Info:    Lists information about open files (similar to fstat(1))
Maint:   obrien@FreeBSD.org
Index:   sysutils
B-deps:
R-deps:
```

A keresés eredményében leginkább a “Path:” kezdetű sorra kell odafigyelnünk, mivel ez árulja el, hol is található meg a portot. Az itt szereplő többi információ nem szükséges a port telepítéséhez, ezért azokkal itt most nem foglalkozunk.

Mélyebb keresésekhez használhatjuk a `make search key=szöveg` parancsot is, ahol a *szöveg* a keresendő szöveg(részlet) lesz. Ezt a rendszer keresni fogja a portok neveiben, megjegyzésekben, leírásokban és függőségekben. Amikor nem ismerjük a keresett program nevét, ez olyan portok keresésére alkalmas, amelyek egy adott témához kapcsolódnak.

A fenti esetek mindegyikében a keresés nem különbözteti meg a kis- és nagybetűket. Tehát az “LSOF” keresése ugyanazt az eredményt adja, mint az “lsof” esetén.

4.4. A csomagrendszer használata

Írta: Chern Lee.

FreeBSD alatt több különböző módon tudunk csomagokat használni:

- A **sysinstall** használatán keresztül a futó rendszeren tudjuk megnézni a telepített csomagokat, tudunk vele csomagokat telepíteni vagy törölni. Ezzel részletesebben a 2.10.11 Szakasz foglalkozik.
- A szakasz további részében ismertetett egyéb parancssoros csomagkezelő segédprogramok.

4.4.1. Csomagok telepítése

A `pkg_add(1)` segédprogram segítségével telepíthetünk FreeBSD-hez készült szoftvercsomagokat lokálisan vagy a hálózaton levő egyik szerveren megtalálható állományokból:

Példa 4-1. Csomagok letöltése manuálisan és telepítése lokálisan

```
# ftp -a ftp2.FreeBSD.org
Connected to ftp2.FreeBSD.org.
220 ftp2.FreeBSD.org FTP server (Version 6.00LS) ready.
331 Guest login ok, send your email address as password.
230-
230-      This machine is in Vienna, VA, USA, hosted by Verio.
230-      Questions? E-mail freebsd@vienna.verio.net.
230-
230-
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /pub/FreeBSD/ports/packages/sysutils/
250 CWD command successful.
ftp> get lsof-4.56.4.tgz
local: lsof-4.56.4.tgz remote: lsof-4.56.4.tgz
200 PORT command successful.
150 Opening BINARY mode data connection for 'lsof-4.56.4.tgz' (92375 bytes).
100% |*****| 92375      00:00 ETA
226 Transfer complete.
92375 bytes received in 5.60 seconds (16.11 KB/s)
ftp> exit
# pkg_add lsof-4.56.4.tgz
```

Ha nincsenek egyáltalán helyben csomagjaink (például egy FreeBSD CD-készletben), akkor a legjobban úgy járunk, ha használjuk a `pkg_add(1)` `-r` kapcsolóját. Ennek hatására a segédprogram önmagától meghatározza a szükséges állományformátumot és verziót, majd FTP-n keresztül letölti és telepíti a csomagot.

```
# pkg_add -r lsof
```

Az iménti példában a program mindenféle további beavatkozás nélkül letölti a megfelelő csomagot és felteszi. Ha a központi helyett egy másik szervert szeretnénk használni, felül kell bírálunk az alapértelmezett beállításokat és igényeinknek megfelelően be kell állítanunk a `PACKAGESITE` környezeti változó értékét. A `pkg_add(1)` a `fetch(3)` programot használja az állományok letöltésére, amely pedig számos egyéb környezeti változót is figyel, mint például az `FTP_PASSIVE_MODE`, az `FTP_PROXY` és az `FTP_PASSWORD`. Ha tűzfal mögött vagyunk, ezek közül néhányat biztosan be kell majd állítanunk, vagy FTP/HTTP proxyt kell használnunk. A `fetch(3)` man oldalán megtaláljuk ezen változók teljes felsorolását. Figyeljük meg, hogy az `lsof-4.56.4` helyett csak `lsof`-ot adtunk meg. Amikor ugyanis kérjük a csomag letöltését is, nem szabad verziószámot megadnunk. A `pkg_add(1)` mindig az alkalmazás legfrissebb verzióját fogja letölteni.

Megjegyzés: Ha a FreeBSD-CURRENT vagy FreeBSD-STABLE verziókat használjuk, a `pkg_add(1)` mindig az alkalmazás elérhető legfrissebb verzióját fogja letölteni. Ha azonban valamelyik -RELEASE verziót használjuk, a csomagnak az adott kiadáshoz készült verzióját fogja leszedni. Ezt a működési módot a `PACKAGESITE` változó felülírásával viszont meg tudjuk változtatni. Például ha a FreeBSD 5.4-RELEASE változatával dolgozunk, a `pkg_add(1)` alapértelmezés szerint a

```
ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-5.4-release/Latest/
```

címről fogja letölteni a csomagokat. Ha mi viszont a FreeBSD 5-STABLE csomagok letöltését akarjuk elérni, állítsuk az `PACKAGESITE` értékét a `ftp://ftp.freebsd.org/pub/FreeBSD/i386/packages-5-stable/Latest/` címre.

A csomagok `.tgz` és `.tbz` formátumokban kerülnek terjesztésre. Ezek az

`ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/` címen, vagy pedig a FreeBSD CD-ken találhatóak meg. A 4 CD-ből álló készlet (illetve a PowerPak stb.) minden CD-jén található csomagokat a `packages/` könyvtárban. A csomagokat tároló könyvtár struktúrája hasonló a `/usr/ports` könyvtárban kialakított könyvtárfához. Minden kategóriának saját könyvtára van, és minden csomag megtalálható az `All` (összes) kategóriában.

A csomagrendszer könyvtárszerkezete tehát megegyezik a portok szétosztásával, ezáltal így képesek egymással összedolgozni a teljes csomag/port rendszer megformálásában.

4.4.2. A csomagok kezelése

A `pkg_info(1)` egy olyan segédprogram, amellyel készíteni lehet egy listát a telepített csomagokról, és emellett még más egyéb információkat tudhatunk meg róluk.

```
# pkg_info
cvsup-16.1          A general network file distribution system optimized for CV
docbook-1.2        Meta-port for the different versions of the DocBook DTD
...
```

A `pkg_version(1)` összefoglalja az összes telepített csomag verzióját. Ezenkívül össze is hasonlítja a csomagok verzióját a portfában található aktuális verziókéval.

```
# pkg_version
cvsup                =
docbook              =
...
```

A második oszlopban látható jelek utalnak a telepített verzió a helyi portfában található verzióéhoz viszonyított korára.

Jel	Jelentés
=	A telepített csomag verziója megegyezik a helyi portfában található verziójával.
<	A telepített verzió a portfában levőnél régebbi.
>	A telepített verzió újabb, mint a portfában található. (A helyi portfa valószínűleg nem lett frissítve.)
?	A telepített csomag nem található a portok között. (Ez akkor történhet meg, amikor például egy portot eltávolítottak a Portgyűjteményből vagy átnevezték.)
*	A csomagnak több verziója is jelen van.
!	A telepített csomag szerepel az indexben, de a <code>pkg_version</code> valamiért nem volt képes összehasonlítani a verziószámát az indexben levő bejegyzéssel.

4.4.3. Csomagok törlése

Egy korábban már telepített csomag eltávolításához használjuk a `pkg_delete(1)` segédprogramot.

```
# pkg_delete xchat-1.7.1
```

A `pkg_delete(1)` használatánál szükség van a csomag teljes nevének és verziószámának megadására. A fenti parancs tehát nem működik, ha csak az `xchat`-et adjuk meg az `xchat-1.7.1` helyett. A telepített csomag verzióját azonban könnyedén kitalálhatjuk a `pkg_version(1)` alkalmazásával. Esetleg egyszerűen dzsókerkaraktereket is használhatunk:

```
# pkg_delete xchat\*
```

Ebben az esetben az összes `xchat`-tel kezdődő csomagot letörli.

4.4.4. Egyebek

A csomagokra vonatkozó összes információ a `/var/db/pkg` könyvtárban található. Az egyes csomagok leírása és hozzájuk telepített állományok listája az ezen a könyvtáron belül elhelyezkedő állományokban tárolódik.

4.5. A Portgyűjtemény használata

A most következő szakaszokban megismerhetjük azokat az alapvető utasításokat, amelyekkel a Portgyűjteményen keresztül tudunk programokat telepíteni és eltávolítani. Az ehhez használható `make` targetek és környezeti változók részletesebb leírását a `ports(7)` man oldalán lelhetjük meg.

4.5.1. A Portgyűjtemény beszerzése

Mielőtt bármelyik portot is tudnánk telepíteni, elsőként magát a Portgyűjteményt kell megszereznünk — ez lényegében a `/usr/ports` könyvtárban megtalálható `Makefile` állományok, javítások és leírások gyűjteménye.

A FreeBSD telepítése közben a **sysinstall** rákérdez a Portgyűjtemény telepítésére is. Ha erre nemet válaszoltunk volna, a portok gyűjteményét az alábbi módokon szerezhetjük be:

A CVSup használatával

A **CVSup** protokoll használatával viszonylag gyorsan el tudjuk érni és naprakészen tudjuk tartani a Portgyűjtemény egy példányát. A **CVSup** használatát alaposabban a **A CVSup használata** című függelékben ismerhetjük meg.

Megjegyzés: A FreeBSD 6.2 változatától kezdve az alaprendszerben a **CVSup** protokollt a **csup** valósítja meg. A FreeBSD korábbi változatának használói ezt a programot a `net/csup` porton vagy csomagon keresztül tudják telepíteni.

Gondoskodjunk róla, hogy a `/usr/ports` üres legyen a **csup** első futtatása előtt! Ha más forrásból raktuk ide a Portgyűjteményt, a **csup** nem fogja lenyesegetni az azóta eltávolított javításokat.

1. Futtassuk a `csup` programot:

```
# csup -L 2 -h cvsup.FreeBSD.org /usr/share/examples/cvsup/ports-supfile
```

Itt írjuk át a `cvsup.FreeBSD.org` címét a hozzánk legközelebb levő **CVSup** szerver címére. Az összes elérhető tükrőszerver címét a CVSup tükrözések (A.6.7 Szakasz) című részben olvashatjuk.

Megjegyzés: Ha például el akarjuk kerülni a **CVSup** szerver megadását a parancssorban, akkor mindenképpen a `ports-supfile` állományból érdemes készíteni egy saját verziót.

1. Ebben az esetben `root` felhasználóként másoljuk a `/usr/share/examples/cvsup/ports-supfile` állományt egy új helyre, például a `/root` könyvtárba vagy a saját felhasználói könyvtárunkba.
2. Szerkesszük át a `ports-supfile` állományt.
3. Írjuk át a `CHANGE_THIS.FreeBSD.org` értéket a hozzánk legközelebb található **CVSup** szerverére. A CVSup tükrözések (A.6.7 Szakasz) című részben megtaláljuk az összes ilyen tükrőszervert.
4. És most indítsuk el a `csup` parancsot az alábbi módon:

```
# csup -L 2 /root/ports-supfile
```

2. A `csup(1)` parancs későbbi futása során már letölti és érvényesíti az észlelt változtatásokat a saját Portgyűjteményünkben, de a telepített portokat nem fogja újrafordítani.

A Portsnap használatával

A **Portsnap** egy másik módszert képvisel a Portgyűjtemény terjesztésére, a lehetőségeinek részletesebb megismeréséhez tekintsük át A Portsnap használata című szakaszt.

1. Töltjük le a Portgyűjtemény tömörített pillanatképét a `/var/db/portsnap` könyvtárba. Ha akarjuk, ezután a lépés után már lekapcsolódhatunk az internetről.

```
# portsnap fetch
```

2. Ha még csak először futtatjuk a **Portsnapet**, bontsuk ki az imént letöltött állapotot a `/usr/ports` könyvtárba:

```
# portsnap extract
```

Ha viszont már korábban is létezett a `/usr/ports` könyvtárunk és most csak frissítjük, akkor helyette ezt a parancsot adjuk ki:

```
# portsnap update
```

A sysinstall használatával

Ebben az esetben a **sysinstall** nevű programmal telepítjük a Portgyűjteményt valamilyen telepítőeszköztől. Ilyenkor azonban a kiadás dátumának megfelelő, valószínűleg régebbi változat kerül fel. Ha rendelkezünk internet-hozzáféréssel, akkor inkább az előbb tárgyalt módszerek valamelyikét alkalmazzuk.

1. `root` felhasználóként adjuk ki a `sysinstall` parancsot, ahogy itt is láthatjuk:

```
# sysinstall
```

2. Menjünk le és álljunk meg a **Configure** (Beállítások) menüpontnál, és nyomjunk **Enter** billentyűt.
3. Menjünk le és keressük meg a **Distributions** (Terjesztések) menüponot, majd nyomjunk meg az **Enter** billentyűt.
4. Menjünk le, válasszuk ki a `ports` elemet a **Szókőz** megnyomásával.
5. Menjünk fel az **Exit** (Kilépés) ponthoz, nyomjunk meg az **Enter** billentyűt.

6. Válasszuk ki a telepítéshez használni kívánt eszközt, mint például CD, FTP stb.
7. Menjünk fel az **Exit** (Kilépés) menüpontig, majd nyomjuk meg az **Enter** billentyűt.
8. Végezetül lépünk ki a **sysinstall** programból, aminhez nyomjuk meg az **X** billentyűt.

4.5.2. Portok telepítése

A “váz” fogalma az első, amit a Portgyűjteménnyel kapcsolatban tisztázni kell. Dióhéjban összefoglalva, egy port váza azon állományok legszűkebb halmaza, amelyek elárulják a FreeBSD számára, hogyan fordítsuk le hibamentesen és hogyan telepítsük az adott programot. Ehhez minden port vázában megtalálható:

- Egy **Makefile** nevű állomány. Ez tartalmazza azokat a különböző utasításokat, amelyek megmondják, hogyan kell lefordítani és hova kell telepíteni a rendszerünkben az adott alkalmazást.
- Egy **distinfo** nevű állomány. Ebben található információ a port lefordításához szükséges állományok letöltéséről, valamint a letöltött állományok ellenőrzéséhez szükséges (az **md5(1)** és **sha256(1)** programokkal számolt) ellenőrzőösszegek.
- Egy **files** alkönyvtár. Itt találhatjuk meg azokat a javításokat, amelyek alkalmazásával le tudjuk fordítani a programot FreeBSD-n is. Ezek a javítások többnyire bizonyos állományok módosításaira vonatkozó apró állományok formájában jelennek meg. Természetüknél fogva szöveges formátumúak, és általában olyanok szerepelnek bennük, hogy “Töröld a 10. sort” vagy “Változtasd meg a 26. sort erre: ...”. Ezeket a javításokat eredetileg patcheknek (foltoknak) nevezik, vagy másképp diffeknek (eltéréseknek) is, mivel a **diff(1)** program segítségével hozzák ezeket létre.

Ez a könyvtár tartalmazhat további állományokat is portok elkészítéséhez.

- Egy **pkg-descr** nevű állomány. Ez a program részletesebb, gyakran többsoros bemutatása.
- Egy **pkg-plist** nevű állomány. Itt találjuk meg a port által telepítendő összes állományt. Ez egyben közli a portrendszerrel is, hogy az eltávolítás során mely állományokat kell majd törölnie.

Egyes portokban szerepelhetnek még egyéb állományok is, mint például a **pkg-message**. Ezeket az állományokat a portrendszer különleges helyzetek kezelésére tartogatja. Ha még többet kívánunk megtudni ezekről az állományokról, vagy magukról a portokról általánosságban, lapozzuk fel a FreeBSD porterek kézikönyv (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/porters-handbook/index.html).

A port ugyan tartalmazza a forráskód lefordításához szükséges utasításokat, de konkrétan a forráskódot nem. Ezt egy CD-ről vagy az internetről tudjuk megszerezni. A forráskód általában a szerzője által kedvelt formában jelenik meg: ez gyakran egy **gzip**-pel tömörített tar állomány, de lehet tömörítve mással is, vagy éppen lehet tömörítetlen. A program forráskódját, legyen akármilyen formában is, nevezik “**distfile**”-nak (terjesztési állománynak). A FreeBSD portok telepítésének két módszerét tárjuk fel a következőkben.

Megjegyzés: A portok telepítéséhez **root** felhasználóként kell bejelentkeznünk.

Figyelem Mielőtt telepítenénk bármelyik portot is, ajánlott frissíteni a Portgyűjteményünket és ellenőriznünk az adott portot a <http://vuxml.freebsd.org> címen található biztonsági adatbázisban.

Az újonnan telepítendő alkalmazások biztonsági sebezhetőségeinek ellenőrzését automatikussá is tehetjük a **portaudit** használatával. Ez a segédeszköz is a Portgyűjteményben található (**ports-mgmt/portaudit**).

Érdemes minden port telepítése előtt letöltenünk a legfrissebb sebezhetőségi adatbázist a `portaudit -F` parancs kiadásával. Mellesleg az adatbázis rendszeres frissítése és ez a biztonsági felülvizsgálat a naponként elvégzendő biztonsági ellenőrzések közt is megjelenik. Ezekről részletesebben a `portaudit(1)` és `periodic(8)` man oldalakon olvashatunk.

A Portgyűjtemény feltételezi, hogy működő internet-hozzáféréssel rendelkezünk. Amennyiben ez nem így lenne, a terjesztési állományokat, forráskódokat saját magunknak kell bemásolnunk a `/usr/ports/distfiles` könyvtárba.

A kezdéshez lépünk be a telepítendő port könyvtárába:

```
# cd /usr/ports/sysutils/lsof
```

Miután beléptünk az `lsof` könyvtárba, láthatjuk a port vázát. A következő lépés a fordítás, avagy a port “buildelése” (elkészítése). Ezt egy szimpla `make` parancs kiadásával kezdeményezhetjük. Miután megtettük, valami ilyesmit kell tapasztalnunk:

```
# make
>> lsof_4.57D.freebsd.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
>> Attempting to fetch from ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/.
==> Extracting for lsof-4.57
...
[ide jön a kitömörítés kimenete]
...
>> Checksum OK for lsof_4.57D.freebsd.tar.gz.
==> Patching for lsof-4.57
==> Applying FreeBSD patches for lsof-4.57
==> Configuring for lsof-4.57
...
[ide jön a configure szkript kimenete]
...
==> Building for lsof-4.57
...
[ide jön a fordítás kimenete]
...
#
```

A fordítás befejeztével visszakapjunk a parancssort. A soron következő lépés a port telepítése lesz. Ehhez mindössze egyetlen szóval kell kiegészítenünk a `make` parancs meghívását: ez a szó pedig az `install` (telepít) lesz.

```
# make install
==> Installing for lsof-4.57
...
[a telepítés kimenete kimarad]
...
==> Generating temporary packing list
==> Compressing manual pages for lsof-4.57
==> Registering installation for lsof-4.57
==> SECURITY NOTE:
    This port has installed the following binaries which execute with
    increased privileges.
#
```

Miután ismét visszakaptuk a parancssort, már futtatni is tudjuk a frissen telepített alkalmazásunkat. Mivel az `lsof` programnak tovább jogosultságokra is szüksége van, egy erről szóló biztonsági figyelmeztetést is láthatunk. A portok létrehozása és telepítése során érdemes figyelni az ehhez hasonló figyelmeztetésekre.

A telepítés befejeztével nem árt törölnünk a fordításhoz felhasznált alkönyvtárat (`work`) is. Ezzel nemcsak a drága lemezterületet spóroljuk meg, hanem megelőzzük a port későbbi frissítése során felmerülő esetleges problémákat is.

```
# make clean
==> Cleaning for lsof-4.57
#
```

Megjegyzés: Az eljárásból két lépést meg is tudunk takarítani, ha egyszerűen csak a `make install clean` parancsot adjuk ki az előbb három lépésben tagolt `make`, `make install` és `make clean` parancsok helyett.

Megjegyzés: Bizonyos parancsértelmezők a `PATH` környezeti változóban felsorolt könyvtárakban található parancsokat gyorsítótárban tárolják, ezzel felgyorsítva a hozzájuk tartozó végrehajtható állományok keresését. Ha történetesen ilyen parancsértelmezőt használnánk, az új portok telepítése után szükségünk lehet a `rehash` parancs kiadására, mivel enélkül nem tudjuk elérni a frissen telepített parancsokat. Ezt a parancsot például a `tcsh` és a hozzá hasonló parancsértelmezőkben találhatjuk meg, az `sh` és rokonainál pedig a `hash -r` ennek a megfelelője. A pontos információkat erről a témáról a parancsértelmezőnk dokumentációjában lelhetjük meg.

Némely külső DVD termék, mint például a FreeBSD Mall (<http://www.freebsdmail.com>)tól megrendelhető FreeBSD Toolkit, tartalmazhatnak terjesztési állományokat. Ezek remekül használhatóak a Portgyűjteménnyel. Ehhez csatlakoztatnunk kell a DVD-t a `/cdrom` könyvtárba. Ettől eltérő csatlakozási pontok használata esetén ne felejtjük el átállítani a `CD_MOUNTPTS` változót sem a `make` számára. Ekkor a fordításhoz szükséges állományokat úgy fogja kezelni a rendszer, mintha a merevlemezünkön lennének.

Megjegyzés: Vigyázzunk arra, hogy néhány portot nem lehet CD-n terjeszteni. Ez részben azért lehet, mert a szükséges állományok letöltéséhez, illetve újbóli terjesztéséhez ki kell tölteni valamilyen regisztrációs nyomtatványt, vagy pedig egyéb okok miatt. Tehát ha olyan portot akarunk telepíteni, ami nincs rajta a CD-n, mindenképpen rendelkezniünk kell internetkapcsolattal.

A portrendszer a `fetch(1)` segédprogramot használja az állományok letöltésére, amely figyelembevesz különféle környezeti változókat, ilyenek többek közt az `FTP_PASSIVE_MODE`, `FTP_PROXY` és az `FTP_PASSWORD`. Ha tűzfal mögött vagyunk, szükségünk lehet ezek némelyikének helyes beállítására, vagy FTP/HTTP proxyt kell használnunk. A `fetch(3)` man oldala tartalmazza ezen változók teljes listáját.

A `make fetch` azon felhasználók számára nyújt segítséget, akik nem csatlakoznak minden esetben a hálózatra. Egyszerűen csak futtassuk le a könyvtárszerkezet legtetéről (`/usr/ports`) ezt a parancsot és a szükséges állományok letöltődnek nekünk. A parancs működik az alsóbb szinteken is, például a `/usr/ports/net` könyvtárban. Azonban legyünk tekintettel arra, hogy ha egy port függ más portoktól vagy függvénykönyvtáraktól, ez a parancs *nem fogja* letölteni a hozzájuk tartozó állományokat. Ilyenkor a `fetch` helyett használjuk a `fetch-recursive` targetet.

Megjegyzés: Ha a `make` parancsot egy felsőbb szinten futtatjuk, akkor ezzel létre tudjuk hozni az összes vagy csak kategóriánként az összes portot, hasonlóan az előbb említett `make fetch` módszerhez. Ez azonban

veszélyes, mivel egyes portok kizárják mások használatát. Emellett előfordulhat az is, hogy bizonyos portok ugyanazon a néven telepítenek több, tartalmukban különböző állományt.

Nagyon ritkán adódhat, hogy a felhasználónak nem a MASTER_SITES által mutatott helyekről kell beszereznie a szükséges állományokat (innen töltődnek ugyanis le). A MASTER_SITES beállítást az alábbi paranccsal bírálhatjuk felül:

```
# cd /usr/ports/könyvtár
# make MASTER_SITE_OVERRIDE= \
ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/ fetch
```

Ebben a példában a MASTER_SITES értékét a ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/ címre változtattuk meg.

Megjegyzés: A portok némelyike lehetővé teszi (esetleg meg is követeli), hogy engedélyezzük vagy letiltuk a készülő program bizonyos elemeit hatékonysági, biztonsági vagy egyéb testreszabási irányelvek mentén. Ilyen többek közt a `www/mozilla`, a `security/gpgme` és a `mail/sylpheed-claws`. Ha elérhetőek ilyen beállítási lehetőségek, arról a rendszer egy üzenetben tájékoztat minket.

4.5.2.1. Az alapértelmezett könyvtárak felülbírálnak

Néha hasznos (vagy kötelező) lehet eltérő munka- és célkönyvtárak alkalmazása. A WRKDIRPREFIX és a PREFIX változókkal ezek alapértelmezéseit tudjuk megváltoztatni. Például a

```
# make WRKDIRPREFIX=/usr/home/example/ports install
```

parancs a portot a /usr/home/example/ports könyvtárban fogja lefordítani és az eredményét a /usr/local könyvtárba telepíti. A

```
# make PREFIX=/usr/home/example/local install
```

parancs hatására a port a /usr/ports könyvtárban készül el és a /usr/home/example/local könyvtárba települ.

Természetesen a

```
# make WRKDIRPREFIX=./ports PREFIX=./local install
```

parancs ötvözi az előbbi kettőt (amelyet most túlságosan is hosszú lenne kiírni, de vélhetően sejthető belőle az alapötlet).

Lehetőség van ezen változókat a saját környezetünkben is beállítani. Ha erre lenne szükségünk, nézzünk utána az ezzel kapcsolatos teendőnek a parancsértelmezőnk man oldalán.

4.5.2.2. Az imake használatáról

Bizonyos portok az (X Window System részeként megjelenő) imake segédprogramra támaszkodnak, ahol viszont nem működik a PREFIX átállítása és mindenképpen a /usr/X11R6 könyvtárba akar telepíteni. Ehhez hasonlóan egyes Perl portok figyelmen kívül hagyják a PREFIX változót és közvetlenül a Perl fájlba kerülnek. Az ilyen portok esetén nagyon nehéz vagy szinte lehetetlen betartatni a PREFIX használatát.

4.5.2.3. A portok újrakonfigurálása

Egyes portok lefordítása előtt megjelenik egy ncurses alapú menü, ahol ki tudunk választani bizonyos fordítási beállításokat. Gyakran előfordul, hogy a port lefordítása után a felhasználók szeretnék újra előhozni ezt a menüt és megadni vagy kivenni bizonyos beállításokat. Erre több mód is kínálkozik. Egyik ilyen lehetőség az, ha belépünk a port könyvtárba és kiadjuk a `make config` parancsot, amivel lényegében ismét előcsaljuk a beállításokat összefoglaló menüt. Másik ilyen lehetőség a `make showconfig` alkalmazása, amivel a porthoz tartozó összes beállítást tudjuk egyszerre megjeleníteni. Ezek mellett még használható a `make rmconfig` parancs is, amivel törölni tudjuk az összes eddigi beállítást és így újakezdhetjük a port konfigurációját. Ezek és a többi ilyen opció a ports(7) man oldalon kerül bővebb kifejtésre.

4.5.3. A portok eltávolítása

Most már tudjuk, miként lehet portokat telepíteni, azonban valószínűleg még az is érdekelhet minket, hogy miként kell ezeket eltávolítani abban az esetben, ha például később meggondolnánk magunkat velük kapcsolatban. A korábban telepített példaportot fogjuk eltávolítani (a figyelmetlenek kedvéért megemlítjük, hogy ez az `lsOf` volt). A portok eltávolítása teljesen egybevág a csomagokéval (erről a csomagokról szóló részben beszéltünk), mivel ekkor is használhatjuk a `pkg_delete(1)` parancsot:

```
# pkg_delete lsOf-4.57
```

4.5.4. A portok frissítése

Először is a `pkg_version(1)` parancs felhasználásával listázzuk ki azokat a portokat, amik felett már eljárt az idő és a Portgyűjteményben található belőlük újabb verzió:

```
# pkg_version -v
```

4.5.4.1. A `/usr/ports/UPDATING` állomány

Miután frissítettük a Portgyűjteményünket, de még mielőtt megpróbálnánk akármelyik portot is frissíteni, érdemes egy pillantást vetnünk a `/usr/ports/UPDATING` állományra. Itt megtalálhatóak azok a problémák és a hozzájuk tartozó lépések, amelyekkel a felhasználóknak a portok frissítése során szembe kell nézniük, beleértve az állományformátumok, a konfigurációs állományok helyének megváltozását vagy egyéb olyan módosításokat, amik a korábbi verziókkal összeférhetetlenséget szülhetnek.

Amennyiben az `UPDATING` állomány tartalma ellentmondana az itt olvasottakkal, mindig az `UPDATING` állományban leírtak az irányadók.

4.5.4.2. Portok frissítése a `portupgrade` használatával

A `portupgrade` nevű segédprogramot a portok egyszerűbb frissítésére találták ki, és a `ports-mgmt/portupgrade` portban található meg. A `make install clean` paranccsal bármelyik más porthoz hasonlóan telepíthetjük:

```
# cd /usr/ports/ports-mgmt/portupgrade
# make install clean
```

A `pkgdb -F` paranccsal fésültessük át a telepített portok listáját, és javítsuk az általa jelentett ellentmondásokat. Érdemes rendszeresen elvégezni ezt, lehetőleg minden frissítés előtt.

Miután kiadtuk a `portupgrade -a` parancsot, a **portupgrade** nekilát frissíteni az összes elavult portot a rendszerünkben. Ha minden egyes frissítést külön meg szeretnénk erősíteni, használjuk a `-i` kapcsolót is.

```
# portupgrade -ai
```

Ha nem akarjuk az összes portot frissíteni, csupán egy bizonyos alkalmazását, használjuk a `portupgrade pkgname` paraméterezést. A `-R` kapcsoló megadásával a **portupgrade** először frissíti az adott alkalmazás függőségeit.

```
# portupgrade -R firefox
```

Ha a művelet során csomagokat kívánunk használni portok helyett, adjuk meg a `-P` kapcsolót. Ennek révén a **portupgrade** megkeresi a csomagokat a `PKG_PATH` környezeti változóban felsorolt könyvtárakban vagy ha itt nem találja, letölti ezeket egy távoli szerverről. Amennyiben a csomagokat sem helyben, sem pedig a távoli szerveren nem találja, a **portupgrade** helyettük portokat fog használni. Ilyenkor a portok használatát a `-PP` kapcsoló beállításával lehet elkerülni:

```
# portupgrade -PP gnome2
```

Csak a terjesztési állományok (vagy a `-P` esetén csomagok) letöltéséhez használjuk a `-F` kapcsolót. Mindezekről részletesebben a `portupgrade(1)` man oldalon olvashatunk.

4.5.4.3. Portok frissítése a Portmanager használatával

A **Portmanager** egy másik hasznos segédprogram a portok könnyű frissítéséhez. A `ports-mgmt/portmanager` porton keresztül érhető el:

```
# cd /usr/ports/ports-mgmt/portmanager
# make install clean
```

Használatával az összes telepített port egyetlen paranccsal frissíthető:

```
# portmanager -u
```

Ha a **Portmanager** minden egyes lépését külön meg kívánjuk erősíteni, akkor a `-ui` kapcsolókat se felejtjük el megadni. A **Portmanager** emellett új portok telepítésére is használható. Eltérően a `make install clean` parancsban megszokottaktól, a kiválasztott port összes függőségét még a fordítás és a telepítés előtt fogja frissíteni.

```
# portmanager x11/gnome2
```

Ha bármilyen gondot tapasztalnánk a kiválasztott port függőségeit illetően, a **Portmanagert** felkérhetjük az összes függőség helyes sorrendben történő újrafordítására. Amikor befejezte, a problémás portot is újra létrehozza.

```
# portmanager graphics/gimp -f
```

Bővebb információkért lásd `portmanager(1)`.

4.5.4.4. Portok frissítése a Portmaster használatával

A **Portmaster** szintén a portok frissítésére alkalmas segédprogram. A **Portmaster** esetében a hangsúly az “alaprendszerben” is megtalálható eszközök használatán van (tehát nem függ semmilyen más porttól) és a `/var/db/pkg/` könyvtárban található információk alapján dönti el, hogy milyen portokat kell frissítenie. A `ports-mgmt/portmaster` portból érhető el:

```
# cd /usr/ports/ports-mgmt/portmaster
# make install clean
```

A **Portmaster** a portokat az alábbi négy kategória valamelyikébe sorolja be:

- Gyökér (root) portok (nem függenek semmitől, semmi sem függ tőlük)
- Törzs (trunk) portok (nem függenek semmitől, de mások függenek tőlük)
- Ág (branch) portok (vannak függőségeik és mások is függenek tőlük)
- Levél (leaf) portok (vannak függőségeik, de semmi sem függ tőlük)

A következő paranccsal le tudjuk kérni az összes telepített portot és az `-L` kapcsolóval frissítéseket keresni hozzájuk:

```
# portmaster -L
====>> Root ports (No dependencies, not depended on)
====>> ispell-3.2.06_18
====>> screen-4.0.3
====>>      ====>> New version available: screen-4.0.3_1
====>> tcpflow-0.21_1
====>> 7 root ports
...
====>> Branch ports (Have dependencies, are depended on)
====>> apache-2.2.3
====>>      ====>> New version available: apache-2.2.8
...
====>> Leaf ports (Have dependencies, not depended on)
====>> automake-1.9.6_2
====>> bash-3.1.17
====>>      ====>> New version available: bash-3.2.33
...
====>> 32 leaf ports

====>> 137 total installed ports
====>> 83 have new versions available
```

Az összes telepített port egyetlen egyszerű paranccsal frissíthető:

```
# portmaster -a
```

Megjegyzés: A **Portmaster** alapértelmezés szerint minden egyes törlendő korábbi portról biztonsági másolatot készít. Amikor az új változat telepítése sikeresen lezajlott, akkor a **Portmaster** ezt a másolatot megsemmisíti. A `-b` paraméterrel azonban megkérhetjük, hogy ne törölje le a biztonsági mentést. Az `-i` megadásával a **Portmaster** interaktív módban indul el, és minden port frissítése előtt a felhasználó megerősítését fogja kérni.

Amennyiben valamilyen hiba lép fel a frissítés folyamán, az `-f` opció megadásával kérhetjük az összes port frissítését és újrafordítását is:

```
# portmaster -af
```

A **Portmaster** használatával új portokat is fel tudunk telepíteni a rendszerre úgy, hogy azok függőségeit is igyekszik frissíteni a lefordításuk előtt:

```
# portmaster shells/bash
```

A további részleteket a portmaster(8) man oldalon találjuk.

4.5.5. A portok tárigénye

A Portgyűjtemény idővel egyre több helyet fog elfoglalni a merevlemezünkön. Miután sikeresen létrehoztunk és telepítettünk egy szoftvert a hozzá tartozó portból, érdemes mindig eltakarítanunk magunk után a `work` könyvtárban menet közben keletkezett átmeneti állományokat a `make clean` parancs használatával. Az egész Portgyűjteményt egyetlen mozdulattal ezzel a paranccsal tudjuk végigsepregetni:

```
# portsclean -C
```

Az idő előrehaladtával a `distfiles` könyvtárban is rengeteg régi forrás tud felhalmozódni. Ezeket eltávolíthatjuk kézzel, vagy az alábbi parancs segítségével törölhetjük az összes olyan terjesztési állományt, amelyekre már egyetlen port sem hivatkozik:

```
# portsclean -D
```

Vagy törölhetjük az összes olyan terjesztési állományt, amelyre egyetlen pillanatnyilag feltelepített port sem hivatkozik a rendszerünkben:

```
# portsclean -DD
```

Megjegyzés: A `portsclean` segédprogram a **portupgrade** programcsomag része.

Ne felejtjük el eltávolítani azokat a portokat, amikre már nincs szükségünk a továbbiakban. Ebben a feladatban egy jól használható segédeszköz lehet a segítségünkre, a `ports-mgmt/pkg_cutleaves` port.

4.6. Telepítés utáni teendők

Az új alkalmazás feltelepítése után minden bizonnyal szeretnénk elolvasni a hozzá társított dokumentációt, az egyedi beállításainknak megfelelően módosítani a konfigurációs állományokat, engedélyezni a rendszerindítás során történő automatikus indítását (ha démonról lenne szó) és így tovább.

Az egyes alkalmazások beállításához elvégzendő lépések nyilvánvalóan egyedenként eltérőek. Azonban tudunk szolgálni néhány általános tanáccsal válaszként az ilyenkor felmerülő “Na és akkor most mi legyen?” kérdésre:

- Kérdezzük meg a `pkg_info(1)` programtól, milyen állományok és hova kerültek fel a telepítés során. Például, ha a `SzuperCsomag 1.0.0`-át raktunk fel, akkor a

```
# pkg_info -L SzuperCsomag-1.0.0 | less
```

parancs kilistázza az összes állományt, amit a csomagból felraktunk. Ezek közül leginkább a `man/` könyvtárban levőkre figyeljünk, mivel ezek lesznek az alkalmazás `man` oldalai. Ehhez hasonlóan az `etc/` könyvtárban a konfigurációs állományok és a `doc/` könyvtárban pedig a nagyobb lélegzetvételű dokumentációk foglalnak helyet.

Ha nem emlékszünk pontosan rá, hogy az alkalmazások melyik verzióját is telepítettük, a

```
# pkg_info | grep -i SzuperCsomag
```

alakú parancs megkeresi az összes olyan csomagot, aminek a nevében szerepel a `SzuperCsomag` szövegrészlet. A fenti példában természetesen igény szerint változtassuk meg a `SzuperCsomag` szöveget a tényleges csomag nevére.

- Ahogy sikerült megtalálnunk az alkalmazáshoz tartozó `man` oldalakat, lapozzuk fel ezeket a `man(1)` segítségével. Ugyanígy nézzük át a mellékelt minta konfigurációs állományokat és az összes elérhető dokumentációt.
- Ha az alkalmazásnak van saját honlapja, kutassunk ott is információk után, olvassuk el a gyakran ismételt kérdéseket és így tovább. Ha nem tudnánk pontosan a honlap címét, a

```
# pkg_info SzuperCsomag-1.0.0
```

kimenetéből könnyen előkeríthető. Itt egy `www:` kezdetű sort kell keresnünk (már amennyiben létezik), amit az alkalmazás honlapjának címe kell kövessen.

- A rendszerrel együtt indítandó portok (ilyenek többek közt az internetes szolgáltatások), általában a `/usr/local/etc/rc.d` könyvtárba rakják a saját indítóskriptjüket. Érdemes leellenőrizni ezt a szkriptet és az igényeinknek megfelelően módosítani, átnevezni. A Szolgáltatások indítása című szakaszban ezt részleteiben is megismerhetjük.

4.7. Teendő a sérült portokkal

Ha véletlenül ráakadnánk egy olyan portra, ami nem működik megfelelően, nagyjából a következőket tudjuk tenni:

1. Derítsük ki a Hibajelentések adatbázisá (<http://www.FreeBSD.org/support.html#gnats>)ból, hogy készül-e már javítás az adott porthoz. Ha igen, akkor annak befejezése után már képesek leszünk használni.
2. Kérjük meg a port karbantartóját, hogy segítsen. A karbantartó elérhetőségének felderítéséhez gépeljük be a `make maintainer` parancsot, vagy keressük meg a `Makefile` állományban a karbantartó e-mail címét. Ne felejtsük el neki megemlíteni a port nevét és verzióját (vagyis mindenképpen küldjük el a `$FreeBSD:` sort a `Makefile` állományból) és a parancs kiadásától a hiba felbukkanásáig tartó kimenetet.

Megjegyzés: Némely portokat nem egyedülálló személyek tartanak karban, hanem egy levelezési lista (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/mailling-list-faq/article.html). A legtöbbjük neve, ha nem is mindé, nagyjából ilyen alakú: `<freebsd-listanév@FreeBSD.org>`. Egy ilyen jellegű kérdés megfogalmazása során ezt is vegyük figyelembe!

Kifejezetten a `<ports@FreeBSD.org>` karbantartóval rendelkező portoknak nincs rendes gazdája. A hozzájuk kapcsolódó javítások és mindenféle segítség, ötlet erről a levelezési listáról érkeznek. Ilyen esetekben számíthatunk az önkéntes segítőkre!

Ha nem kapunk semmilyen választ, a hiba bejelentésére használhatjuk a `send-pr(1)` programot is (erről bővebben lásd a FreeBSD-s hibajelentések írása (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/problem-reports/article.html) című cikket).

3. Javítsuk meg mi magunk! A porterek kézikönyve (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/porters-handbook/index.html) részletesen taglalja a “portok” belső felépítését, így onnan elindulva akár magunktól is meg tudunk javítani egy esetlegesen sérült portot, vagy be is küldhetjük a sajátunkat!
4. Töltsük le a porthoz tartozó csomagot a hozzánk legközelebb levő FTP oldalról. A “központi” csomaggyűjtemény az `ftp.FreeBSD.org` címen, a `packages` nevű könyvtárban (<ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/>) található, de mielőtt ide fordulnánk, nézzük meg a hozzánk legközelebb levő tükörszerveret (<http://mirrorlist.FreeBSD.org/>) is! Ha egy csomagot így telepítünk, akkor több eséllyel fog működni és ráadásul még jóval gyorsabb is. A csomag telepítésére használjuk a `pkg_add(1)` programot.

Fejezet 5. Az X Window System

Az X.Org X11 szerveréhez igazította: Ken Tom és Marc Fonvieille.

5.1. Áttekintés

A FreeBSD az X11-en keresztül nyújt a felhasználók számára hatékony grafikus felhasználói felületet. Az X11 az X Window System szabadon elérhető változata, melyet az **Xorg** és az **XFree86** egyaránt implementál (valamint más egyéb programcsomagok is, amelyeket itt viszont nem tárgyalunk). A FreeBSD verziói a FreeBSD 5.2.1-RELEASE kiadással bezárólag a The XFree86 Project, Inc. által kiadott X11 szerveret, az **XFree86**-ot tartalmazzák alapértelmezés szerint. A FreeBSD 5.3-RELEASE kiadástól kezdve az X11 alapértelmezett és hivatalos változata az **Xorg**, melyet az X.Org alapítvány a FreeBSD-éhez nagyon hasonló licenc alatt fejleszt. A FreeBSD-hez kereskedelmi X szerverek is elérhetőek.

Ebben a fejezetben az X11 telepítését és beállítását járjuk végig, miközben a hangsúlyt az **Xorg** 7.5 kiadására helyezzük. Az **XFree86** (vagyis a FreeBSD olyan régebbi változata, ahol az **XFree86** az alapértelmezett X11 rendszer) vagy az **Xorg** korábbi kiadásainak beállításával kapcsolatban mindig találhatunk információkat a FreeBSD kézikönyv <http://docs.FreeBSD.org/doc/címen> található archivált változataiban.

Az X11 által támogatott megjelenítőkről bővebben az Xorg (<http://www.x.org/>) honlapján olvashatunk.

A fejezet elolvasása során megismerjük:

- az X Window System különböző alkotóelemeit, és hogy ezek miként működnek együtt;
- hogyan telepítsük és állítsuk be az X11-et;
- hogyan telepítsük és használjuk a különféle ablakkezelőket;
- hogyan használjunk TrueType® betűtípusokat az X11-ben;
- hogyan állítsuk be rendszerünkön a grafikus bejelentkezést (**XDM**).

A fejezet elolvasásához ajánlott:

- külső programok telepítésének ismerete (4 fejezet).

5.2. Az X áttekintése

Az X használata elsőre megdöbbentő lehet azok számára, akik olyan más grafikus környezetekben járatosak, mint például a Microsoft Windows vagy a Mac OS.

Míg az X minden komponensének részleteit és azok kapcsolatát nem szükséges megérteni a használatukhoz, néhány alapvető ismeret velük kapcsolatban elősegíti kiaknázni az X erősségeit.

5.2.1. Miért X?

Az X ugyan nem az első UNIX-ra íródott ablakozó rendszer, de fajtáját tekintve a legnépszerűbb. Az X eredeti fejlesztőcsapata az X előtt egy másik ablakozó rendszeren dolgozott, aminek a neve “W” (mint “Window”, azaz ablak) volt. Az X pedig az arab ábécében pontosan ezt a betűt követi.

Az X-et hívhatjuk “X”-nek, “X Window System”-nek, és még sok más néven. Előfordulhat azonban, hogy az “X Windows” elnevezés sértő lehet egyes emberek számára. Erről többet a X(7) man oldalon tudhatunk meg többet.

5.2.2. Az X kliens-szerver modellje

Az X-et már az elejétől kezdve hálózatközpontúnak tervezték, és ezért az ún. “kliens-szerver” modellt használja.

Az X modelljében az “X szerver” egy olyan számítógépen fut, amelyhez billentyűzetet, monitort és egeret csatlakoztattunk. A szerver feladatai között találjuk a megjelenítés irányítását az egerről és a billentyűzetről, valamint a többi bemeneti és kimeneti eszközről érkező adatok feldolgozását és így tovább (például a digitális táblák is használhatóak beviteli eszközként, illetve egy projektor is lehet megjelenítő). Mindegyik X alkalmazás (mint például az **XTerm** vagy a **Netscape**) egy kliens. A kliens üzeneteket küld a szervernek, például “Kérlek, rajzolj egy ablakot ezekre a koordinátákra”, és a szerver pedig olyan üzeneteket küld, mint például “A felhasználó az OK gombra kattintott”.

Az otthoni vagy a kisebb irodai környezetben az X szerver és az X kliensek általában ugyanazon a számítógépen futnak. Emellett azonban nagyon is lehetséges, hogy az X szerver egy kevésbé erős gépen fusson, miközben az X alkalmazások (a kliensek) az irodát kiszolgáló erősebb és drágább gépen fussanak. Egy ilyen konfigurációban az X kliensei és szerverei közti kommunikáció a hálózaton keresztül zajlik.

Jegyezzük meg, hogy az X szerver az a számítógép, ahol a monitor és a billentyűzet található, az X kliensek pedig azok a programok, amelyek az ablakokat jelenítik meg.

A protokollban semmi sem várja el, hogy a kliens és a szerver ugyanazon az operációs rendszeren vagy éppen ugyanolyan típusú számítógépen fusson. Ezért akár Microsoft Windows-on vagy Apple® Mac OS-en is indíthatunk X szervert, és számos különböző szabad valamint kereskedelmi alkalmazás képes pontosan erre.

5.2.3. Az ablakkezelő

Az X kialakításának filozófiája leginkább a UNIX kialakításának filozófiájához hasonlítható, vagyis “eszközöket, ne szabályokat”. Ez tehát azt jelenti, hogy az X nem köti meg, miként oldjuk meg vele a feladatokat. Helyette különféle eszközöket ad a felhasználó kezébe, és onnantól a saját felelőssége eldönteni, hogyan használja ki ezeket.

Ez a filozófia az X-ben egészen addig terjed, hogy nem rögzíti, hogyan nézzenek ki a képernyőn megjelenő ablakok, miként kell ezeket mozgatni az egérrel, milyen billentyűk lenyomásával közlekedhetünk az ablakok között (ami a Microsoft Windows esetén az **Alt+Tab**), hogyan nézzen ki az ablakok címsora, a bezárás funkciónak legyen-e rajtuk gombja és így tovább.

Ehelyett az X az összes ezzel járó felelősséget átadja az “ablakkezelő” (window manager) részére. Tucatnyi ilyen ablakkezelőt találhatunk az X-hez: **AfterStep**, **Blackbox**, **ctwm**, **Enlightenment**, **fvwm**, **Sawfish**, **twm**, **Window Maker** és még sok más. Ezen ablakkezelők mindegyike más és más kinézetet és hangulatot kínál fel: némelyikük támogatja a “virtuális munkaasztalok” (virtual desktop) létrehozását; néhányuk pedig megengedi, hogy mi magunk állítsuk be az asztal irányításához használt gombkombinációkat; köztük találhatunk olyat is, amelynek van “Start” gombja vagy ehhez hasonló eszköze; némelyek közülük ismerik a “témákat”, aminek révén a kinézetük és hangulatuk teljesen megváltoztatható. Az említett ablakkezelők és társaik a Portgyűjtemény x11-wm kategóriájában érhetőek el.

Ráadásul a **KDE** és a **GNOME** munkakörnyezetek mindegyikének van saját integrált ablakkezelője.

Az egyes ablakkezelők mellelleg eltérő beállítási módszerrel rendelkeznek. Némelyikük kézzel összeállított konfigurációs állományt vár, mások pedig külön grafikus eszközöket tartalmaznak erre a feladatra is. Az egyikük (a

Sawfish) konfigurációs állományát például a Lisp programozási nyelv egyik dialektusában kell megírni.

Az irányítás átadása: Az ablakkezelő másik fontos feladata lekezelni, hogy az egerrel miként tudjuk átadni az ablakok között az irányítást, vagyis a fókuszt (focus policy). Minden ablakkezelő rendszerben el kell tudnunk valahogy dönteni, hogy a beérkező billentyűleütések melyik ablakhoz vándoroljanak, valamint az ilyen értelemben aktív ablakot valamilyen módon jeleznünk is kell.

Ennek egyik ismert módszere a “fókuszt kattintásra” megoldás, amely modellt a Microsoft Windows rendszerekben találhatjuk meg. Itt az ablakok akkor válnak aktívvá, amikor rájuk kattintunk az egerrel.

Az X viszont nem kötelezi el magát egyik vezérlésátadási módszer mellett sem, helyette az ablakkezelő fogja majd eldönteni, melyik ablak birtokolja a fókuszt az adott pillanatban. A különböző ablakkezelők különböző fókuszevezérlési technikákat ismernek. Mindegyikük ismeri a kattintásos fókuszt, azonban a többségük emellett még sok más megoldást is felkínál.

A legnépszerűbb fókuszevezérlési elvek:

A fókuszt az egeret követi (focus-follows-mouse)

Az egérmutató alatt található ablak kapja meg fókuszt. Az érintett ablaknak nem kell feltétlenül az összes többi felett elhelyezkednie. Ilyenkor a fókuszt egyszerűen úgy vihetjük át egy másik ablakra, ha rámutatunk az egerrel, amihez még kattintanunk sem kell.

Hanyag fókuszt (sloppy-focus)

Ez az elv az előbbi apró kibővítése. Amikor a fókuszt az egérmutatót követi, és az egeret a leghátsó ablakra (vagy a háttérre) visszük, akkor valójában egyik ablak sem birtokolja az irányítást, ezért a leütemezett billentyűk elvesznek. A hanyag fókuszt használatával azonban az irányítás csak abban az esetben kerül át máshová, amikor egy másik ablakba lépünk be, nem pedig akkor, amikor a jelenlegiből lépünk ki.

Fókuszt kattintásra (click-to-focus)

Az aktív ablakot egy egérekattintással választjuk ki. Ilyenkor a kiválasztott ablak “felemelkedhet” és a többi előtt jelenhet meg. Ezt követően az összes irányítás ebbe az ablakba vándorol, még abban az esetben is, amikor egy másik ablakra visszük az egérmutatót.

Sok ablakkezelő ismer ezekből különböző variációkat, valamint rajtuk kívül más egyéb vezérlési elvet is. Ezzel kapcsolatban az adott ablakkezelő dokumentációjából deríthetünk ki a legtöbbet.

5.2.4. Widgetek

Az X megközelítése, vagyis az eszközök és nem a szabályok felsorakoztatása, kiterjed az egyes alkalmazásokban látható különféle widgetekre is.

A “widget” (window gadget, vagyis widget, de magyarul sok helyen a “műtyürke”) elnevezést azokra a felhasználói felületen megjelenő elemekre használjuk, amelyekkel valamilyen módon kapcsolatba léphetünk: kattinthatunk rájuk, “piszkálhatjuk” ezeket. Ilyenek többek közt a gombok, jelölőnégyzetek, rádiógombok, ikonok, listák és a többi. A Microsoft Windows nyelvén ezeket “vezérlőknek” (control) nevezzük.

A Microsoft Windows és az Apple Mac OS ezen a téren nagyon merev. Az alkalmazások fejlesztőinek gondoskodniuk kell róla, hogy a programjaik az elterjedt kinézetet és kialakítást kövessék. Az X viszont nem várja az egységes vezérlőeszközök vagy grafikai stílus használatát.

Ennek eredményeképpen az X cseppet sem kívánja meg az alkalmazásoktól, hogy közös kinézetben vagy viselkedésben osztozzanak. Természetesen léteznek népszerű eszközrendszerek és azoknak számos variációja is kialakult, beleértve az MIT Athenáját, a **Motif**®ot (amiről a Microsoft Windows eszközeit is mintázták, az összes ferde élet és a három szürkeárnyalatot), az **OpenLook**ot és társaikat.

Napjaink X alkalmazásai a **KDE** fejlesztéséhez használt Qt, esetleg a **GNOME**-hoz használt GTK+ könyvtárból származó, korszerű kinézetű widgeteket tartalmaznak. Ebből a szempontból megfigyelhető egyfajta tendencia a grafikus UNIX-alkalmazások felépítésében, ami minden bizonnyal megkönnyíti a kezdő felhasználók tájékozódását.

5.3. Az X11 telepítése

Az X11 FreeBSD-n alapértelmezett implementációja az **Xorg**. Az **Xorg** az X.Org alapítvány által kiadott, az X Window Systemet megvalósító nyílt forráskódú X szerver. Az **Xorg** az **XFree86 4.4RC2** és X11R6.6 kódja alapján készült. A FreeBSD Portgyűjteményében jelenleg az **Xorg 7.5** változata érhető el.

Az **Xorg**-ot a Portgyűjteményből így tudjuk lefordítani, majd telepíteni:

```
# cd /usr/ports/x11/xorg
# make install clean
```

Megjegyzés: Az egész **Xorg** lefordításához legalább 4 GB szabad helyre van szükségünk.

Az X11-et természetesen telepíthetjük közvetlenül csomagok segítségével is. A `pkg_add(1)` használatával telepíthető bináris csomagok is elérhetőek az X11-hez. Amikor a `pkg_add(1)` programra bízunk a csomag letöltését, ne adjunk meg verziószámot, a `pkg_add(1)` ugyanis mindig automatikusan az alkalmazás legfrissebb verzióját tölti le.

Az **Xorg** csomagjának letöltéséhez és telepítéséhez egyszerűen csak ennyit írjunk be:

```
# pkg_add -r xorg
```

Megjegyzés: A fentebb megadott példák a teljes X11 rendszert telepíteni fogják, beleértve a szervereket, klienseket, betűtípusokat stb. Az X11 egyes részeihez külön találhatunk csomagokat és portokat.

Ha csak az X11 legszükségesebb elemeit szeretnénk telepíteni, akkor alternatívaként választhatjuk az `x11/xorg-minimal` portot.

A fejezet további részében szót ejtünk az X11, valamint egy irodai használatra alkalmas munkakörnyezet beállításáról.

5.4. Az X11 beállítása

Írta: Christopher Shunway.

5.4.1. Mielőtt nekilátnánk

Az X11 beállítása előtt a célrendszer következő adataira lesz szükségünk:

- A monitor jellemzői
- A videokártya chipkészlete
- A videokártya memóriájának mérete

Az X11 a monitor jellemzőiből állapítja meg, hogy milyen felbontásban és frissítési frekvenciával működtesse azt. Ezek általában a monitorhoz tartozó dokumentációból vagy a gyártó honlapjáról deríthetők ki. Igazából két értékre van szükségünk: a függőleges és a vízszintes frissítési frekvenciára.

A videokártya chipkészlete határozza meg, hogy az X11 melyik meghajtóján keresztül kommunikál a grafikus hardverrel. Ez a legtöbb chipkészlet esetén magától megállapítható, de ennek ellenére mégis jó tisztában lenni ezzel arra az esetre, ha az automatikus felismerés mégsem működne.

A grafikus kártya memóriájának mérete határozza meg a rendszer által kihasználható felbontást és színmélységet. Ezt fontos tudunk ahhoz, hogy ismerjük a rendszerünk korlátait.

5.4.2. Az X11 beállítása

Az **Xorg** 7.3-as változatában gyakran mindenféle konfigurációs állomány használata nélkül egyszerűen csak adjuk ki a következő parancsot:

```
% startx
```

A **Xorg** 7.4 verziójától kezdődően a számítógépünkhöz csatlakoztatott egerek és billentyűzetek HAL segítségével automatikusan felismerhetők. Ennek megfelelően a `x11/xorg` port függőségeként telepítődni fognak a `sysutils/hal` és `devel/dbus` portok, viszont az `/etc/rc.conf` állományban a következő sorok hozzáadásával külön engedélyeznünk kell még ezeket:

```
hald_enable="YES"
dbus_enable="YES"
```

Ezeket a szolgáltatásokat még az **Xorg** beállítása előtt el kell indítanunk (a parancssorból manuálisan vagy a rendszer újraindításával).

Bizonyos hardvereszközök esetén az automatikus felismerés még nem működik megbízhatóan vagy nem jól állítja be az értékeket. Ilyen esetekben kézzel kell megadnunk a szükséges beállításokat.

Megjegyzés: A különböző munkakörnyezetek, mint például a **GNOME**, a **KDE** vagy éppen az **Xfce** általában tartalmaznak olyan segédprogramokat, amelyekkel a felhasználó könnyedén be tudja állítani a megjelenítés paramétereit, többek közt a képernyő felbontását. Tehát ha az alapértelmezések nem megfelelőek, viszont használni akarunk majd valamilyen munkakörnyezetet is, akkor egyszerűen csak telepítsük az adott környezetet és a hozzá tartozó eszközön keresztül állítsuk be a megjelenítést.

Az X11 beállítása egy többlépcsős folyamat. Első lépésünk egy alap konfigurációs állomány összeállítása lesz. Rendszeradminisztrátorként adjuk ki az alábbi parancsot:

```
# Xorg -configure
```

Ennek segítségével az X11 `xorg.conf.new` néven létrehozza a konfigurációs állomány vázát a `/root` könyvtárban (akár a `su(1)` parancsot használjuk, akár közvetlenül így jelentkezőnk be, az így örökölt rendszeradminisztrátori szerepkör maga után vonja a `$HOME` könyvtár átállítását is). Az X11 megpróbálja megkeresni a célrendszerben elérhető grafikus eszközöket, és létrehozni egy olyan konfigurációs állományt, amely az észlelt eszközökhöz tartozó meghajtókat tölti be.

A következő lépésünk legyen az imént létrehozott beállítás kipróbálása, amin keresztül ellenőrizhetjük, hogy az **Xorg** tényleg képes működni a célrendszer grafikus eszközén. Az **Xorg** 7.3 és azt megelőző változataiban ezt így tehetjük meg:

```
# Xorg -config xorg.conf.new
```

A **Xorg** 7.4 és későbbi változataiban a próba eredménye egy fekete képernyő lesz, amely meglehetősen megnehezítheti az X11 helyes működésének megállapítását. A `-retro` kapcsoló használatával azonban továbbra is elérhetjük a korábbi verziókban megszokott viselkedési módot:

```
# Xorg -config xorg.conf.new -retro
```

Ha ezután a képernyőn egy fekete-fehér rácsot látunk egy X alakú egérmutatóval a közepén, akkor jó a beállítás. A próbát úgy szakíthatjuk meg, ha először a **Ctrl+Alt+Fn** billentyűk együttes lenyomásával átváltunk valamelyik virtuális konzolra (például az **F1** esetén az elsőre), majd megnyomjuk a **Ctrl+C** gombokat.

Megjegyzés: Az **Xorg** korábbi változataiban a 7.3 verzióig bezárólag a **Ctrl+Alt+Backspace** billentyűkombinációval tudjuk leállítani a működését. Amennyiben erre továbbra is szükségünk lenne, a 7.4 és későbbi változatokban ezt úgy tudjuk engedélyezni, ha a begépeljük a következő parancsot egy X terminálablakban:

```
% setxkbmap -option terminate:ctrl_alt_bksp
```

Egy másik lehetséges megoldás, ha a billentyűzet beállításához létrehozunk a `/usr/local/etc/hal/fdi/policy` könyvtárban egy konfigurációs állományt `x11-input.fdi` néven a **hald** számára. Ebben az állományban a következőknek kell szerepelnie:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<deviceinfo version="0.2">
  <device>
    <match key="info.capabilities" contains="input.keyboard">
      <merge key="input.x11_options.XkbOptions" type="string">terminate:ctrl_alt_bksp</merge>
    </match>
  </device>
</deviceinfo>
```

A **hald** a számítógép újraindításával fogja majd beolvasni ezt az állományt.

Ilyenkor az `xorg.conf.new` állomány `ServerLayout` vagy `ServerFlags` szekciójához vegyük még hozzá az alábbi sort:

```
Option "DontZap" "off"
```

Ha az egér még nem működne, mindenképpen be kell állítanunk a továbblépés előtt. Ezzel kapcsolatban a FreeBSD telepítéséről szóló fejezetben levő 2.10.10 Szakaszt ajánljuk elolvasásra. Fontos megemlíteni, hogy az **Xorg 7.4** változatától kezdődően az `xorg.conf` `InputDevice` szekcióit az eszközök automatikusan észlelt beállításai felülbírálják. A régebbi változatok viselkedését úgy tudjuk visszanyerni, ha a `ServerLayout` és `ServerFlags` szekciók valamelyikéhez hozzáadjuk az alábbi sort:

```
Option "AutoAddDevices" "false"
```

Ezt követően a beviteli eszközök a lehetséges beállítási opciók (például a billentyűzet-kiosztás váltása) mentén a korábbiakban megszokott módon konfigurálhatóak.

Megjegyzés: Ahogy arról korábban szó esett, a 7.4 verziótól kezdődően a **hald** magától érzékelni fogja a számítógépre csatlakoztatott billentyűzetet. Előfordulhat, hogy a billentyűzet típusa vagy éppen kiosztása nem lesz megfelelő. Ennek beállítására többnyire a népszerűbb munkakörnyezetek, mint például a **GNOME**, **KDE** vagy **Xfce** tartalmaznak külön segédprogramot. A `setxkbmap(1)` vagy a **hald** konfigurációs szabályával azonban akár közvetlenül is meg tudjuk változtatni a billentyűzethez társított tulajdonságokat.

Például ha egy 102 gombos billentyűzetet szeretnénk használni francia kiosztással, akkor ehhez a `/usr/local/etc/hal/fdi/policy` könyvtárban kell létrehoznunk egy `x11-input.fdi` nevű állományt a **hald** részére. Ebben az állományban szerepeljenek az alábbi sorok:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<deviceinfo version="0.2">
  <device>
    <match key="info.capabilities" contains="input.keyboard">
      <merge key="input.x11_options.XkbModel" type="string">pc102</merge>
      <merge key="input.x11_options.XkbLayout" type="string">fr</merge>
    </match>
  </device>
</deviceinfo>
```

Ha létezik már ilyen állományunk, akkor a billentyűzet megfelelő beállításához egyszerűen csak másoljuk ki a fenti sorokat és adjuk hozzá.

Indítsuk újra a számítógépet, hogy a **hald** beolvassa az állományt.

Ugyanezt egy X terminálból is kényelmesen el tudjuk végezni:

```
% setxkbmap -model pc102 -layout fr
```

A paraméterként megadható billentyűzettípusokat és -kiosztásokat a `/usr/local/share/X11/xkb/rules/base.lst` állományban találhatjuk meg.

Ezután az ízlésünknek megfelelően hangoljuk be az `xorg.conf.new` állományt, nyissuk meg egy szövegszerkesztőben, például az `emacs(1)`-ben vagy az `ee(1)`-ben. Elsőként adjuk meg a célrendszerhez csatlakoztatott monitor frekvenciájára vonatkozó adatokat. Ezek általában a függőleges és a vízszintes frissítés értékei, melyeket az `xorg.conf.new` állomány "Monitor" szakaszában (Section) kell feltüntetni:

```
Section "Monitor"
  Identifier      "Monitor0"
  VendorName      "A monitor gyártója"
  ModelName       "A monitor típusa"
  HorizSync       30-107
  VertRefresh     48-120
```

```
EndSection
```

A konfigurációs állományból valószínűleg csak a `HorizSync` és `VertRefresh` kulcsszavak fognak hiányozni. Amennyiben ez tényleg így lenne, a megfelelő vízszintes frissítés értékét a `HorizSync` kulcsszó után, a hozzá tartozó függőleges frissítés értékét pedig a `VertRefresh` kulcsszó után kell hozzátennünk a szakaszhoz. Az iménti példában már megadtuk a célrendszer monitorának frissítési értékeit.

Az X megengedi, hogy DPMS (Energy Star) energiagazdálkodási szabványt ismerő monitorok lehetőséget is kihasználjunk. A `xset(1)` program vezérli a monitorok ki- és bekapcsolását, és segítségével készenléti vagy energiatakarékos üzemmódba tudjuk helyezni azokat. Ha engedélyezni kívánjuk a monitorunk DPMS lehetőségeit, egyszerűen csak tegyük hozzá az alábbi sort a monitorunkat leíró szakaszhoz:

```
Option      "DPMS"
```

Ha már a `xorg.conf.new` konfigurációs állomány szerkesztésével vagyunk elfoglalva, válasszuk ki számunkra kedvező alapértelmezett felbontást és színmélységet is. Ezt a "Screen" (Képernyő) nevű szakaszban tehetjük meg:

```
Section "Screen"
    Identifier "Screen0"
    Device     "Card0"
    Monitor    "Monitor0"
    DefaultDepth 24
    SubSection "Display"
        Viewport 0 0
        Depth    24
        Modes    "1024x768"
    EndSubSection
EndSection
```

A `DefaultDepth` kulcsszó után adjuk meg a rendszer alapértelmezett színmélységét. Ezt később az `Xorg(1)` `-depth` paraméterével bírálhatjuk felül a parancssorból. A `Modes` kulcsszó után jelennek meg azok a felbontások, amelyekben az adott színmélység elérhető. Itt csak olyan VESA szabványú módok jelenhetnek meg, amelyet a célrendszer grafikus eszköze is támogat. A fenti példában az alapértelmezett színmélység képpontonként huszonnégy bit, és ebben a színmélységben az elfogadott felbontás 1024-szer 768 pixel.

Végezetül mentsük el a szerkesztett konfigurációs állományt és próbáljuk ki a korábban leírt módszer szerint.

Megjegyzés: A hibakeresés során maguk az X11 naplóállományai is hasznos eszköznek bizonyulhatnak, mivel ezek minden olyan eszközről tartalmaznak információt, amelyekhez az X11 szervernek sikerült csatlakoznia. Az **Xorg** naplót a `/var/log/Xorg.0.log` elnevezést követő állományokban találjuk meg. A konkrét naplók nevei `Xorg.0.log`-tól `Xorg.8.log`-ig és így tovább terjedhetnek.

Ha minden a legnagyobb rendben haladt eddig, a konfigurációs állományt el kell tennünk egy olyan központi helyre, ahol az `Xorg(1)` képes lesz majd megtalálni. Ez a hely általában az `/etc/X11/xorg.conf` vagy a `/usr/local/etc/X11/xorg.conf`.

```
# cp xorg.conf.new /etc/X11/xorg.conf
```

Az X11 beállítását ezzel befejeztük. Az **Xorg** innentől elindítható a `startx(1)` segédprogram vagy az `xdm(1)` használatával.

5.4.3. Témák idősebbeknek és haladóknak

5.4.3.1. Az i810 grafikus chipkészlet beállítása

Az Intel i810 integrált chipkészletének meghajtásához szükségünk lesz az `agpart` nevű AGP programozási felületre az X11-ben. Erről az `agp(4)` meghajtó man oldalán olvashatuk többet.

Ennek segítségével ezt a hardvert is a többi grafikus kártyához hasonlóan állíthatjuk be. Vegyük figyelembe azonban, hogy az `agp(4)` meghajtót beépítve nem tartalmazó rendszermaggal futó rendszerekben a `kldload(8)` paranccsal utólag már nem tudjuk betölteni! Ezt a meghajtót már a rendszerindítás során be kell tudnunk tölteni: vagy a rendszermagba fordítjuk, vagy pedig a `/boot/loader.conf` állományban hivatkozunk rá.

5.4.3.2. Widescreen Flat Panel monitorok használata

Ebben a részben feltételezünk némi tapasztalatot a beállítások terén. Amennyiben a szabványos konfigurációs eszközök csődöt mondtak a beállítás során, magukból a naplóállományokból is kinyerhetünk elegendő információt ahhoz, hogy működésre bírjuk rendszerünket. Ehhez mindenképpen legyen kéznél egy szövegszerkesztő!

A jelenlegi szélesvásznú (WSXGA, WSXGA+, WUXGA, WXGA, WXGA+ és társai) formátumok a 16:10-es és 10:9-es képarányokat ismerik, amik néha gondot okozhatnak. Például a 16:10-es képarány felbontásai:

- 2560x1600
- 1920x1200
- 1680x1050
- 1440x900
- 1280x800

Bizonyos szempontból egyszerűen csak a fenti felbontások valamelyikét kell felvenni a "Screen" szakasz Mode sorába, valahogy így:

```
Section "Screen"
Identifier "Screen0"
Device      "Card0"
Monitor     "Monitor0"
DefaultDepth 24
SubSection "Display"
    Viewport 0 0
    Depth    24
    Modes     "1680x1050"
EndSubSection
EndSection
```

Az **Xorg** elég intelligens ahhoz, hogy a szélesvásznú megjelenítéssel kapcsolatos információkat lekérje a monitor I2C/DDC adatai közül, ezért meg tudja állapítani, hogy az eszköz milyen frissítési frekvenciákat és felbontásokat bír el.

Ha az alábbi `ModeLine` értékek nem szerepelnének a meghajtókban, akkor velük kapcsolatban egy kicsit sügnunk kell az **Xorg**-nak. A `/var/log/Xorg.0.log` átrágásával elegendő információt tudunk gyűjteni ahhoz, hogy manuálisan vegyünk fel használható `ModeLine` értékeket. Nem kell mást tennünk, mint ehhez hasonló sorokat keresnünk:

```
(II) MGA(0): Supported additional Video Mode:
(II) MGA(0): clock: 146.2 MHz   Image Size:  433 x 271 mm
(II) MGA(0): h_active: 1680   h_sync: 1784   h_sync_end 1960 h_blank_end 2240 h_border: 0
(II) MGA(0): v_active: 1050   v_sync: 1053   v_sync_end 1059 v_blanking: 1089 v_border: 0
(II) MGA(0): Ranges: V min: 48   V max: 85 Hz, H min: 30   H max: 94 kHz, PixClock max 170 MHz
```

Ezeket nevezik EDID-adatoknak (Extended display identification data, vagyis “bővített megjelenítési azonosító adatoknak”). Belőlük a megfelelő ModeLine sor létrehozása csupán annyiból áll, hogy a számértékeket a megfelelő sorrendbe tesszük:

```
ModeLine <name> <clock> <4 horiz. timings> <4 vert. timings>
```

Ezáltal a példában látott "Monitor" szakasz ModeLine sora így fog kinézni:

```
Section "Monitor"
Identifier      "Monitor1"
VendorName     "Bigname"
ModelName      "BestModel"
ModeLine       "1680x1050" 146.2 1680 1784 1960 2240 1050 1053 1059 1089
Option         "DPMS"
EndSection
```

Miután végrehajtottuk ezeket az egyszerű beállítási lépéseket, az X most már valószínűleg el fog indulni az új szélesvásznú monitorunkon.

5.5. Betűtípusok használata az X11-ben

Írta: Murray Stokely.

5.5.1. Type1 betűtípusok

Az X11-hez tartozó alap betűtípusok nem mondhatók kifejezetten ideálisnak például egy átlagos asztali kiadványszerkesztő alkalmazás számára. A nagyobb méretű bemutatókon a betűi szögletesen és idétlenül néznek ki, a **Netscape**ben megjelenő kisebb betűk pedig szinte teljességgel olvashatatlanok. Viszont manapság már rengeteg szabad, nagyon jó minőségű és könnyen használható Type1 (PostScript®) betűtípus érhető el az X11-hez. Például az URW betűtípus-gyűjtemény (`x11-fonts/urwfonts`) a szabványos Type1 betűtípusok (Times Roman®, Helvetica®, Palatino® és még sok más) jó minőségű változatait tartalmazza. A Freefonts nevű gyűjtemény (`x11-fonts/freefonts`) is tartalmaz sok más betűtípust, de a legtöbbjüket inkább csak a **Gimp**ben és a hozzá hasonló grafikai alkalmazásokban tudjuk használni, illetve nincsenek is még kellő mértékben befejezve a hétköznapi munkákhoz. Ezeken felül az X11 minimális ügyeskedéssel beállítható a TrueType betűtípusok használatára is. Erről részleteket a X(7) man oldalon, illetve a TrueType betűtípusokról szóló szakaszban olvashatunk.

A Portgyűjteményből az imént említett Type1 betűtípusokat az alábbi parancsok segítségével telepíthetjük:

```
# cd /usr/ports/x11-fonts/urwfonts
# make install clean
```

Ugyanígy járjunk el a freefont és a többi gyűjtemény esetén is. Az X szerver akkor fogja észlelni ezeket a betűtípusokat, ha hozzáadjuk a következő sort a konfigurációs állományához (`/etc/X11/xorg.conf`):

```
FontPath "/usr/local/lib/X11/fonts/URW/"
```

Vagy megtehetjük mindezt az X futtatása során is:

```
% xset fp+ /usr/local/lib/X11/fonts/URW
% xset fp rehash
```

Ez utóbbi beállítás viszont el fog veszni az X leállításával, hacsak nem vesszük hozzá az indítószkriptjéhez (ez az `~/xinitrc` a `startx` használata esetén, illetve az `~/xsession`, amikor egy **XDM**-szerű grafikus bejelentkezést használunk). Ezek mellett használhatjuk a `/usr/local/etc/fonts/local.conf` állományt is: erről az élsimítással foglalkozó szakaszban szólnunk részletesebben.

5.5.2. TrueType® betűtípusok

Az **Xorg** beépített támogatást tartalmaz a TrueType betűtípusok rendereléséhez. Két különböző modul valósítja meg ezt a feladatot. Ebben példában a `freetype` nevű modult használjuk, mivel sokkal jobban illeszkedik a többi betűrenderelőhöz. A `freetype` modul használatához mindössze az `/etc/X11/xorg.conf` állomány "Module" szakaszába kell beírunk a következő sort:

```
Load "freetype"
```

Most pedig hozzunk létre egy könyvtárat a TrueType betűtípusok számára (ez legyen például a `/usr/local/lib/X11/fonts/TrueType`), majd másoljuk az összes TrueType betűtípusunkat ide. Vigyázzunk rá, hogy Macintosh®-ról TrueType betűtípusok közvetlenül nem hozhatóak át, az X11 számára UNIX/MS-DOS/Windows formátumban kell lenniük. Miután sikerült átmásolnunk az állományokat ebbe a könyvtárba, használjuk a `ttmkfdir` parancsot a `fonts.dir` állomány létrehozására, aminek révén az X betűrenderelője tudni fogja, hogy új állományokat telepítettünk. A `ttmkfdir x11-fonts/ttmkfdir` néven elérhető a FreeBSD Portgyűjteményéből.

```
# cd /usr/local/lib/X11/fonts/TrueType
# ttmkfdir -o fonts.dir
```

Ezután adjuk hozzá a TrueType könyvtárat a betűtípusok könyvtáraihoz. Itt is a Type1 betűtípusoknál leírtak szerint kell eljárunk, vagyis használjunk a

```
% xset fp+ /usr/local/lib/X11/fonts/TrueType
% xset fp rehash
```

parancsot, vagy adjunk hozzá a `xorg.conf` állományhoz egy további `FontPath` sort.

Ezzel végeztünk is. Innentől kezdve a **Netscape**, **Gimp**, a **StarOffice™** és mindegyik X alkalmazás fel fogja ismerni a frissen telepített TrueType betűtípusokat. A nagyon kicsi betűk (egy honlap megtekintése során, nagyfelbontásban) és a nagyon nagy betűk (a **StarOffice** használatakor) most már sokkal jobban fognak mutatni.

5.5.3. A betűk élsimítása

Frissítette: Joe Marcus Clarke.

Az X11 által használt, a `/usr/local/lib/X11/fonts/` és a `~/fonts/` könyvtárakban található összes betűtípus élsimítása automatikusan elérhető az Xft-re felkészített alkalmazások számára. A mostanság megjelenő legtöbb

alkalmazás, mint például a **KDE**, **GNOME** és **Firefox**, ismeri az Xft-t.

A betűtípusok élsimításának be- és kikapcsolásához, valamint élsimítási jellemzőinek beállításához hozzuk létre (vagy ha már létezne, módosítsuk) a `/usr/local/etc/fonts/local.conf` állományt. Az Xft betűrendszer számos kifinomult lehetősége hangolható ezzel az állománnyal, amelyekből ebben a szakaszban csupán rövidke ízelítőt fogunk adni. A pontosabb részletekről a `fonts-conf(5)` man oldalon tájékozódhatunk.

Az állománynak XML formátumúnak kell lennie. Különösen ügyeljünk a kis- és nagybetűkre, illetve győződjünk meg mindig róla, hogy lezártuk-e az összes taget. Az állomány a szokásos XML-fejléccel kezdődik, amelyet egy DOCTYPE definíció követ, majd a `<fontconfig>` tag:

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<fontconfig>
```

Ahogy azt már korábban is említettük, a `/usr/local/lib/X11/fonts` és a `~/.fonts/` könyvtárakban található összes betűtípus élsimítása elérhető az Xft-re felkészített alkalmazások számára. Amennyiben ezeken túl még további könyvtárakat is fel kívánunk venni, írjuk bele a `/usr/local/etc/fonts/local.conf` állományba, nagyjából ilyen alakban:

```
<dir>/az/en/betu/tipusaim</dir>
```

Az új betűtípusok, de legfőképpen az új betűtípusokat tartalmazó könyvtárak hozzáadása után a betűkkel kapcsolatos gyorsítótárak frissítéséhez mindenképpen javasolt lefuttatni az alábbi parancsot:

```
# fc-cache -f
```

Az élsimítás hatására a betűk kontúrjai egy kissé elmosódnak, aminek köszönhetően a nagyon kis méretű szövegek sokkal olvashatóbbá válnak és eltűnnek a nagy méretű betűkről a "lépcsők", azonban a normál méretű betűknél megfájdulhat tőle a szemünk. A 14 pontnál kisebb méretű betűk esetén az alábbi sorok hozzáadásával tudjuk kikapcsolni az élsimítást:

```
<match target="font">
  <test name="size" compare="less">
    <double>14</double>
  </test>
  <edit name="antialias" mode="assign">
    <bool>>false</bool>
  </edit>
</match>
<match target="font">
  <test name="pixelsize" compare="less" qual="any">
    <double>14</double>
  </test>
  <edit mode="assign" name="antialias">
    <bool>>false</bool>
  </edit>
</match>
```

Bizonyos egyenszélességű (monospaced) betűtípusok élsimítása esetén a betűk távolsága nem megfelelő. Ez leginkább a **KDE** használata esetén merül fel. Ezt a problémát úgy is orvosolhatjuk, ha az ilyen betűtípusok térközét kézzel 100-ra állítjuk. Ehhez írjuk be a következő sorokat:

```

<match target="pattern" name="family">
  <test qual="any" name="family">
    <string>fixed</string>
  </test>
  <edit name="family" mode="assign">
    <string>mono</string>
  </edit>
</match>
<match target="pattern" name="family">
  <test qual="any" name="family">
    <string>console</string>
  </test>
  <edit name="family" mode="assign">
    <string>mono</string>
  </edit>
</match>

```

(ezzel lefedjük összes rögzített méretű (fixed) betűtípust "mono"-ként), majd vegyük hozzá ezt is:

```

<match target="pattern" name="family">
  <test qual="any" name="family">
    <string>mono</string>
  </test>
  <edit name="spacing" mode="assign">
    <int>100</int>
  </edit>
</match>

```

Egyes betűtípusoknál, mint például a Helveticánál, gondok akadhatnak az élsimítással. Ez általában egy függőlegesen kettévágottnak látszó betű képében jelenik meg. De ami a legrosszabb, hogy emiatt némely alkalmazás képes összeomlani. Ennek elkerülésére tegyük hozzá még az alábbi sorokat a `local.conf` állományhoz:

```

<match target="pattern" name="family">
  <test qual="any" name="family">
    <string>Helvetica</string>
  </test>
  <edit name="family" mode="assign">
    <string>sans-serif</string>
  </edit>
</match>

```

Miután befejeztük a `local.conf` szerkesztését, ellenőrizzük, hogy szerepel-e az állomány végén a `</fontconfig>` tag. Ha ugyanis nem zárjuk le rendesen, akkor a változtatásaink érvénytelenné válnak.

Végezetül a felhasználók is megadhatják a saját beállításait a saját `.fonts.conf` állományuk segítségével. Ehhez nem kell mást tenni, mindössze létrehozni egy `~/ .fonts.conf` XML-állományt.

Még egy utolsó ötlet: LCD képernyők esetén szükségünk lehet az ún. "sub-pixel sampling" (részképpont mintavételezési) technikára. Ezzel lényegében a (vízszintesen elválasztott) vörös, zöld és kék összetevőket külön-külön kezeljük a horizontális felbontás javítására. Bámulatos eredményeket lehet elérni a segítségével! A bekapcsolásához a következő sorokat kell beszúrnunk valahova a `local.conf` állományba:

```

<match target="font">
  <test qual="all" name="rgba">

```

```

        <const>unknown</const>
    </test>
    <edit name="rgba" mode="assign">
        <const>rgb</const>
    </edit>
</match>

```

Megjegyzés: A megjelenítő fajtájától függően lehet, hogy az `rgb` értéket `bgr-re`, `vrgb-re` vagy `vbgr-re` kell cserélnünk. Próbálgassuk és kiderül, hogy melyikkel működik jobban.

5.6. Az X bejelentkeztető képernyője

Írta: Seth Kingsley.

5.6.1. Összefoglalás

Az X bejelentkeztető képernyője (az X Display Manager vagy röviden csak **XDM**) az X Window System egyik kiegészítő eleme, melyet a bejelentkezések lebonyolítására használunk. Számtalan helyzetben hasznosnak bizonyulhat, beleértve a legkisebb “X terminálokat” és a legnagyobb hálózati szervereket is. Mivel az X Window System független hálózattól és protokolltól, a hálózaton összekapcsolt, X klienseket és szervereket futtató különböző számítógépek széles kombinációja előfordulhat. Az **XDM** egy grafikus felületen keresztül segít választani az elérhető szerverek között, valamint a felhasználók, például felhasználónév és jelszón keresztül, hitelesítésében.

Az **XDM** tulajdonképpen a felhasználó számára ugyanazokat a funkciókat nyújtja, mint a `getty(8)` program (erről bővebben lásd 26.3.2 Szakasz). Tehát: belépteti a felhasználót a szerverre, ahova csatlakozott, illetve elindítja helyette a hozzá tartozó munkamenet kezelőjét (ami általában egy X-es ablakkezelő). Az **XDM** megvárja ennek a programnak a befejeződését, ami egyben jelzi számára, hogy a felhasználó elvégezte a dolgát, és kilépteti a szerverről. Ezután az **XDM** újra várakozni kezd a következő felhasználóra, miközben a bejelentkezéshez és a szerver kiválasztásához szükséges képernyőket jeleníti meg.

5.6.2. Az XDM használata

A **XDM** használatához először telepítenünk kell rendszerünkre a `x11/xdm` portot (mivel az **Xorg** újabb változatai ezt alapértelmezés szerint már nem telepítik). Ezt követően az **XDM** démon a `/usr/local/bin/xdm` helyen található meg. A programot `root` felhasználóként bármikor tudjuk futtatni, és ez veszi kezelésbe a helyi gépen futó X szervert. Amennyiben az **XDM**-et a számítógép minden egyes indulása során el akarjuk indítani, egyszerűen csak adjuk hozzá a megfelelő bejegyzést az `/etc/ttys` állományhoz. Ennek a formai szabályairól és használatáról bővebben lásd 26.3.2.1 Szakasz. Az `/etc/ttys` alapértelmezett változatában az **XDM** démont ebben a formában találjuk meg a virtuális terminálok között:

```
tttyv8    "/usr/local/bin/xdm -nodaemon"  xterm    off secure
```

Ez a bejegyzés alapból nem aktív. Az engedélyezéséhez írjuk át az ötödik mezőben szereplő `off` (kikapcsolva) értéket `on` (bekapcsolva)-ra, majd indítsuk újra az `init(8)` programot a 26.3.2.2 Szakaszban leírtak szerint. Az első

mezőben találhatjuk a program által kezelt terminált, ez jelen esetünkben a `tttyv8`. Ennek megfelelően az **XDM** a 9. virtuális terminálon kezdi meg a futását.

5.6.3. Az XDM beállítása

Az **XDM** beállításait tartalmazó könyvtár a `/usr/local/lib/X11/xdm`. Itt találhatjuk meg azokat az állományokat, amelyek megváltoztatásával befolyásolhatjuk az **XDM** megjelenését és viselkedését. Általában a következő állományok bukkannak fel ezen a helyen:

Állomány	Leírás
<code>Xaccess</code>	A kliens hitelesítésének szabályrendszere.
<code>Xresources</code>	Az X erőforrásainak alapértelmezett értékei.
<code>Xservers</code>	Az ismert távoli és helyi X szerverek listája.
<code>Xsession</code>	A bejelentkezések során lefutó alapértelmezett szkript.
<code>Xsetup_*</code>	A bejelentkező felület indítása előtt indítandó alkalmazásokkal kapcsolatos szkript.
<code>xdm-config</code>	A gépen futó összes X szerver globális beállításai.
<code>xdm-errors</code>	A szerver által jelentett hibák.
<code>xdm-pid</code>	A jelenleg futó XDM-hez tartozó azonosító.

Ebben a könyvtárban találunk még néhány olyan programot és szkriptet, amelyekkel be tudjuk állítani a munkaasztalunkat az **XDM** futása alatt. Ezen állományok céljait egyenként ismertetni fogjuk. A felépítésükről és használatukról az `xdm(1)` man oldala árul el többet.

Az alapértelmezett beállítás egy téglalap alakú bejelentkező ablak, aminek tetején nagy betűkkel a gép neve olvasható, valamint alatta a “Login:” (felhasználói név) és “Password:” (jelszó) mezők várnak kitöltésre. Ez egy remek kiindulási alap az **XDM**-képernyő kinézetének megváltoztatásához.

5.6.3.1. Xaccess

Az **XDM**-mel szabályozott X szerverek által használt protokoll az X Display Manager Connection Protocol (XDMCP). Ez az állomány tartalmazza a távoli számítógépekről érkező XDMCP-kapcsolatok vezérlésére vonatkozó szabályokat. Ezt a rendszer általában figyelmen kívül hagyja, hacsak az `xdm-config` állományban be nem állítottuk a távoli számítógépek csatlakoztathatóságát. Alapértelmezés szerint viszont semmilyen klienst nem enged csatlakozni.

5.6.3.2. Xresources

Ez tartalmazza a szerverválasztó és bejelentkező képernyő alapértelmezéseit. Segítségével a bejelentkeztetést végző program kinézetét változtathatjuk meg. Formátuma hasonló az X11 dokumentációjában leírt `app-defaults` állományhoz.

5.6.3.3. Xservers

A szerverválasztó által felkínálandó távoli X szerverek felsorolását tartalmazza.

5.6.3.4. Xsession

A felhasználó bejelentkezése után ez az **XDM**-szkript fog lefutni. Általában minden felhasználóhoz tartozik egy saját `~/.xsession` szkript, ami ezt felülbírálja.

5.6.3.5. Xsetup_*

Ezek fognak automatikusan lefutni a szerverválasztó vagy bejelentkeztető felületek megjelenése előtt. Minden általunk használt X szerverhez tartozik egy ilyen szkript, amelyek neve `xsetup_`-al kezdődik és a helyi X szerver sorszámaival folytatódik (például `xsetup_0`). Ezek a szkriptek általában egy-két programot, mint például az `xconsole`, indítanak el a háttérben.

5.6.3.6. xdm-config

Az `app-defaults` nevű állományéhoz hasonló alakban tartalmaz beállításokat a program által kezelt minden egyes X szerverhez.

5.6.3.7. xdm-errors

Ebben található meg az **XDM** által futtatni próbált X szerverek kimenete. Itt érdemes hibüzenetek után kutatni, ha az **XDM** által indított X szerver valamiért megállna. Ezek az üzenetek egyébként a felhasználó `~/.xsession-errors` állományába is beíródnak.

5.6.4. Hálózati X szerver futtatása

Az X szerverünkhöz csak akkor tudnak kívülről más felhasználók is kapcsolódni, ha átírjuk a hozzáférésre vonatkozó szabályokat és engedélyezzük rajta a kapcsolódást. Az alapértelmezett szabályok nagyon óvatosak. Ha tehát engedélyezni akarjuk a kívülről érkező kapcsolódásokat, akkor ahhoz először az `xdm-config` állományból vegyük ki az alábbi sort:

```
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort: 0
```

Ezután indítsuk újra az **XDM**-et. Ne felejtsük el, hogy az `app-defaults` állományokban a megjegyzések “!” (felkiáltó)jellel kezdődnek, nem pedig a megszokott “#” (kettőskereszt)tel. A fentieknél természetesen szigorúbb hozzáférési szabályok is szükségesek lehetnek — ezzel kapcsolatban nézzük meg `xaccess` állományban szereplő példákat, illetve lapozzuk fel az `xdm(1)` man oldalt.

5.6.5. Az XDM helyett

Az alapértelmezett **XDM** feladatát számos más program is képes ellátni. Ezek közül az egyik a **kdm** (a **KDE** része), amire ebben a fejezetben még vissza fogunk térni. A **kdm** különféle vizuális effekteket és egyéb kozmetikázást ígér, valamint lehetővé teszi a felhasználók számára, hogy a bejelentkezés előtt kiválaszthassák a használni kívánt ablakkezelőt.

5.7. Munkakörnyezetek

Írta: Valentino Vaschetto.

Ebben a szakaszban a FreeBSD-n futó X-hez elérhető különböző munkakörnyezetekről (desktop environment) lesz szó. Maga a “munkakörnyezet” elnevezés sok mindenre utalhat egy mezei ablakkezelőtől kezdve az asztali alkalmazások teljes garmadájáig, ahogy igaz ez a **KDE** vagy a **GNOME** esetében is.

5.7.1. A GNOME

5.7.1.1. Röviden a GNOME-ról

A **GNOME** egy felhasználóbarát munkakörnyezet, aminek segítségével a felhasználók számára gyerekjáték a számítógép használata és beállítása. A **GNOME**-ban találhatunk egy panelt (az alkalmazások indítására és különféle állapotjelzők megjelenítéséhez), egy asztalt (ahova az alkalmazások és az adatok kerülnek), szabványos asztali eszközöket és alkalmazásokat, valamint számos konvenciót, aminek mentén az alkalmazások könnyen együtt tudnak működni és tartani egymással az összhangot. Más operációs rendszerek vagy környezetek ismerői otthon érezhetik magukat ebben a **GNOME** által nyújtott vizuális környezetben. A FreeBSD és a **GNOME** kapcsolatáról bővebb információkat a FreeBSD GNOME Projekt (<http://www.FreeBSD.org/gnome>) honlapján találhatunk. Ezen az oldalon a **GNOME** telepítéséről, beállításáról és karbantartásáról egy meglehetősen átfogó leírást olvashatunk.

5.7.1.2. A GNOME telepítése

A programot könnyen fel tudjuk telepíteni csomagból vagy a Portgyűjtemény segítségével:

A hálózatról a **GNOME** csomagját mindössze ennek a sornak a beírásával fel tudjuk telepíteni:

```
# pkg_add -r gnome2
```

A portfa felhasználásával pedig a **GNOME**-ot így tudjuk forrásból telepíteni:

```
# cd /usr/ports/x11/gnome2
# make install clean
```

Miután a **GNOME**-ot sikerült feltelepítenünk, meg kell mondanunk az X szervernek, hogy az alapértelmezett ablakkezelő helyett a **GNOME**-ot indítsa el.

A **GNOME**-ot legkönnyebben a **GDM**, vagyis a GNOME Display Manager használatával indíthatjuk el. A **GDM** a **GNOME** részeként települ (habár alaphoz nincs bekapcsolva), és úgy tudjuk aktiválni, ha `/etc/rc.conf` állományba beírjuk a `gdm_enable="YES"` sort. Újraindítás után a **GDM** automatikusan elindul.

Ha a **GDM** mellett az összes **GNOME** szolgáltatást is el akarjuk indítani, vegyük fel a `gdm_enable="YES"` sort az `/etc/rc.conf` állományba.

A **GNOME**-ot parancssorból is elindíthatjuk, ha hozzá megfelelően beállítjuk az `.xinitrc` nevű állományt. Ha már van egy saját `.xinitrc` állományunk, akkor nincs más teendők, mint átírni az aktuális ablakkezelőnket hívó sort a `/usr/local/bin/gnome-session` sorra. Ha nem csináltunk előtte semmilyen különleges dolgot az említett konfigurációs állománnyal, akkor elegendő csak ennyit beírunk:

```
% echo "/usr/local/bin/gnome-session" > ~/.xinitrc
```

Ezt követően írjuk be a `startx` parancsot, és a **GNOME** munkakörnyezete fog elindulni.

Megjegyzés: Ha az **XDM**-hoz hasonló régebbi bejelentkeztető képernyőt használunk, ez a módszer nem fog működni. Helyette hozzunk létre egy `.xsession` nevű futtatható állományt, amely ezt a parancsot tartalmazza. Ehhez nyissuk meg és cseréljük ki benne a korábbi ablakkezelőnk hívását a `/usr/local/bin/gnome-session` utasításra:

```
% echo "#!/bin/sh" > ~/.xsession
% echo "/usr/local/bin/gnome-session" >> ~/.xsession
% chmod +x ~/.xsession
```

Megcsinálhatjuk azt is, hogy a bejelentkezéskor választható legyen az ablakkezelő. A KDE-ről bővebben című szakaszban látni fogjuk, hogyan tudjuk ezt a **KDE** bejelentkeztető képernyője, a **kdm** esetén beállítani.

5.7.2. A KDE

5.7.2.1. Röviden a KDE-ről

A **KDE** egy könnyen használható modern munkakörnyezet. Ízelítőül a **KDE** felhasználók számára felkínált lehetőségei közül:

- Gyönyörű, korszerű munkafelület
- Az asztal hálózaton keresztüli transzparens kezelése
- A **KDE** asztal és alkalmazásainak használatában egy beépített sűgőrendszer segíti a kényelmes és összefüggő közlekedést
- A **KDE** alkalmazásainak összehangolt kinézete és hangulata
- Szabványosított menük és eszköztárak, billentyű-hozzárendelések, színsémák stb.
- Honosítás: a **KDE** több, mint 40 nyelven elérhető
- Központosított, összehangolt, párbeszédablak alapú asztalbeállítás
- Számos hasznos **KDE**-alkalmazás

A **KDE**-hez egy **Konqueror** nevű böngésző is tartozik, mely a többi UNIX-os böngésző komoly ellenfelének bizonyul. A **KDE**-ről többet a KDE honlapján (<http://www.kde.org/>) olvashatunk. A **KDE** FreeBSD-re vonatkozó tudnivalóiról és a hozzá tartozó anyagokról a FreeBSD KDE csapat (<http://freebsd.kde.org/>) honlapján találhatunk információkat.

FreeBSD alatt a **KDE** két verziója érhető el: a harmadik változat már régóta használható, nagyon megbízható, amely mellett viszont a következő generációt képviselő negyedik változat is megtalálható a Portgyűjteményben. Akár egymás mellé is telepíthetők.

5.7.2.2. A KDE telepítése

Ahogy a **GNOME** és a többi más munkakörnyezet esetében is, maga a program könnyen telepíthető csomagból vagy a Portgyűjtemény segítségével is:

A **KDE3** csomagját hálózaton keresztül így tudjuk telepíteni:

```
# pkg_add -r kde
```

A **KDE4** csomagját pedig hálózaton keresztül így tudjuk telepíteni:

```
# pkg_add -r kde4
```

A `pkg_add(1)` magától letölti az alkalmazás legfrissebb verzióját.

Ha a **KDE3** környezetet forrásból akarjuk telepíteni, használjuk a portfát:

```
# cd /usr/ports/x11/kde3
# make install clean
```

Ha viszont a **KDE4** környezetet akarjuk inkább a portfa felhasználásával forrásból telepíteni, akkor ezeket a parancsokat adjuk ki:

```
# cd /usr/ports/x11/kde4
# make install clean
```

Miután a **KDE**-t sikeresen telepítettük, tudatnunk kell az X szerverrel, hogy az alapértelmezett ablakkezelő helyett ezt indítsa el. Ezt az `.xinitrc` állomány módosításával érhetjük el.

KDE3 esetén:

```
% echo "exec startkde" > ~/.xinitrc
```

KDE4 esetén:

```
% echo "exec /usr/local/kde4/bin/startkde" > ~/.xinitrc
```

Mostantól pedig mindig **KDE** lesz az asztalunk, amikor az X Window Systemet elindítjuk a `startx` paranccsal.

Ha az **XDM**-et használjuk bejelentkeztető képernyőként, a beállítást némileg máshogyan kell elvégezni. Ekkor az iménti helyett az `.xsession` állományt kell szerkeszteni. A **kdm**-re vonatkozó utasítások a fejezet későbbi részében találhatók meg.

5.7.3. A KDE-ről bővebben

Most, miután telepítettük a **KDE**-t a rendszerünkre, a dolgok többsége felfedezhető a különféle súgók segítségével vagy egyszerűen a menükre történő kattintással. A Windows-hoz vagy Mac®-hez szokott felhasználók itt most már egészen otthonosan érezhetik magukat.

A **KDE**-hez a legtöbb segítséget a saját internetes dokumentációjából nyerhetjük. A **KDE** a saját böngészőjét, a **Konqueror**t tartalmazza, valamint tucatnyi ügyes alkalmazást és temérdek mennyiségű dokumentációt. A szakasz további részeiben ezért inkább olyan problémákkal foglalkozunk, amelyek megoldásai céltalan kóborlással már nem fedezhetőek fel olyan egyszerűen.

5.7.3.1. A KDE bejelentkeztető képernyője

Egy többfelhasználós rendszer karbantartója minden bizonnyal szeretné üdvözölni rendszere felhasználóit egy grafikus bejelentkező képernyőn keresztül. A korábbiakban erre a célra az **XDM**-et javasoltuk. Azonban a **KDE** erre ajánl egy alternatívát, a **kdm**-et, amely jóval látványosabb és sokoldalúbb. Ez különösen abban merül ki, hogy a

felhasználók (egy menün keresztül) ki tudják választani a bejelentkezés után használni kívánt munkakörnyezetet (legyen az **KDE**, **GNOME** vagy bármi más).

A **kdm** használatához a **KDE** aktuális verziójától függően különböző állományokat kell szerkesztenünk.

KDE3 esetén a `/etc/ttys` állományban szereplő `ttv8` sort kell az alábbiak szerint módosítanunk:

```
ttv8 "/usr/local/bin/kdm -nodaemon" xterm on secure
```

KDE4 esetén a következő sorokat kell felvennünk az `/etc/rc.conf` állományba:

```
local_startup="{local_startup} /usr/local/kde4/etc/rc.d"
kdm4_enable="YES"
```

5.7.4. Az Xfce

5.7.4.1. Röviden az Xfce-ről

Az **Xfce** a **GNOME** által használt GTK+-ra épülő munkakörnyezet, amely azonban sokkal könnyedebb és azoknak készült, akik egy szimpla, hatékony, mindazonáltal könnyen használható és beállítható munkafelületre vágynak. Látvány szempontjából leginkább a kereskedelmi rendszereken megtalálható **CDE**-hez hasonlítható. Íme az **Xfce** néhány jellemzője:

- Egyszerű, könnyen kezelhető munkaasztal
- Tökéletesen konfigurálható egérrel, drag-and-droppal ("vonszolás") stb.
- A menükkal, kisalkalmazásokkal és alkalmazásindítókka tarkított főpanelje hasonló a **CDE** paneljéhez
- Beépített ablak-, állomány- és hangkezelővel, **GNOME** kompatibilitási modullal és még sok minden mással rendelkezik
- Használhatunk témákat (mivel GTK+-ra épül)
- Gyors, könnyű és hatékony: ideális régebbi vagy lassabb, esetleg kevés memóriával rendelkező számítógépekhez

Az **Xfce**-ről részletesebben az Xfce honlapján (<http://www.xfce.org/>) olvashatunk.

5.7.4.2. Az Xfce telepítése

Az **Xfce**-hez tartozik bináris csomag (legalábbis az leírás készítésének pillanatában). Ezt a következő módon tudjuk telepíteni:

```
# pkg_add -r xfce4
```

Vagy a Portgyűjtemény használatával forrásból is felrakhatjuk:

```
# cd /usr/ports/x11-wm/xfce4
# make install clean
```

Ezután világosítsuk fel az X szerveret, hogy a következő indulása során mi már az **Xfce**-t kívánjuk használni. Ehhez csak ennyit kell tennünk:

```
% echo "/usr/local/bin/startxfce4" > ~/.xinitrc
```

Így az X következő indításakor már az **Xfce** lesz a munkakörnyezetünk. Ahogy azt már korábban is jeleztük, az **XDM** használata során a GNOMEban leírtak szerint létre kell hoznunk az `.xsession` állományt, azonban ezúttal a `/usr/local/bin/startxfce4` parancs használatával. Vagy a `kdm`-ről szóló szakaszban tárgyaltak mentén beállíthatjuk úgy a bejelentkeztető képernyőt, hogy a bejelentkezés előtt válasszuk ki a munkakörnyezetet.

II. Gyakori feladatok

Miután az alapokat már átvettük, a FreeBSD kézikönyv következő része néhány gyakorta alkalmazott funkciót tárgyal. Az itt szereplő fejezetek:

- Bemutatnak különféle hasznos és népszerű asztali alkalmazásokat: böngészőket, irodai elősegítő eszközöket, dokumentum-megjelenítőket stb.
- Bemutatják a FreeBSD alatt is elérhető multimédia eszközöket.
- Kifejtik egy saját FreeBSD rendszermag elkészítésének folyamatát, amellyel így bővíteni tudjuk rendszerünk funkcionalitását.
- Részletesen bemutatják a nyomtatásért felelős alrendszert, asztali és hálózati nyomtatók használata esetén egyaránt.
- Megmutatják, hogyan futassunk Linuxra íródott alkalmazásokat a FreeBSD rendszerünkön.

Egyes fejezetek elolvasásához ajánlott bizonyos mértékű felkészülés, amely megemlítsre is kerül az érintett fejezetek áttekintésében.

Fejezet 6. Asztali alkalmazások

Írta: Christophe Juniet.

6.1. Áttekintés

A FreeBSD-n asztali alkalmazások széles spektrumát lehet futtatni, például böngészőket és szövegszerkesztőket. Legtöbbjük csomagként áll rendelkezésre, illetve automatizált módon lefordíthatóak a Portgyűjteményből. Az új felhasználók közül sokan szeretnének ilyen fajta alkalmazásokat használni, ezért ez a fejezet bemutatja, miként lehet a népszerűbb asztali alkalmazásokat minden különösebb erőfeszítés nélkül telepíteni, legyen szó az előre csomagolt vagy a Portgyűjteményben megtalálható formájukról.

Amikor portként telepítünk egy programot, lényegében a forráskódját fordítjuk le. Ez bizonyos esetekben nagyon sokáig is eltarthat attól függően, hogy pontosan mit is fordítunk le, illetve mekkora az erre a célra felhasznált számítógépünk vagy számítógépeink teljesítménye. Amennyiben a fordításra nem tudunk vagy nem kívánunk elegendő időt szánni, a Portgyűjteményben található programok többségét már előre lefordított csomagból is telepíthetjük.

Mivel a FreeBSD-ben bináris szintű Linux kompatibilitás is található, ezért az eredetileg Linuxra fejlesztett alkalmazások is használhatóak a munkakörnyezetünkben. Azonban határozottan javasoljuk, hogy a linuxos alkalmazások használatához először figyelmesen olvassuk át a 10 fejezetet. A linuxos bináris kompatibilitást használó portok neve általában a "linux-" előtaggal kezdődik, amit ne felejtünk el figyelembe venni, amikor például a `whereis(1)` segítségével keressük valamelyiket. A fejezet további részében feltételezzük, hogy a linuxos alkalmazások telepítése előtt aktiváltuk a bináris Linux kompatibilitást.

Íme a fejezetben tárgyalt kategóriák:

- Böngészők (mint a **Firefox**, **Opera**, **Konqueror**)
- Irodai eszközök (mint a **KOffice**, **AbiWord**, **The GIMP**, **OpenOffice.org**)
- Dokumentum-megjelenítők (mint az **Acrobat Reader®**, **gv**, **Xpdf**, **GQview**)
- Pénzügyi szoftverek (mint a **GnuCash**, **Gnumeric**, **Abacus**)

A fejezet elolvasásához ajánlott:

- a külső alkalmazások telepítésének ismerete (4 fejezet);
- linuxos alkalmazások telepítésének ismerete (10 fejezet).

a multimédiás környezet kialakítására vonatkozó információkért a 7 fejezetet érdemes elolvasni. Az elektronikus levelezés beállítását és használatát a 28 fejezetből tudhatjuk meg.

6.2. Böngészők

A FreeBSD-vel együtt nem települ semmilyen böngésző. Helyette keressük meg a Portgyűjteményben a `www` (<http://www.FreeBSD.org/ports/www.html>) könyvtárat, ahol ezzel szemben rengeteg böngésző áll telepítésre készen. Ha nem lenne időnk mindent lefordítani (ami egyes esetekben akár rengeteg időnkbe is kerülhet), ezek csomagolt formában is elérhetőek.

A **KDE**-hez és a **GNOME**-hoz eleve tartoznak HTML-böngészők. Ezen komplett munkakörnyezetek beállításához a 5.7 Szakaszt olvassuk el.

Ha viszont csak egy kevés erőforrást igénylő böngészőkre vágyunk, érdemes megnéznünk a Portgyűjteményben található `www/dillo2`, `www/links` vagy `www/w3m` portokat.

Ez a rész az alábbi alkalmazásokat említi:

Alkalmazás	Erőforrásigény	Telepítés forrásból	Főbb függőségek
Firefox	közepes	nehéz	Gtk+
Opera	kevés	könnyű	Vannak FreeBSD-s és linuxos változatai is. A linuxos verzió használatához azonban szükség van a bináris Linux kompatibilitásra és a linux-openmotif portra.
Konqueror	közepes	nehéz	A KDE függvénykönyvtárai.

6.2.1. Firefox

A **Firefox** egy modern, szabad és nyílt forráskódú böngésző, amely tökéletesen használható FreeBSD alatt. Megtalálható benne egy, a jelenlegi HTML szabványoknak nagyon jól megfelelő megjelenítő motor, a lapokra bontható böngészés támogatása, a kéretlenül felbukkanó ablakok blokkolása, különböző kiterjesztések, javított biztonsági lehetőségek és még sok minden más. A **Firefox** forrása a **Mozilla** kódján alapszik.

Csomagból így telepíthető:

```
# pkg_add -r firefox
```

Ekkor a **Firefox** 3.6 változata fog települni. Ha helyette a **Firefox** 3.5 változatát szeretnénk használni, akkor ezt a parancsot adjuk ki:

```
# pkg_add -r firefox35
```

Ha forrásból szeretnénk felrakni, használhatjuk a Portgyűjteményben található portját is:

```
# cd /usr/ports/www/firefox
# make install clean
```

A **Firefox** 3.5 telepítéséhez az iménti parancsban cseréljük ki a `firefox` részt a `firefox35` könyvtárra.

6.2.2. A Firefox és a Java™ plugin

Megjegyzés: Ennél és a következő résznél feltételezzük, hogy már korábban telepítettük a **Firefox** alkalmazást.

A Java™ plugin jelenleg nem működik a **Firefox** 3.6 változatával.

A FreeBSD Alapítvány megegyezett a Sun Microsystems-szel, hogy terjesztheti a Java futtatókörnyezet (JRE™) és a Java fejlesztőkörnyezet (JDK™) FreeBSD-re lefordított bináris változatait. Ezek a csomagok elérhetőek a FreeBSD Alapítvány (<http://www.freebsdoundation.org/downloads/java.shtml>) honlapjáról.

Ha tehát Java-támogatást szeretnénk hozzáadni a **Firefox** böngészőhöz, elsőként fel kell telepítenünk a `java/javavmwrapper` portot. Ezután le kell töltenünk a **Diablo JRE** csomagot a <http://www.freebsdoundation.org/downloads/java.shtml> címről, majd telepítenünk azt a `pkg_add(1)` segítségével.

Megjegyzés: Ezen az oldalon nem találunk bináris csomagokat FreeBSD 8.x rendszerekhez, azonban a FreeBSD 7.x rendszerekhez készült csomagok használhatóak 8.x esetén is. Ehhez mindössze a `misc/compat7x` portot kell előtte telepítenünk.

A másik lehetőség a **Diablo JRE** (valamint a **Diablo JDK**) telepítése a Portgyűjteményből (`java/diablo-jre16` és `java/diablo-jdk16`). Ehhez a szükséges forrásokat (distfile állományokat) licenelési megkötések miatt nekünk kell külön letölteni. A kapcsolódó utasításokat a `make maketarget` parancs meghívásával kaphatjuk meg.

Indítsuk el a böngészőnket, és írjuk be a címsorba, hogy `about:plugins` és nyomjuk le az **Enter** billentyűt. Az eredményül kapott oldalon láthatjuk az eddig telepített pluginok listáját, ahol mostanra már a **Java** pluginnak is meg kell jelennie. Amennyiben ez nem következne be, mindegyik felhasználónál adjuk ki az alábbi parancsot:

```
% ln -s /usr/local/diablo-jre1.6.0/plugin/i386/ns7/libjavaplugin_oji.so \
  $HOME/.mozilla/plugins/
```

Vagy ha a **Diablo JDK** csomagot telepítettük:

```
% ln -s /usr/local/diablo-jdk1.6.0/jre/plugin/i386/ns7/libjavaplugin_oji.so \
  $HOME/.mozilla/plugins/
```

Ezt követően indítsuk újra a böngészőnket.

Megjegyzés: Ezek a parancsok az i386 architektúra használatát feltételezik, de a csomagok az amd64 architektúra esetén is elérhetőek.

6.2.3. A Firefox és a Macromedia® Flash™ plugin

A Macromedia® Flash™ plugin nem érhető el közvetlenül FreeBSD-re. Azonban létezik egy, a plugin linuxos verziójára épített szoftveres réteg (wrapper). Ez a wrapper még többek közt az Adobe® Acrobat® és a RealPlayer® pluginjait is használhatóvá teszi.

Attól függően, hogy a FreeBSD melyik változatát használjuk, különböző módokon tudjuk működésbe hozni:

1. FreeBSD 7.x

Telepítsük a `www/nspluginwrapper` portot. Ehhez szükség lesz még az `emulators/linux_base-fc4` portra is, amely viszonylag nagy méretű.

Következő lépésként telepítsük a `www/linux-flashplugin9` portot. Ezáltal megkapjuk a Flash 9.x változatát, amely FreeBSD 7.x alatt remekül működik.

Megjegyzés: A 7.1-RELEASE előtti FreeBSD verziók esetén a `www/linux-flashplugin7` portot telepítsük és hagyjuk ki a `linprocfs(5)` használatára vonatkozó részt.

2. FreeBSD 8.x

Telepítsük a `www/nspluginwrapper` portot. Ehhez szükségünk lesz majd a valamivel nagyobb `emulators/linux_base-f10` portra.

Ezt követően telepítsük a `www/linux-f10-flashplugin10` portot. Ekkor a Flash 10.x változatát kapjuk, amely FreeBSD 8.x alatt remekül használható.

Ezen változat beüzemeléséhez még létre kell hoznunk az alábbi linket:

```
# ln -s /usr/local/lib/npapi/linux-f10-flashplugin/libflashplayer.so \
  /usr/local/lib/browser_plugins/
```

Miután a FreeBSD rendszerünk változatának megfelelően elvégeztük a Flash port telepítését, a plugint az egyes felhasználóknak a `nspluginwrapper` paranccsal tehetjük elérhetővé:

```
% nspluginwrapper -v -a -i
```

Ha Flash animációkat szeretnénk lejátszani, akkor ehhez a `/usr/compat/linux/proc` könyvtárba csatlakoztatnunk kell egy `linprocfs(5)` típusú linuxos `proc` állományrendszert. Ezt a következő paranccsal tehetjük meg:

```
# mount -t linprocfs linproc /usr/compat/linux/proc
```

Az `/etc/fstab` állományban az alábbi sor hozzáadásával azonban ennek csatlakoztatása akár automatikussá is tehető a rendszerindítás során:

```
linproc    /usr/compat/linux/proc  linprocfs    rw          0          0
```

Ezután indítsuk el a böngészőt, majd gépeljük be az `about:plugins` szöveget a címsorba és nyomjuk le az **Enter** billentyűt. Ekkor a jelenleg elérhető pluginok listájának kell megjelennie.

6.2.4. A Firefox és az Swfdec Flash plugin

Az Swfdec egy Flash animációk dekódolásáért és megjelenítéséért felelős programkönyvtár. Az Swfdec-Mozilla pedig egy **Firefox** böngészőkhöz készített plugin, amely az Swfdec könyvtáron keresztül játszik le SWF állományokat. Jelenleg még aktív fejlesztés alatt áll.

Ha nem akarjuk vagy netalán nem tudjuk forrásból lefordítani, akkor egyszerűen csak telepítsük csomagként a hálózaton keresztül:

```
# pkg_add -r swfdec-plugin
```

Ha valamiért mégsem érhető el hozzá csomag, akkor a Portgyűjteményből is telepíthetjük:

```
# cd /usr/ports/www/swfdec-plugin
# make install clean
```

Miután telepítettük a plugint, a használatához indítsuk újra a böngészőt.

6.2.5. Opera

Az **Opera** egy sokoldalú és szabványokkal kompatibilis böngésző. Tartalmaz beépített levelező klienst és hírolvasót, IRC-klienst, RSS/Atom-olvasót és még sok mindent mást. Ennek ellenére az **Opera** viszonylag pehelysúlyúnak és gyorsnak számít. Két fajta módon is használható: létezik “natív” FreeBSD-s változata, valamint a Linux emulációval futó változata.

Az **Opera** FreeBSD-s változatát a megfelelő csomag telepítésével érhetjük el:

```
# pkg_add -r opera
```

Habár egyes FTP oldalakon nem található meg az összes csomag, viszont a Portgyűjteményből még ekkor is be tudjuk szerezni az **Operat**:

```
# cd /usr/ports/www/opera
# make install clean
```

A linuxos **Opera** telepítéséhez `opera` helyett `linux-opera` nevet kell megadnunk a fenti parancsokban. Ennek a verzióknak a használata akkor lehet előnyös, ha olyan plugineket akarunk elérni, amelyek csak Linuxra léteznek. Ilyen például az **Adobe Acrobat Reader**. Ettől eltekintve azonban a FreeBSD-s és a linuxos változatok szinte teljesen megegyeznek.

6.2.6. Konqueror

A **Konqueror** a **KDE** része, de a használatához elegendő, ha csak a `x11/kdebase3` portot telepítjük fel. A **Konqueror** több, mint egy egyszerű böngésző: állománykezelő és multimédiás nézegető is.

Számtalan plugin áll rendelkezésre a **Konqueror**hoz, melyeket a `misc/konq-plugins` portban találunk meg.

A **Konqueror** ismeri a **Flasht** is. A **Flash** és a **Konqueror** kapcsolatával egy külön “Hogyan” is foglalkozik, amelyet a <http://freebsd.kde.org/howtos/konqueror-flash.php> címen olvashatunk el.

6.3. Irodai eszközök

Amikor irodai felhasználásról van szó, az új felhasználók gyakorta keresnek egy jó irodai programcsomagot vagy egy barátságos szövegszerkesztőt. Habár az egyes munkakörnyezetek, mint például a **KDE**, gyakran saját irodai eszközöket is tartalmaznak, FreeBSD alatt nincs alapértelmezett irodai programcsomag. A rendszer a munkakörnyezetektől függetlenül igyekszik felkínálni mindazt, amire szükségünk lehet.

Ebben a részben a következő alkalmazásokról esik szó:

Alkalmazás	Erőforrásigény	Telepítés forrásból	Főbb függőségek
KOffice	kevés	nehéz	KDE
AbiWord	kevés	könnyű	Gtk+ vagy GNOME
The Gimp	kevés	nehéz	Gtk+
OpenOffice.org	sok	nagyon nehéz	JDK, Mozilla

6.3.1. KOffice

A KDE közösség által kiadott munkakörnyezethez társul egy irodai programcsomag is, amely a **KDE**-től függetlenül is használható. Tartalmazza a többi irodai programcsomagban is megtalálható négy szabványos komponenst: a **KWord** szövegszerkesztőt, a **KSpread** táblázatkezelőt, a **KPresenter** prezentációkészítőt és végezetül a **Kontourt**, mellyel grafikus dokumentumokat tudunk elkészíteni.

A legfrissebb **KOffice** telepítése előtt bizonyosodjunk meg róla, hogy a **KDE** legfrissebb verziójával is rendelkezünk.

Ha a **KOffice**-t csomagként akarjuk telepíteni, akkor adjuk ki az alábbi parancsot:

```
# pkg_add -r koffice
```

Amennyiben ez a csomag nem érhető el, telepíthetjük a Portgyűjteményből is. Például a **KDE3**-hoz tartozó **KOffice**-t így rakhatjuk fel:

```
# cd /usr/ports/editors/koffice-kde3
# make install clean
```

6.3.2. AbiWord

Az **AbiWord** egy szabad szövegszerkesztő program, a **Microsoft Word**-höz hasonló kinézettel. Remekül használható levelek, beszámolók, feljegyzések, cikkek stb. írásához. Nagyon gyors, rengeteg funkciót ajánl fel, és kifejezetten felhasználóbarát.

Az **AbiWord** képes többféle állományformátumba exportálni és onnan importálni, beleértve az olyan zárt formátumokat is, mint például a Microsoft .doc.

Az **AbiWord** csomagból telepíthető a következő módon:

```
# pkg_add -r abiword
```

Amennyiben ez a csomag nem érhető el, lefordítható a Portgyűjteményből is, ami ráadásul sokszor egy frissebb verziót tartalmaz. Ezt így tudjuk megtenni:

```
# cd /usr/ports/editors/abiword
# make install clean
```

6.3.3. The GIMP

Képek készítésére vagy retusálásra a **The GIMP** a legfejlettebb képszerkesztő program. Egyszerű rajzolóprogram gyanánt is használható, de akár minőségi fényképretusálásra is. Óriási mennyiségű plugin található hozzá és magában foglal egy szkriptes interfészt is. A **The GIMP** formátumok széles skáláját ismeri. Számos scanner és digitális rajztábla csatlakoztatható hozzá.

A hozzá tartozó csomag a következő módon telepíthető fel:

```
# pkg_add -r gimp
```

Ha a csomagoknak beállított FTP oldalon nem található meg ez a csomag, megpróbálkozhatunk vele a Portgyűjteményen keresztül is. A gyűjtemény graphics (<http://www.FreeBSD.org/ports/graphics.html>) könyvtárában ezen felül fellelhetjük a **The Gimp Manualt**, vagyis a **The GIMP** kézikönyvét. Így kell ezeket innen telepíteni:

```
# cd /usr/ports/graphics/gimp
# make install clean
# cd /usr/ports/graphics/gimp-manual-pdf
# make install clean
```

Megjegyzés: A Portgyűjtemény graphics (<http://www.FreeBSD.org/ports/graphics.html>) könyvtárában a **The GIMP** fejlesztői változatával is találkozhatunk a `graphics/gimp-devel` alkönyvtárban. A **The Gimp Manual** HTML változata pedig a `graphics/gimp-manual-html` alkönyvtárban található.

6.3.4. OpenOffice.org

Az **OpenOffice.org** tartalmaz minden olyan elengedhetetlenül fontos alkalmazást, amelyek napjaink bármelyik irodájához hozzátartoznak: egy szövegszerkesztőt, egy táblázatkezelőt, egy prezentációszerkesztőt és egy rajzolóprogramot. A felhasználói felülete nagyon hasonlít a többi irodai programcsomagéhoz, és képes többféle elterjedt állományformátumot kezelni. Számos különböző nyelven elérhető — a honosítása kiterjed a felületekre, helyesírás-ellenőrzőkre és szótárakra is.

Az **OpenOffice.org** szövegszerkesztője natív XML állományformátumot használ a hordozhatóság és a rugalmasság növeléséhez. A táblázatkezelője tartalmaz egy makrónyelvet és könnyedén összekapcsolható külső adatbázisokkal. Az **OpenOffice.org** natívan és megbízhatóan fut Windows-on, SolarisTM-on, Linux-on, FreeBSD-n és Mac OS X-en. Az **OpenOffice.org**-ról bővebb információt a projekt saját honlapján (<http://www.openoffice.org/>) találhatunk. A FreeBSD-s változatra vonatkozó információkat és a csomagokat pedig a FreeBSD OpenOffice.org Porting Team (<http://porting.openoffice.org/freebsd/>) honlapján lelhetjük meg.

Az **OpenOffice.org** telepítéséhez ennyit kell csak beírni:

```
# pkg_add -r openoffice.org
```

Megjegyzés: Ha a FreeBSD -RELEASE ágát használjuk, ennek működnie kell. Ettől eltérő esetben érdemes egy pillantást vetni a FreeBSD **OpenOffice.org** Porting Team honlapjára, ahonnan le tudjuk tölteni a verziókhöz megfelelő csomagot, amelyet ezután a `pkg_add(1)`-al fel is tudunk telepíteni. A legfrissebb megbízható és a fejlesztői változat egyaránt elérhető erről a helyről.

Ahogy sikerült feltelepíteni a csomagot, egyszerűen csak be kell gépelni a következő parancsot az **OpenOffice.org** futtatásához:

```
% openoffice.org
```

Megjegyzés: Az első futtatás során válaszolnunk kell még néhány további kérdésre is, valamint a felhasználói könyvtárunkban keletkezik egy `.openoffice.org` könyvtár.

Ha nem érhetőek el **OpenOffice.org** csomagok, lefordíthatjuk a forrását is. Azonban mielőtt még ennek nekilátnánk, el kell fogadnunk, hogy ez a művelet a lemezünkön rettenetesen sok területet fog igényelni és meglehetősen sokáig tart.

```
# cd /usr/ports/editors/openoffice.org-3
# make install clean
```

Megjegyzés: Ha egy honosított verziót szeretnénk fordítani, az utolsó parancs helyett írjuk inkább ezt:

```
# make LOCALIZED_LANG=nyelv install clean
```

A `nyelv` helyett itt természetesen a nyelvnek megfelelő ISO-kódot kell megadni. Az itt támogatott nyelvek kódjának listája a port könyvtárán belül, a `files/Makefile.localized` állományban található meg.

Ahogy a fordítás befejeződött, az **OpenOffice.org** így indítható el parancssorból:

```
% openoffice.org
```

6.4. Dokumentum-megjelenítők

A UNIX megjelenése óta néhány új népszerű dokumentumformátum is felbukkant, melyek szabványos megjelenítői nem minden esetben részei az alaprendszernek. Ebben a részben azt tekintjük át, hogyan lehet ilyen megjelenítőket telepíteni.

Ez a rész az alábbi alkalmazásokat említi:

Alkalmazás	Erőforrásigény	Telepítés forrásból	Főbb függőségek
Acrobat Reader	kevés	könnyű	Bináris Linux kompatibilitás
gv	kevés	könnyű	Xaw3d
Xpdf	kevés	könnyű	FreeType
GQview	kevés	könnyű	Gtk+ vagy GNOME

6.4.1. Acrobat Reader®

A dokumentumok többsége manapság PDF (Portable Document Format, avagy “hordozható dokumentumformátum”) állományok formájában terjed. Az ilyen típusú állományok megnézésére az egyik legmegfelelőbb alkalmazás az **Acrobat Reader**, melyet az Adobe adott ki Linuxra. De mivel a FreeBSD képes Linux binárisok futtatására, ezért így FreeBSD-re is elérhető.

Ha az **Acrobat Reader 8**-at a Portgyűjteményből akarjuk telepíteni, akkor írjuk be:

```
# cd /usr/ports/print/acroread8
# make install clean
```

Licencelési megszorítások miatt csomag nem áll rendelkezésre.

6.4.2. gv

A **gv** egy PostScript és PDF megjelenítő. Eredetileg a **ghostview** alapján készült, de a **Xaw3d**-nek köszönhetően

sokkal szebben néz ki. Gyors és a felülete letisztult. A **gv** sok mindent tud, többek közt beállítható benne a dokumentum tájolása, a papírméret, skálázás és az élsimítás. Szinte bármelyik művelet elvégezhető csak billentyűzetről vagy egérrel.

A **gv** csomagjának telepítéséhez a következő parancsot használhatjuk:

```
# pkg_add -r gv
```

Ha pedig nem tudjuk letölteni a csomagot, használhatjuk a Portgyűjteményt is:

```
# cd /usr/ports/print/gv
# make install clean
```

6.4.3. Xpdf

Ha egy egyszerű FreeBSD-s PDF megjelenítőre lenne szükségünk, erre a célra az **Xpdf** pontosan megfelel. Nagyon kevés erőforrást igényel és nagyon megbízható. A szabványos X-beli betűtípusokat használja, és nincs szüksége sem a **Motifra**, sem pedig más X-es eszközkészletre.

Az **Xpdf** csomagjának felrakásához az alábbi parancs javasolt:

```
# pkg_add -r xpdf
```

Amennyiben nem áll rendelkezésre az említett csomag, vagy egyszerűen csak a Portgyűjteményből szeretnénk felrakni, adjuk ki ezeket a parancsokat:

```
# cd /usr/ports/graphics/xpdf
# make install clean
```

Ahogy a telepítés befejeződik, már el is indíthatjuk az **Xpdf** alkalmazást, ahol a jobb egérgombbal tudjuk aktiválni a menüt.

6.4.4. GQview

A **GQview** egy képkezelő. Állományokat tudunk megnyitni benne egyetlen kattintással, külső szerkesztőprogramot tudunk indítani vagy akár még a képek kicsinyített változatait is láthatjuk és így tovább. Megtalálható benne a diavetítés és az alapvető állományműveletek. Képgyűjteményeket is kezelhetünk és könnyedén megtalálhatjuk a bennük levő képek között az egyezőket. A **GQview** teljes képernyős nézegetést is megenged, illetve támogatja a honosítást.

A **GQview** csomag telepítéséhez ezt a parancsot kell kiadni:

```
# pkg_add -r gqview
```

Amikor ez a csomag nem tölthető le, vagy amikor inkább a Portgyűjteményből szeretnénk felrakni, ezt írjuk be:

```
# cd /usr/ports/graphics/gqview
# make install clean
```

6.5. Pénzügyi szoftverek

Ha bármilyen ok folytán a FreeBSD-vel szeretnénk kezelni személyes pénzügyeinket, akadnak olyan kellően komoly és könnyen kezelhető alkalmazások, amelyek csak a telepítésükre várnak. Néhány közülük kompatibilis az elterjedtebb állományformátumokkal, mint például amiben a **Quicken®** és az **Excel** is tárolja az adatait.

Ebben a részben az alábbi programokat vesszük sorra:

Alkalmazás	Erőforrásigény	Telepítés forrásból	Főbb függőségek
GnuCash	kevés	nehéz	GNOME
Gnumeric	kevés	nehéz	GNOME
Abacus	kevés	könnyű	Tcl/Tk
KMyMoney	kevés	nehéz	KDE

6.5.1. GnuCash

A **GnuCash** a **GNOME** része, és egy felhasználóbarát, mégis hatékony eszközt ad a felhasználók kezébe. A **GnuCash** segítségével nyilván tudjuk tartani a bevételeinket és kiadásainkat, bankszámláinkat és befektetéseinket. Felülete intuitív, miközben továbbra is professzionális minőségű.

A **GnuCash**-ben megtalálhatunk egy intelligens nyilvántartást, a számlák hierarchikus rendszerét, és számtalan billentyűkombinációt és automatikus kiegészítést, amivel felgyorsul a munkánk. Egyetlen tranzakciót képes felbontani több kisebb és részletesebb elemre. A **GnuCash** képes importálni és exportálni a **Quicken** QIF típusú állományait. Ezenkívül még kezeli a legtöbb nemzetközi dátumformátumot és pénznemet.

A **GnuCash**-t az alábbi módon tudjuk telepíteni a rendszerünkre:

```
# pkg_add -r gncash
```

Ha ez a csomag nem érhető el, használhatjuk a Portgyűjteményt is:

```
# cd /usr/ports/finance/gnucash
# make install clean
```

6.5.2. Gnumeric

A **Gnumeric** egy táblázatkezelő program, a **GNOME** munkakörnyezet része. Sok esetben képes a helyzethez alkalmazkodva automatikusan "kitalálni" a felhasználó gondolatait a cellák formátumának megfelelő automatikus kiegészítő rendszerével. Be tud olvasni számos népszerűbb formátumot, mint például az **Excel**, **Lotus 1-2-3** vagy a **Quattro Pro** állományait. A **math/guppi** grafikonkészítő programon keresztül támogatja grafikonok rajzolását is. Nagyszámú beépített funkcióval rendelkezik, és ismeri az összes megszokott cellaformátumot, legyen az szám, pénznem, dátum, idő vagy bármi más.

A **Gnumeric** telepítését az alábbi paranccsal adhatjuk ki:

```
# pkg_add -r gnumeric
```

Ha valamiért nem érhető el ez a csomag, a Portgyűjteményből is fel tudjuk rakni:

```
# cd /usr/ports/math/gnumeric
```

```
# make install clean
```

6.5.3. Abacus

Az **Abacus** egy kicsi és egyszerűen használható táblázatkezelő program. Számos olyan funkciót tartalmaz beépítve, amelyek kifejezetten hasznosnak bizonyulhatnak a statisztika, pénzügyek és a matematika területén. Importálni és exportálni tudja az **Excel** állományformátumát is. Az **Abacus** még PostScript formátumú kimenetet is tud készíteni.

Az **Abacus** telepítéséhez csupán ennyit kell tennünk:

```
# pkg_add -r abacus
```

Amennyiben viszont nem érhető el ez a csomag, használhatjuk a Portgyűjteményt is:

```
# cd /usr/ports/deskutils/abacus
# make install clean
```

6.5.4. KMyMoney

A **KMyMoney** a **KDE** részeként kifejlesztett személyi pénzügyi nyilvántartó. A **KMyMoney** igyekszik az összes kereskedelmi pénzügyi nyilvántartó programban megtalálható fontosabb lehetőséget magában foglalni és rendelkezésre bocsátani. Mindezek mellett egy könnyen használható és nagyon ügyes kettős könyvelést is találhatunk benne. A **KMyMoney** képes beolvasni a szabványos Quicken Interchange Format (QIF) szerint készült állományokat, követni a befektetéseket, többféle pénznemet kezelni és sokfajta kimutatást tudunk vele készíteni. A megfelelő bővítmény hozzáadásával még az OFX formátumú állományok olvasására is alkalmas.

A **KMyMoney** csomagként így telepíthető:

```
# pkg_add -r kmymoney2
```

Ha ez a csomag nem érhető el, akkor a Portgyűjteményen keresztül is fel tudjuk rakni:

```
# cd /usr/ports/finance/kmymoney2
# make install clean
```

6.6. Összefoglalás

Miközben a FreeBSD igen népszerű az internetszolgáltatók körében a teljesítménye és megbízhatósága révén, a hétköznapi használatban is remekül beválik. Többezernyi olyan alkalmazás érhető el hozzá csomag (<http://www.FreeBSD.org/where.html>)ként vagy port (<http://www.FreeBSD.org/ports/index.html>)ként, amelyekkel az igényeinknek megfelelő munkakörnyezetet tudjuk kiépíteni.

Íme egy rövidke emlékeztető azokról az asztali alkalmazásokról, melyeket a fejezetben tárgyaltunk:

Alkalmazás	Csomag	Port
Opera	opera	www/opera
Firefox	firefox	www/firefox

Alkalmazás

KOffice

AbiWord

The GIMP

OpenOffice.org

Acrobat Reader

gv

Xpdf

GQview

GnuCash

Gnumeric

Abacus

KMyMoney

Csomag

koffice-kde3

abiword

gimp

openoffice

acroread

gv

xpdf

gqview

gnucash

gnumeric

abacus

kmymoney2

Port

editors/koffice-kde3

editors/abiword

graphics/gimp

editors/openoffice.org-3

print/acroread8

print/gv

graphics/xpdf

graphics/gqview

finance/gnucash

math/gnumeric

deskutils/abacus

finance/kmymoney2

Fejezet 7. Multimédia

Szerkesztette: Ross Lippert.

7.1. Áttekintés

A FreeBSD a hangkártyák széles választékát ismeri, ami által képesek vagyunk számítógépünkkel hi-fi minőségű hangzást létrehozni. Ennek részeként rögzíteni és visszajátszani tudunk többek közt MPEG Audio Layer 3 (MP3), WAV és Ogg Vorbis formátumokban. A FreeBSD Portgyűjteménye ezenkívül tartalmaz még olyan alkalmazásokat is, amelyekkel szerkeszteni lehet a felvett hangokat, effekteket hozzátenni és vezérelni a hangkártyánkhoz csatlakoztatott MIDI eszközöket.

Némi kísérletezéssel a FreeBSD még videoállományok és DVD-k lejátszására is rávehető. A különféle videoanyagok kódolására, konvertálására és visszajátszására alkalmas programok száma azonban jóval kisebb, mint a hanganyagok esetén. Például az írás pillanatában nincs a FreeBSD Portgyűjteményében a formátumok közti konvertálásra alkalmas, a videókat olyan jól újrakódolni tudó alkalmazás, amilyen az audio esetén az `audio/sox`. Azonban ezen a területen a szoftverek palettája gyorsan változik.

Ebben a fejezetben bemutatjuk a hangkártyánk beállításához szükséges lépéseket. Az X11 telepítése és beállítása (5 fejezet) során ugyan már foglalkoztunk a videokártyánkkal kapcsolatos hardveres problémákkal, azonban a jobb visszajátszás érdekében további cselfogásokat is be kell majd vetnünk.

A fejezet elolvasása során megismerjük:

- hogyan állítsuk be úgy a rendszerünket, hogy felismerje a hangkártyánkat;
- hogyan bizonyosodjunk meg róla, hogy a kártyánk valóban működik;
- hogyan oldjuk meg a hangkártya beállítása során felmerülő problémákat;
- hogyan játsszunk le és kódoljunk MP3-at vagy más egyéb hangformátumot;
- hogyan támogatja a videókat az X szerver;
- hogyan adnak az egyes lejátszók és kódolók még jobb eredményt
- hogyan játsszunk le DVD-ket, `.mpg` és `.avi` állományokat;
- hogyan mentjük a CD-k és DVD-k tartalmát állományokba;
- hogyan állítsuk be a TV kártyánkat
- hogyan állítsunk be egy scannert.

A fejezet elolvasásához ajánlott:

- egy új rendszermag beállításának és telepítésének ismerete (8 fejezet).

FigyelemHa zenei CD-ket próbálunk meg a `mount(8)` paranccsal csatlakoztatni, akkor az hibával, vagy a legrosszabb esetben akár *teljes rendszerösszeomlással* is járhat. Az ilyen típusú lemezek az ISO szabványú állományrendszerektől eltérő kódolással rendelkeznek.

7.2. A hangkártya beállítása

Írta: Moses Moore. A FreeBSD 5.X verziójához igazította: Marc Fonvieille.

7.2.1. A rendszer beállítása

A művelet megkezdése előtt ki kell derítenünk, milyen típusú hangkártyánk van, milyen chip van rajta, PCI vagy ISA buszon csatlakozik-e. A FreeBSD rengeteg PCI és ISA buszos kártyát ismer egyaránt. A sajátunk beazonosításához a támogatott hangeszközök listáját a Hardware Notes (<http://www.FreeBSD.org/releases/8.1R/hardware.html>) (Hardverjegyzék) oldalán találhatjuk meg. Ebből a jegyzékből mellesleg azt is megtudhatjuk, hogy melyik meghajtó kezeli a kártyánkat.

A hangeszközünk használatához be kell töltenünk a neki megfelelő meghajtót. Ez két módon is megtehető. Ezek közül az a legkönnyebb, ha a `kldload(8)` paranccsal egyszerűen betöltjük a rendszermag hangkártyánkhoz tartozó modulját. Ezt megtehetjük közvetlenül parancssorból:

```
# kldload snd_emu10k1
```

vagy a `/boot/loader.conf` állományból az alábbihoz hasonló sor hozzáadásával:

```
snd_emu10k1_load="YES"
```

A fenti példák a Creative SoundBlaster® Live! hangkártyára vonatkoznak. A többi betölthető hangkártya-modul felsorolása a `/boot/defaults/loader.conf` állományban található. Ha nem vagyunk benne biztosak, hogy melyik meghajtót is akarjuk pontosan használni, akkor próbálkozzunk az `snd_driver` modul betöltésével:

```
# kldload snd_driver
```

Ez egy olyan metameghajtó, ami egyszerre betölti az összes érintett eszközmeghajtót, és segítségével felgyorsíthatjuk a megfelelő meghajtó megtalálását. A `/boot/loader.conf` használatával is be tudjuk ugyanígy tölteni az összes meghajtót.

Az `snd_driver` metameghajtó betöltése után úgy kereshetjük meg a ténylegesen használatban levő meghajtót, ha megnézzük a `/dev/sndstat` állományt a `cat /dev/sndstat` paranccsal.

A második módszer szerint a hangkártyánk támogatását statikusan beépítjük a rendszermagba. A lentebb található szakaszban olvashatjuk mindazok az információkat, amelyekre szükségünk lehet ennek elvégzése közben. A rendszermag újrafordításával kapcsolatban forduljunk a 8 fejezethez.

7.2.1.1. A hangkártya támogatásával rendelkező saját rendszermag összeállítása

Elsőként hozzá kell adnunk a rendszermaghoz a hangeszközök alapmeghajtóját, a `sound(4)` eszközt. Ezt a rendszermag beállításait tartalmazó állományban az alábbi sor felvételével tehetjük meg:

```
device sound
```

Ezután tegyük még hozzá a hangkártyánkhoz kapcsolódó támogatást is. Ehhez viszont pontosan tudunk kell, melyik meghajtó képes működtetni a kártyát. A hangkártyához tartozó meghajtót a Hardware Notes (<http://www.FreeBSD.org/releases/8.1R/hardware.html>) (Hardverjegyzék)-ben található eszközök listájából deríthetjük ki. Például a Creative SoundBlaster Live! hangkártyát a `snd_emu10k1(4)` meghajtó kezeli. Ennek a hangkártyának a támogatását az alábbi sorral állíthatjuk be:

```
device snd_emu10k1
```

Az itt használatos formátumot a meghajtó man oldalának átolvasásából tudhatjuk meg. Azonban az összes támogatott hangkártya meghajtó megadásának pontos formátuma megtalálható a `/usr/src/sys/conf/NOTES` állományban is.

A PnP (Plug n Play)-t nem ismerő ISA kártyák esetén az összes többi nem PnP-s ISA kártyához hasonlóan szükséges lehet a rendszermag számára megadnunk a kártya hardveres beállításait (IRQ, I/O port stb). Ezt a `/boot/device.hints` állományon keresztül tehetjük meg. A rendszerindítási folyamat során a loader(8) beolvassa ezt az állományt, majd átadja a benne szereplő információkat a rendszermagnak. Például a Creative SoundBlaster 16, nem PnP-s ISA kártya az `snd_sb16` meghajtóval együtt az `snd_sbc(4)` meghajtót használja. A kártya használatához a rendszermag beállításait tartalmazó állományba ezeket a sorokat kell megadni:

```
device snd_sbc
device snd_sb16
```

valamint a `/boot/device.hints` állományba ezeket:

```
hint.sbc.0.at="isa"
hint.sbc.0.port="0x220"
hint.sbc.0.irq="5"
hint.sbc.0.drq="1"
hint.sbc.0.flags="0x15"
```

Ekkor a kártya a 0x220 I/O portot és 5 IRQ-t használja.

A `/boot/device.hints` állományban alkalmazott felírási módról bővebben a `sound(4)`, valamint a kérdéses meghajtó man oldalán tájékozódhatunk.

A fentiekben bemutatott beállítások alapértelmezettek, néhány esetben azonban a kártyánknak megfelelően meg kell változtatnunk az IRQ és egyéb értékeket. Erről a kártyáról konkrétan a `snd_sbc(4)` man oldalon olvashatunk részletesebben.

7.2.2. A hangkártya kipróbálása

Miután újraindítottuk a számítógépünket a módosított rendszermaggal, vagy miután betöltöttük a szükséges modult, a hangkártyának valahogy így kell megjelennie a rendszerünk üzenetpufferében (`dmesg(8)`):

```
pcm0: <Intel ICH3 (82801CA)> port 0xdc80-0xdcbf,0xd800-0xd8ff irq 5 at device 31.5 on pci0
pcm0: [GIANT-LOCKED]
pcm0: <Cirrus Logic CS4205 AC97 Codec>
```

A hangkártyánk állapota a `/dev/sndstat` állományon keresztül ellenőrizhető:

```
# cat /dev/sndstat
FreeBSD Audio Driver (newpcm)
Installed devices:
pcm0: <Intel ICH3 (82801CA)> at io 0xd800, 0xdc80 irq 5 bufsz 16384
kld snd_ich (1p/2r/0v channels duplex default)
```

Ez a kírás rendszerenként eltérhet. Ha nem látunk semmilyen `pcm0` eszközt, akkor menjünk vissza és nézzük át újra, pontosan mit is csináltunk. Vizsgáljuk át a rendszermagunk beállításait tartalmazó állományt és győződjünk meg róla, hogy a megfelelő meghajtót adtuk meg. Az itt felmerülő gyakori gondokkal a 7.2.2.1 Szakasz foglalkozik.

Ha azonban minden remekül haladt, akkor most már van egy működő hangkártyánk. Ha rendesen összekapcsoltuk hangkártyánkat a CD- vagy DVD-meghajtónk audio csatlakozásával, akkor tegyünk egy CD-t a meghajtóba és kezdjük el játszani a `cdcontrol(1)` paranccsal:

```
% cdcontrol -f /dev/acd0 play 1
```

Az olyan alkalmazások, mint például az `audio/workman`, ehhez egy sokkal barátságosabb felületet nyújtanak. Az MP3 formátumú állományok meghallgatásához pedig minden bizonnyal jól fog jönni egy olyan alkalmazás is, mint például az `audio/mpg123`.

A kártyát úgy is tesztelhetjük, ha az alábbihoz hasonló módon adatokat küldünk a `/dev/dsp` állományba:

```
% cat állománynév > /dev/dsp
```

ahol az `állománynév` tetszőleges állomány neve lehet. A parancs hatására valamilyen zajt kell hallanunk, és ez egyben meg is erősíti, hogy a hangkártyánk működik.

A hangkártyánk csatornáinak jellemzőit a `mixer(8)` paranccsal állíthatjuk. Erről további részleteket a `mixer(8)` man oldalon olvashatunk.

7.2.2.1. Gyakori problémák

Hiba	Megoldás
<code>sb_dspwr(XX) timed out</code>	Nem állítottuk be jól az I/O portot.
<code>bad irq XX</code>	Nem állítottuk be jól az IRQ értékét. Gondoskodjunk róla, hogy a beállított érték megegyezik a hangkártyánkével.
<code>xxx: gus pcm not attached, out of memory</code>	Nincs elég memória az eszköz használatához.
<code>xxx: can't open /dev/dsp!</code>	A <code>fstat grep dsp</code> parancs kiadásával ellenőrizzük, hogy valamelyik alkalmazás használja-e már az eszközt. Gyakori bajkeverő az esound és a KDE hangtámogatása.

7.2.3. Több hangforrás kihasználása

Írta: Munish Chopra.

Gyakran szükségünk lehet több hangforrás egyidejű használatára, főleg olyankor, amikor az **esound** vagy az **artsd** bizonyos alkalmazásokkal nem hajlandó megosztani a hangeszközt.

A FreeBSD ezt a virtuális hangcsatornák használatával oldja meg, amit a `sysctl(8)` eszközön keresztül tudunk engedélyezni. Amikor a rendszermagban virtuális csatornák használatával keverünk, akkor lényegében képesek vagyunk a hangkártyánk által egyszerre játszható hangok számát megtöbbszörözni.

A virtuális csatornák számának beállításához a `sysctl` három változóját kell módosítanunk, amelyet `root` felhasználóként így tehetünk meg:

```
# sysctl dev.pcm.0.play.vchans=4
# sysctl dev.pcm.0.rec.vchans=4
```

```
# sysctl hw.snd.maxautovchans=4
```

A fenti példa négy virtuális csatornát hoz létre, ami egészen jellemző a mindennapi használatban. A `dev.pcm.0.play.vchans` és `dev.pcm.0.rec.vchans` a `pcm0` eszköz lejátszásra és felvételre használt virtuális csatornáinak számát adja meg, amelyet az eszköz csatlakoztatása után tudunk beállítani. A `hw.snd.maxautovchans` az új eszközhöz tartozó virtuális csatornákat adja meg, ami akkor állítódik be, amikor a `kldload(8)` paranccsal csatlakoztatjuk. Mivel a `pcm` modul a többi eszközmeghajtótól függetlenül töltődik be, ezért a `hw.snd.maxautovchans` azt tárolja, hogy a később hozzá csatlakozó eszközök mennyi virtuális csatornát fognak majd kapni. Erről részletesebben a `pcm(4)` man oldalon olvashatunk.

Megjegyzés: A használatban levő eszközöknél nem tudjuk megváltoztatni a virtuális csatornák számát. Ehhez először le kell állítanunk az eszközt használó összes programot, tehát a zenelejátszókat és hangdémonokat.

Amennyiben nem használjuk ki a `devfs(5)` által nyújtott lehetőségeket, az összes alkalmazásnak a `/dev/dsp0.x` eszközre kell mutatnia, ahol az `x` értéke 0-tól 3-ig terjedhet attól függően, hogy a `dev.pcm.0.rec.vchans` értékét a fenti példához hasonlóan 4-re állítottuk-e. A `devfs(5)` megoldását használó rendszerek esetén ez a folyamat automatikusan lezajlik, tehát az összes `/dev/dsp` eszközre irányuló kérés magától átirányítódik.

7.2.4. A keverő alapértelmezett értékeinek beállítása

Írta: Josef El-Rayes.

A keverőben megjelenő különböző csatornák alapértékei a `pcm(4)` meghajtó forráskódjában huzalozottan találhatók meg. Számos alkalmazás és démon segít két hívás közt megőrizni a keverőben beállított értékeket, azonban ez nem teljesen tiszta megoldás. A meghajtó szintjén is be tudjuk állítani a keverő alapértékeit — ezt a `/boot/device.hints` állomány megfelelő módosításával érhetjük el, például:

```
hint.pcm.0.vol="50"
```

Ezzel a `pcm(4)` modul betöltése során a hangerő (volume) csatorna alapértelmezett értéket 50-re állítjuk.

7.3. MP3

Írta: Chern Lee.

Az MP3 (MPEG Layer 3 Audio) használatával közel CD minőségű hangot lehet elérni, ezért a mi FreeBSD munkaállomásunk sem maradhat ki előnyeinek élvezetéből.

7.3.1. MP3 lejátszók

Az **XMMS** (X Multimedia System) kiemelkedően a legnépszerűbb X11-es MP3 lejátszó. Mivel az **XMMS** grafikus felhasználói felülete szinte teljesen megegyezik a Nullsoft **Winamp**-jának felületével, ezért még a **Winamp** skinjeit is használhatjuk vele. Az **XMMS**-ben ezenkívül még a natív pluginek támogatását is megtalálhatjuk.

Az **XMMS** a `multimedia/xmms` portból vagy csomagból telepíthető.

Az **XMMS** használatára könnyű ráérezni: megtaláljuk benne a lejátszandó számok listáját, egy grafikus hangszínszabályzót és még sok minden mást. Akik már ismerik a **Winamp** működését, azok az **XMMS**-t is egyszerűnek érzik majd.

Mellette az `audio/mpg123` port egy másik, parancssoros MP3 lejátszót kínál fel.

Az **mpg123** futtatásához paraméterként meg kell adnunk a hangeszközt és lejátszandó MP3 állományt. Ha a hangeszközünk a `/dev/dsp1.0` és a *IzéMizé-Slágér.mp3* nevű MP3 állományt akarjuk rajta lejátszatni, akkor a következőt kell begépelnünk:

```
# mpg123 -a /dev/dsp1.0 IzéMizé-Slágér.mp3
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2 and 3.
Version 0.59r (1999/Jun/15). Written and copyrights by Michael Hipp.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!
```

```
Playing MPEG stream from IzéMizé-Slágér.mp3 ...
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
```

7.3.2. Sávok lementése CD-ről

Mielőtt MP3 formátumba tömörítenénk egy CD-t vagy annak egy sávját, a CD-n található audio adatot valahogy le kell tudnunk szedni a merevlemezre. Ezt úgy tehetjük meg, ha a nyers CDDA (CD Digital Audio) adatot WAV formátumú állományokba mentjük.

A `sysutils/cdrtools` csomag részeként elérhető `cdda2wav` segédprogrammal tudjuk a CD-en levő audio és a hozzájuk tartozó egyéb információkat leszedni.

A meghajtóban levő CD teljes tartalmát (`root` felhasználóként) a következő parancs kiadásával lehet (sávonként) különálló WAV állományokba menteni:

```
# cdda2wav -D 0,1,0 -B
```

A **cdda2wav** ismeri az ATAPI (IDE) CD-meghajtókat, használatukhoz a SCSI egység sorszáma helyett az eszköz nevét kell megadni. Tehát például így szedjük le egy IDE-meghajtóról a 7. sávot:

```
# cdda2wav -D /dev/acd0 -t 7
```

A `-D 0,1,0` a `0,1,0` sorszámú SCSI eszközre utal, ami megfelel `cdrecord -scanbus` parancs eredményének.

Az egyes sávok lementéséhez a `-t` kapcsoló használható:

```
# cdda2wav -D 0,1,0 -t 7
```

A példa szerint a zenei CD-ről a hetedik sávot szedjük le. Egyszerre több sávot, például az elsőől a hetedikig, egy tartomány megadásával menthetünk le:

```
# cdda2wav -D 0,1,0 -t 1+7
```

A dd(1) segédprogram is használható ATAPI eszközökön levő hangsávok kimentéséhez. Ennek lehetőségéről részletesebben a 18.6.5 Szakaszban olvashatunk.

7.3.3. MP3 állományok tömörítése

Az MP3 állomány tömörítésére manapság a legtöbben a **lame** elnevezésű kódolót választják. A portfában a **lame** az audio/lame helyen található meg.

Az előbb kimentett WAV állományok felhasználásával az alábbi paranccsal tudjuk átalakítani a audio01.wav állományt audio01.mp3 állománnyá:

```
# lame -h -b 128 \
--tt "Izé dal címe" \
--ta "Izé-mizé előadó" \
--tl "Izé-mizé album" \
--ty "2001" \
--tc "Leszedte és tömörítette: Izé" \
--tg "Műfaj" \
audio01.wav audio01.mp3
```

A 128 kbites tömörítés a gyakorlatban leginkább használt kódolási arány, sokan azonban a sokkal jobb minőségű 160 vagy 192 kbites tömörítést szeretik. Minél nagyobb a kódolási arány, annál több helyet fog foglalni a keletkező MP3 állomány — habár a minősége is jobb lesz. A -h kapcsoló alkalmazásával tudjuk aktivizálni a “jobb minőségű de valamivel lassabb” módot. A --t kezdetű paraméterek ID3 tageket adnak meg, amelyek segítségével az MP3 állományokba rájuk vonatkozó információkat tudunk beágyazni. A tömörítés további beállításairól a **lame** man oldalán tájékozódhatunk.

7.3.4. MP3 állományok kitömörítése

Ha MP3 formátumú állományokat szeretnénk audio CD-re írni, akkor ehhez először tömörítetlen WAV formátumba kell ezeket alakítanunk. Az **XMMS** és az **mpg123** is egyaránt lehetőség ad az MP3 állományok kitömörítésére.

Lemezre írás az **XMMS**-sel:

1. Indítsuk el az **XMMS** alkalmazást.
2. Az **XMMS** menüjének felhozásához kattintsunk jobb gombbal az ablakjára.
3. Válasszuk az Options almenüben található Preference menüpontot.
4. Változtassuk meg az “Output Plugin” beállítást a “Disk Writer Plugin” értékre.
5. Nyomjunk a Configure gombra.
6. Írjuk be (vagy válasszuk ki a Browse gombbal) a könyvtárat, ahová majd a kitömörített állományok kerülnek.
7. Az eddig megszokottak szerint töltsük be az **XMMS**-be az MP3 állományt, állítsuk 100%-ra a hangerőt és kapcsoljuk ki a hangszínszabályzót (EQ, equalizer).
8. Nyomjuk le a Play gombot — úgy fog tűnni, mintha az **XMMS** játszaná az MP3 állományt, de nem hallunk semmit. Ekkor a tartalmát állományba menti.
9. Mikor befejeztük a kitömörítést, ne felejtjük el visszaállítani az “Output Plugin” értékét az alapértelmezettre.

Írás a szabványos kimenetre az **mpg123**-mal:

1. Futtassuk le a `mpg123 -s audio01.mp3 > audio01.pcm` parancsot.

Az **XMMS** az állományokat WAV formátumban írja, miközben az **mpg123** nyers PCM hangadatokat képez belőlük. A **cdrecord** használata során mind a két formátumból hozhatóak létre audio CD-k. A nyers PCM a `burncd(8)` programmal használható. Amikor WAV állományokkal dolgozunk, minden egyes sáv elején egy apró kattánást hallhatunk: ez a WAV állomány fejléce lesz. A (`audio/sox` portból vagy csomagból telepíthető) **SoX** segédprogrammal a WAV formátumú állományok fejléce pillanatok alatt eltávolítható:

```
% sox -t wav -r 44100 -s -w -c 2 track.wav track.raw
```

A CD-írók FreeBSD alatti használatával kapcsolatban olvassuk el a 18.6 Szakaszt.

7.4. Videók lejátszása

Írta: Ross Lippert.

A videolejátszás egy nagyon friss és gyorsan fejlődő alkalmazási terület. Legyünk türelmesek, ez nem minden fog annyira könnyen menni, mint a hangok esetében.

A kezdéshez nem árt tudnunk, hogy a videokártyánk milyen gyártmányú és milyen chipet használ. Míg az **Xorg** és az **XFree86** számos különféle videokártyát ismer, csupán töredékükkel lehet jó lejátszási teljesítményt előhozni. Az X11 futtatása közben az `xdpyinfo(1)` parancs kiadásával kérdezhetjük le az X szervertől a kártyánk használatával elérhető kiterjesztéseket.

Érdemes a kezünk ügyében tartani egy rövidke MPEG formátumú állományt, amellyel majd ki tudjuk próbálni a különféle lejátszókat és azok beállításait. Mivel egyes DVD lejátszók alapértelmezés szerint a `/dev/dvd` helyen keresik a lejátszandó DVD eszközt, vagy egyszerűen csak így írták meg ezeket, mindenképpen hasznos lehet, ha szimbolikus linkeket hozunk létre a megfelelő eszközökre:

```
# ln -sf /dev/acd0 /dev/dvd
# ln -sf /dev/acd0 /dev/rdvd
```

A `devfs(5)` működése miatt azonban ezek a kézzel létrehozott linkek az újraindítás után már nem maradnak meg. A szimbolikus linkeket a rendszer minden egyes indulásakor úgy tudjuk automatikusan létrehozni, hogyha az `/etc/devfs.conf` állományba felvesszük az alábbi sort:

```
link acd0 dvd
link acd0 rdvd
```

Emellett a DVD-k titkosításának feloldása, mely a DVD-meghajtók speciális funkcióit igényli, a DVD eszközökön írási jogot is igényel.

Az X11 osztott memóriát kezelő felületének gyorsításához javasolt néhány `sysctl(8)` változó értékének megnövelése is:

```
kern.ipc.shmmax=67108864
kern.ipc.shmall=32768
```

7.4.1. A megjelenítő képességeinek megállapítása

Több különböző úton lehet X11 alatt videókat nézni, de ennek tényleges módját igazából a rendelkezésre álló hardver határozza meg. Az itt leírt módszerek által kihozható minőség hardverenként eltérhet. Másodsorban a videók megjelenítése az X11-ben az utóbbi időben igen nagy hangsúlyt kapott, ezért az **Xorg** és az **XFree86** minden egyes változatával jelentősen javulhat a helyzet ezen a téren.

A videók megjelenítésére használt gyakori felületek:

1. X11: az X11 normális kimenete osztott memórián keresztül
2. XVideo: az X11 felületének kiterjesztése, ami tetszőleges X11 által kirajzolható objektum esetén támogat videót
3. SDL: a Simple Directmedia Layer
4. DGA: a Direct Graphics Access (közvetlen grafikus hozzáférés)
5. SVGAlib: alacsonyszintű konzolos grafikus réteg

7.4.1.1. XVideo

Az **Xorg** és az **XFree86 4.X** rendelkezik egy *XVideo* (avagy Xvideo, Xv, xv) elnevezésű kiterjesztéssel, amelyen keresztül egy speciális gyorsítás segítségével a kirajzolható objektumokban közvetlenül meg tudunk jeleníteni videókat. Ezzel a kiterjesztéssel még a gyengébb gépeken is nagyon jó minőségű lejátszást tudunk elérni.

A kiterjesztés működéséről az `xvinfo` parancs kiadásával győződhetünk meg:

```
% xvinfo
```

Ha a parancs eredménye ehhez hasonló, akkor a kártyánk támogatja az XVideo-t:

```
X-Video Extension version 2.2
screen #0
  Adaptor #0: "Savage Streams Engine"
    number of ports: 1
    port base: 43
    operations supported: PutImage
    supported visuals:
      depth 16, visualID 0x22
      depth 16, visualID 0x23
    number of attributes: 5
      "XV_COLORKEY" (range 0 to 16777215)
        client settable attribute
        client gettable attribute (current value is 2110)
      "XV_BRIGHTNESS" (range -128 to 127)
        client settable attribute
        client gettable attribute (current value is 0)
      "XV_CONTRAST" (range 0 to 255)
        client settable attribute
        client gettable attribute (current value is 128)
      "XV_SATURATION" (range 0 to 255)
        client settable attribute
        client gettable attribute (current value is 128)
      "XV_HUE" (range -180 to 180)
        client settable attribute
```

```

        client gettable attribute (current value is 0)
maximum XvImage size: 1024 x 1024
Number of image formats: 7
  id: 0x32595559 (YUY2)
    guid: 59555932-0000-0010-8000-00aa00389b71
    bits per pixel: 16
    number of planes: 1
    type: YUV (packed)
  id: 0x32315659 (YV12)
    guid: 59563132-0000-0010-8000-00aa00389b71
    bits per pixel: 12
    number of planes: 3
    type: YUV (planar)
  id: 0x30323449 (I420)
    guid: 49343230-0000-0010-8000-00aa00389b71
    bits per pixel: 12
    number of planes: 3
    type: YUV (planar)
  id: 0x36315652 (RV16)
    guid: 52563135-0000-0000-0000-000000000000
    bits per pixel: 16
    number of planes: 1
    type: RGB (packed)
    depth: 0
    red, green, blue masks: 0x1f, 0x3e0, 0x7c00
  id: 0x35315652 (RV15)
    guid: 52563136-0000-0000-0000-000000000000
    bits per pixel: 16
    number of planes: 1
    type: RGB (packed)
    depth: 0
    red, green, blue masks: 0x1f, 0x7e0, 0xf800
  id: 0x31313259 (Y211)
    guid: 59323131-0000-0010-8000-00aa00389b71
    bits per pixel: 6
    number of planes: 3
    type: YUV (packed)
  id: 0x0
    guid: 00000000-0000-0000-0000-000000000000
    bits per pixel: 0
    number of planes: 0
    type: RGB (packed)
    depth: 1
    red, green, blue masks: 0x0, 0x0, 0x0

```

Az XVideo nem mindegyik implementációjában vannak jelen a felsorolt formátumok (YUV2, YUV12 stb.), ami viszont néhány lejátszó számára akadályokat jelenthet.

Amennyiben viszont ezt látjuk:

```

X-Video Extension version 2.2
screen #0
no adaptors present

```

Akkor a kártyánk nem rendelkezik XVideo támogatással.

Ha az XVideo nem támogatott a kártyánk számára, akkor az csupán csak annyit jelent, hogy a gépünknek nehéz dolga lesz a videók megjelenítéséhez szükséges számítási kapacitás kiszolgáltatásában. Azonban a videokártyánktól és processzorunktól függően még így is kielégítő eredményt tudunk előcsalni. Ekkor viszont minden bizonnyal érdemes lesz átolvasnunk a 7.4.3 Szakaszban, miként tudjuk növelni a teljesítményét.

7.4.1.2. A Simple Directmedia Layer

A Simple Directmedia Layer, vagy SDL, eredetileg a Microsoft Windows, BeOS és UNIX közti hordozhatóságot szándékozta megvalósítani, aminek segítségével a hangot és grafikát hatékonyan használni tudó alkalmazások hozhatóak létre. Az SDL által nyújtott réteg a hardver olyan alacsony szintű absztrakcióját öleli fel, amely gyakran még az X11 felületénél is hatékonyabb.

Az SDL a `devel/sdl1.2` helyen található.

7.4.1.3. Direct Graphics Access (Közvetlen grafikus hozzáférés)

A közvetlen grafikus hozzáférés az X11 egy olyan kiterjesztése, ami lehetővé teszi a programok számára az X szerver megkerülését és így közvetlenül a videokártya memóriáját képesek elérni. Mivel a megosztás hatékony megvalósításához ez nagyban építkezik alacsony szintű leképzési műveletekre, ezért az ilyet használó programokat `root` felhasználóként kell futtatni.

A DGA kiterjesztés a `dga(1)` segítségével tesztelhető és mérhető. A `dga` parancs kiadása után minden billentyű lenyomására megváltoztatja a képernyőn látható színeket. A kilépéshez a `q` billentyűt kell lenyomni.

7.4.2. A videókkal foglalkozó portok és csomagok

Ebben a szakaszban a FreeBSD Portgyűjteményéből a videók lejátszására alkalmas programokat vesszük számba. A videolejátszás nagyon gyorsan fejlődő terület, ezért az itt említett különböző alkalmazások képességei az itt leírtaktól némileg eltérhetnek.

Először is fontos tisztában lennünk azzal, hogy számos FreeBSD-n futó videoalkalmazás eredetileg linuxos alkalmazásként indult, és közülük sokan még csak béta minőségűek. Íme a FreeBSD-n is megtalálható videocsomagokkal kapcsolatos néhány olyan gond, amivel esetleg összefuthatunk:

1. Az egyik alkalmazás nem képes visszajátszani olyan állományt, amit egy másik alkalmazás hozott létre.
2. Az alkalmazás nem képes visszajátszani a saját maga által készített állományokat.
3. Ugyanazon alkalmazás két különböző gépen, amikor mind a kettőn az adott konfigurációra fordítjuk le, ugyanazt az állományt másképpen játssza vissza.
4. Egy olyan látszólag egyértelmű szűrő, mint például a kép átméretezése, a hibás átméretező rutin miatt nagyon csúnya eredményt produkál.
5. Az alkalmazás gyakran elszáll.
6. A porthoz nem találjuk a dokumentációt, egyedül csak az interneten vagy a port work könyvtárában van.

Sok alkalmazás a "linuxizmus" jeleit is hordozza, vagyis gondok adódhatnak abból, hogy a szerzők az alkalmazások működtetéséhez a Linux rendszermag és a különféle terjesztésekben megtalálható módosított szabványos könyvtárak

különlegességeit használják ki. Ezeket a portok karbantartói nem mindig észlelik és javítják ki, ami miatt az alábbiak bármikor bekövetkezhetnek:

1. A processzor jellemzőit a `/proc/cpuinfo` állományon keresztül állapítják meg.
2. A szálak helytelen használatuk miatt a program befejeződésekor összeakadnak.
3. Az alkalmazással gyakran együtt használt egyéb alkalmazások még nem nincsenek benne a FreeBSD Portgyűjteményében.

Az ilyen alkalmazások fejlesztői a hordozhatóság javításával és a problémák megoldásával kapcsolatban eddig mindig igyekeztek együttműködni a portok karbantartóival.

7.4.2.1. MPlayer

Az **MPlayer** az utóbbi időben felbukkant, gyorsan fejlődő videolejátszó. Fejlesztőinek célja a sebesség és rugalmasság a Linux, illetve más UNIX rendszereken. A kezdeményezés abból fakadt, hogy a fejlesztés mögött álló csapat alapítójának elege lett az akkoriban elérhető lejátszók teljesítményéből. Mondhatnánk, hogy ez a program feláldozta a grafikus felületet az áramvonalas kialakításért, azonban ha hozzászokunk a parancssori beállításokhoz és a billentyűkön keresztüli vezérléshez, remekül működik.

7.4.2.1.1. Az MPlayer lefordítása

Az **MPlayer** a `multimedia/mplayer` helyen található. A program a fordítási folyamat során elvégez számos hardverellenőrzést, aminek eredményeképpen az egyik rendszeren fordított program nem vihető a másikra. Ezért különösen fontos portból fordítani és nem pedig bináris csomagot használni. Mindezek mellett a `Makefile` állományban még számos, a `make` parancsnak a fordítás megkezdésekor átadható beállítást találhatunk:

```
# cd /usr/ports/multimedia/mplayer
# make
N - O - T - E
```

```
Take a careful look into the Makefile in order
to learn how to tune mplayer towards you personal preferences!
For example,
make WITH_GTK1
builds MPlayer with GTK1-GUI support.
If you want to use the GUI, you can either install
/usr/ports/multimedia/mplayer-skins
or download official skin collections from
http://www.mplayerhq.hu/homepage/dload.html
```

Az üzenet fordítása:

```
F - I - G - U - R - E - M
```

Az `mplayer`t személyes igényeinkhez úgy tudjuk igazítani, ha figyelmesen átnézzük a `Makefile` állományt! Például a `WITH_GTK1` megadásával az `MPlayer` GTK1 alapú grafikus felülettel jön létre. A grafikus felület használatához telepítenünk kell a `/usr/ports/multimedia/mplayer-skins` portot is, vagy letölteni a hivatalos skingyűjteményt a <http://www.mplayerhq.hu/homepage/dload.html> oldalról.

A port alapbeállításai a legtöbb felhasználó számára megfelelőek, habár az Xvid kódok használatához meg kell adnunk a `WITH_XVID` beállítást. Rajta kívül még az alapértelmezett DVD eszközt is érdemes megadni a `WITH_DVD_DEVICE` beállítással, amelynek alapértéke a `/dev/acd0`.

A leírás elkészítésének időpontjában az **MPlayer** portja létrehozta a HTML dokumentációt és a két végrehajtható állományt: az `mplayer` lejátsszót és a videók újrakódolásáért felelős `mencoder` segédprogramot.

Az **MPlayer** HTML dokumentációja nagyon közlékeny, és ha az olvasó nem találná valamelyik videohardver vagy felület leírását ebben a fejezetben, akkor ez a dokumentáció mindenképpen hasznos olvasnivalónak bizonyul. Ha a UNIX-ok alatt elérhető videotámogatás leírását keressük, határozottan megéri időt szánni az **MPlayer** dokumentációjának alapos végigolvasására.

7.4.2.1.2. Az MPlayer használata

Az **MPlayer** használatához a felhasználói könyvtárunkban rendelkezünk kell egy `.mplayer` elnevezésű könyvtárral. Ezt a következő paranccsal tudjuk létrehozni:

```
% cd /usr/ports/multimedia/mplayer
% make install-user
```

Az `mplayer` parancssori paraméterei a hozzá tartozó man oldalon találhatók meg, valamint mindezek a HTML dokumentációban még részletesebben. Ebben a szakaszban csupán néhányukat mutatjuk be.

Egy állomány, mint például a `tesztvideo.avi`, a `-vo` beállításával játszható le a különböző felületeken:

```
% mplayer -vo xv tesztvideo.avi
% mplayer -vo sdl tesztvideo.avi
% mplayer -vo x11 tesztvideo.avi
# mplayer -vo dga tesztvideo.avi
# mplayer -vo 'sdl:dga' tesztvideo.avi
```

Érdemes az itt felsorolt konfigurációk mindegyikét kipróbálni, mivel az egymáshoz mért teljesítményük rengeteg tényezőn múlik, de közülük talán maga a hardver a legjelentősebb.

A DVD-k lejátszásához cseréljük ki a `tesztvideo.avi` paramétert a `dvd://N -dvd-device ESZKÖZ` paraméterekkel, ahol az `N` a lejátszandó fejezet sorszáma, valamint az `ESZKÖZ` a DVD-hez tartozó eszközeleíró. Például így tudjuk elkezdni `/dev/dvd` eszközeleíró a 3. fejezet lejátszását:

```
# mplayer -vo xv dvd://3 -dvd-device /dev/dvd
```

Megjegyzés: A port fordítása során a `WITH_DVD_DEVICE` paraméter segítségével megadható az alapértelmezett DVD eszköz, amely alpból a `/dev/acd0`. Erről többet a port `Makefile` állományában találhatunk.

A leállításához, szüneteltetéshez, továbblépéshez és többi hasonló funkcióhoz tartozó billentyűket a `mplayer -h` parancs kimenetéből vagy a man oldal elolvasásából deríthetjük ki.

A lejátszáshoz tartozó néhány viszonylag fontos beállítás: az `-fs -zoom` teljesképernyős módra vált, valamint a `-framedrop` segít növelni a teljesítményt.

A lejátszáskor kiadandó parancs túlburjánzását el tudjuk kerülni, ha létrehozunk egy `.mplayer/config` állományt és itt állítjuk be a gyakori opciókat:

```
vo=xv
fs=yes
zoom=yes
```

Végezetül megemlíti, hogy az `mplayer` segítségével a DVD-n található fejezeteket ki tudjuk menteni `.vob` állományokba. A DVD második fejezetének kimentéséhez gépeljük be ezt:

```
# mplayer -dumpstream -dumpfile out.vob dvd://2 -dvd-device /dev/dvd
```

A parancs eredményeképpen keletkező `out.vob` állomány formátuma MPEG lesz, amit a fejezetben bemutatott további csomagokkal tudunk feldolgozni.

7.4.2.1.3. mencoder

A `mencoder` használatának megkezdése előtt javasolt alaposan beleásnunk magunkat a HTML dokumentációba és megismerkednünk az alapvető beállításával. Van külön man oldala is, azonban a HTML leírás nélkül önmagában ez nem túl sokat ér. Megszámlálhatatlan úton és módon növelhető benne a minőség, csökkenthető a kódolási arány, változtatható a formátum, és ezen apró finomságok felelősek a jó vagy éppen a rossz teljesítményért. A témába néhány példa bemutatásával igyekszünk beavatni az olvasót. Először vegyünk egy egyszerű másolást:

```
% mencoder bemenő.avi -oac copy -ovc copy -o eredmény.avi
```

A parancssori paraméterek helytelen kombinációja olyan állományokat eredményezhet, amelyeket még maga az `mplayer` sem képes lejátszani. Ezért ha csak le akarunk szedni egy állományt, akkor maradjunk meg az `mplayer -dumpfile` opciójánál.

A `bemenő.avi` állományt MPEG4 video- és MPEG3 hangtömörítéssel (amihez kell majd a `audio/lame`) így tudjuk lekódolni:

```
% mencoder bemenő.avi -oac mp3lame -lameopts br=192 \
  -ovc lavc -lavcopts vcodec=mpeg4:vhq -o eredmény.avi
```

Ezzel az `mplayer` és `xine` programok számára is egyaránt lejátszható állomány jön létre.

A DVD fejezeteit úgy tudjuk közvetlenül kódolni, ha a parancssorban kicseréljük a `bemenő.avi` állományt az `dvd://1 -dvd-device /dev/dvd` beállításra, illetve ha a programot `root` felhasználóként futtatjuk. De mivel elsősorban általában ritkán vagyunk elégedettek a kódolással, érdemes először inkább lementeni az egész fejezetet egy állományba, majd azon dolgozni.

7.4.2.2. A xine videolejátszó

A **xine** egy széles hatókörű projekt, amelynek nem csak az a célja, hogy egy “mindenes” videolejátszó alkalmazást fejlesszenek, hanem az is, hogy újrahasznosítható függvénykönyvtárakat és egy moduláris felépítésű programot hozzanak létre, amely kiegészítőkkel bővíthető. A `multimedia/xine` helyen portként, valamint csomagként is elérhető.

A **xine** itt-ott még valamelyest durva, de mindenképpen egy dicséretes kezdeményezés. A **xine** a gyakorlatban erős processzort és mellé gyors videokártyát kíván, vagy pedig az XVideo kiterjesztés támogatását. A grafikus felhasználói felülete ugyan használható, de még kicsit esetlen.

Az írás pillanatában a **xine** mellé még nem kapunk olyan modult, amivel le tudnánk játszani a CSS kódolású DVD-eket. Léteznek azonban olyan külsős modulok, amelyekkel meg lehet valósítani ezt a feladatot, azonban a FreeBSD Portgyűjteményében ezeket még nem találhatjuk meg.

A **xine** az **MPlayer**hez képes többet tesz a felhasználóért, azonban ezzel egyidőben el is veszi tőle a finomhangolás lehetőségét. A **xine** legjobban az XVideo-t ismerő felületeken teljesít.

A **xine** alapértelmezés szerint grafikus felülettel indul, ahol a menük segítségével tudunk megnyitni egy adott állományt:

```
% xine
```

Vagy a grafikus felület használata nélkül kiadhatjuk közvetlenül is az állomány lejátszását:

```
% xine -g -p kedvenemozim.avi
```

7.4.2.3. A transcode

A **transcode** nem egy újabb lejátszó, hanem a video- és audio állományok újratömörítésére használható programok gyűjteménye. A **transcode** segítségével a szabványos be- és kimeneten keresztül parancssoros programokkal képesek vagyunk videoállományokat összefűzni, megjavítani.

A multimedia/transcode port fordítása során temérdek beállítást adhatunk meg, amelyek közül az alábbi parancsban foglaljuk össze az általunk javasolandókat:

```
# make WITH_OPTIMIZED_CFLAGS=yes WITH_LIBA52=yes WITH_LAME=yes WITH_OGG=yes \
WITH_MJPEG=yes -DWITH_XVID=yes
```

Ezek a beállítások a legtöbb felhasználó számára elegendőek.

A **transcode** képességeinek illusztrálásához lássunk egy példát, amiben megmutatjuk, hogyan kell egy DivX állományt PAL szabványú MPEG-1 formátumú (PAL VCD) állománnyá alakítani:

```
% transcode -i bemenő.avi -V --export_prof vcd-pal -o output_vcd
% mplex -f 1 -o eredmény_vcd.mpg eredmény_vcd.m1v eredmény_vcd.mpa
```

Az eredményül keletkező *eredmény_vcd.mpg* MPEG állomány akár már játszható is **MPlayer**rel. Ha az állományt kiírjuk egy írható CD-re, akkor ezzel video CD-t is létre tudunk hozni, amihez viszont szükségünk van mind a multimedia/vcdimager és sysutils/cdrdao programokra.

A **transcode** parancsnak van saját man oldala, azonban ehelyett a **transcode** wiki (<http://www.transcoding.org/cgi-bin/transcode>)ben érdemes inkább további információkat és példákat keresni.

7.4.3. Ajánlott olvasmányok

A FreeBSD-hez tartozó videoszoftverek nagyon gyorsan fejlődnek. Könnyen elképzelhető, hogy az imént tárgyalt problémák legközelebb hamarosan megoldódik. Addig viszont bárkinek, aki a legtöbbet szeretné kihozni a FreeBSD audio- és video lehetőségeiből, rengeteg leírás és dokumentáció elolvasása alapján kell

összezsírozni a különböző beállításokat, és csak néhány alkalmazás mellett érdemes kitaratnia. Ebben a szakaszban igyekszünk segíteni az olvasónak megtalálni az ilyen jellegű információkat.

Az MPlayer dokumentációja (<http://www.mplayerhq.hu/DOCS/>) szakmai szempontból igen közlékeny. Ezt mindenkinek érdemes elolvasnia, aki a későbbiekben magasabb szakmai szinten akar foglalkozni a UNIX-os videózással. Az **MPlayer** levelezési listája viszont alig tolerálja a dokumentációt rendesen el nem olvasó emberek kérdéseit, ezért minden egyes hiba bejelentése előtt lehetőleg rendesen nézzük át a dokumentáció odavágó részeit.

A xine HOGYAN (http://dvd.sourceforge.net/xine-howto/en_GB/html/howto.html) egyik külön fejezetében az összes lejátszó esetén érvényesíthető teljesítménynövelési módszereket mutat be.

Végül íme néhány ígéretes alkalmazás, amelyeket érdemes kipróbálnunk:

- Avifile (<http://avifile.sourceforge.net/>), ami egyben a `multimedia/avifile` port
- Ogle (<http://www.dtek.chalmers.se/groups/dvd/>), ami a `multimedia/ogle` port
- Xtheater (<http://xtheater.sourceforge.net/>)
- `multimedia/dvdauthor`, egy nyílt forráskódú DVD-tartalom szerkesztő

7.5. TV kártyák beállítása

Eredetileg írta: Josef El-Rayes. Kiegészítette, továbbfejlesztette: Marc Fonvieille.

7.5.1. Bevezetés

A TV kártyák segítségével kábeles vagy antennás televízióadásokat tudunk nézni a számítógépünkön. A legtöbbjük RCA vagy S-video bemenettel rendelkezik, valamint néhányukon még FM rádiókészülék is megtalálható.

A FreeBSD a bktr(4) meghajtón keresztül a Brooktree Bt848/849/878/879, illetve a Conexant CN-878/Fusion 878a típusú, PCI-os videorögzítő chipeket ismeri. Ügyelnünk kell arra, hogy a kártyánkon levő vevőkészülék is használható legyen, amit pedig a bktr(4) man oldalán megtalálható támogatott eszközök listájából ellenőrizhetünk.

7.5.2. A meghajtó beállítása

A kártyánk használatához be kell töltenünk a bktr(4) meghajtót, ami csupán annyiból áll, hogy a `/boot/loader.conf` állományhoz hozzáadunk egy ilyen sort:

```
bktr_load="YES"
```

Másik lehetőségünk, ha a TV kártya támogatását statikusan beleépítjük a rendszermagba. Ha ezt a megoldást választjuk, a következő sorokat kell elhelyeznünk a rendszermag beállításait tartalmazó állományba:

```
device bktr
device iicbus
device iicbb
device smbus
```

A fentebb látható egyéb eszközök megadása azért szükséges, mert a kártya részegységei egy I2C buszon csatlakoznak egymáshoz. Miután beillesztettük a szükséges változtatásokat, fordítsuk le és telepítsük az új rendszermagot.

A támogatás hozzáadása után újra kell indítanunk a számítógépünket. A rendszerindítási folyamat során meg kell jelennie a TV kártyánknak is, valahogy így:

```
bktr0: <BrookTree 848A> mem 0xd7000000-0xd7000fff irq 10 at device 10.0 on pci0
iicbb0: <I2C bit-banging driver> on bti2c0
iicbus0: <Philips I2C bus> on iicbb0 master-only
iicbus1: <Philips I2C bus> on iicbb0 master-only
smbus0: <System Management Bus> on bti2c0
bktr0: Pinnacle/Miro TV, Philips SECAM tuner.
```

Természetesen a fenti üzenetek az aktuális hardvereszközünknek megfelelően némileg eltérhetnek. Ellenőrizzük, hogy a vevőkészüléket helyesen ismerte-e fel a rendszer. Ha nem sikerült volna, akkor a sysctl(8) és a rendszermag beállításai segítségével még mindig van lehetőségünk állítani rajta. Például, ha egy Philips SECAM vevőkészüléket akarunk beállítani, akkor a rendszermag beállításaihoz még hozzá kell adni a következő sort:

```
options OVERRIDE_TUNER=6
```

vagy erre közvetlenül használhatjuk a sysctl(8) programot is:

```
# sysctl hw.bt848.tuner=6
```

A bktr(4) man oldalán és a /usr/src/sys/conf/NOTES állományban megtalálhatjuk a többi beállítás részletes leírását is.

7.5.3. Hasznos alkalmazások

A TV kártyánk tényleges használatához azonban még a következő alkalmazások valamelyikét is telepítenünk kell:

- A multimedia/fxrtv használatával ablakban tévézhetünk, valamint lehetőségünk van kép/audio/video kimentésére is.
- A multimedia/xawtv az **fxrtv**-hez hasonló lehetőségekkel bíró tévé néző alkalmazás.
- A misc/alevt dekódolja és megjeleníti a műsorhoz kapcsolódó Videotex/Teletext üzeneteket.
- Az audio/xmradio segítségével az egyes TV kártyákon megtalálható FM rádiókészülékeket tudjuk használatba venni.
- Az audio/wmtune a rádióvevőkhöz használható hasznos grafikus alkalmazás.

Ebben a témában a FreeBSD Portgyűjteményében további érdekes alkalmazások találhatók még.

7.5.4. Hibakeresés

Ha bármilyen gond adódna a TV kártyánkkal kapcsolatosan, akkor először mindenképpen érdemes megnézni, hogy a rajta levő videorögzítő chipet és vevőkészüléket a bktr(4) meghajtó ténylegesen ismeri-e, illetve hogy jól állítottuk-e be. A TV kártyákra irányuló különféle egyéb kérdések és segítség tekintetében érdemes lehet még levelet küldeni a freebsd-multimedia (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-multimedia>) címére is.

7.6. Lapolvasók

Írta: Marc Fonvieille.

7.6.1. Bevezetés

A FreeBSD lapolvasókhoz a **SANE** (Scanner Access Now Easy) elnevezésű API (alkalmazásfejlesztői felület) segítségével képes hozzáférni, amelyet a Portgyűjteményben találhatunk meg. A lapolvasást végző hardvereszközök használatához a FreeBSD a **SANE** mellett még néhány eszközmeghajtóra is támaszkodik.

A FreeBSD egyaránt ismeri az SCSI és USB csatlakoztatású lapolvasókat is. Még mielőtt nekikezdenénk a lapolvasó beállításához, bizonyosodjunk meg róla, hogy a **SANE** támogatja. A **SANE** által ismert eszközök (<http://www.sane-project.org/sane-supported-devices.html>) felsorolásában ellenőrizhetjük a lapolvasónk támogatottságának állapotát. A FreeBSD 8.x előtti kiadásában ezenkívül még a `uscanner(4)` man oldalon is láthatjuk az ismert USB-s lapolvasók listáját.

7.6.2. A rendszermag beállítása

A korábbiak értelmében tehát mind a SCSI, mind pedig a USB felületen csatlakozó eszközök támogatottak. A lapolvasónknak megfelelően eltérő eszközmeghajtók szükségesek.

7.6.2.1. Beállítás USB felületen

A **GENERIC** rendszermag alapértelmezés szerint tartalmazza az USB-s lapolvasók használatához szükséges eszközmeghajtókat. Ha valamilyen okból mégis saját rendszermagot akarunk használni, akkor ne felejtsük el ellenőrizni, hogy a rendszermag beállításai között megtalálhatóak a következő sorok:

```
device usb
device uhci
device ohci
device ehci
```

A FreeBSD 8.x előtti kiadásában még a következő sorra is szükségünk lesz:

```
device usscanner
```

A FreeBSD ezen változataiban a `uscanner(4)` eszközmeghajtón keresztül tudjuk használni az USB csatolóval rendelkező lapolvasókat. A FreeBSD 8.0 változatától kezdődően pedig ehhez a `libusb(3)` függvénykönyvtár nyújt közvetlen támogatást.

A megfelelően előkészített rendszermag elindítása után csatlakoztassuk az USB-s lapolvasónkat. Ez a sor fog megjelenni a rendszer üzenetpufferében (`dmesg(8)`):

```
ugen0.2: <EPSON> at usb0
```

Vagy FreeBSD 7.x rendszerek esetében:

```
uscanner0: EPSON EPSON Scanner, rev 1.10/3.02, addr 2
```

Ezek az üzenetek elárulják nekünk, hogy a lapolvasóhoz mostantól a használt FreeBSD verziótól függően a `/dev/ugen0.2` vagy a `/dev/uscanner0` eszközeleíró tartozik. A fenti példában egy EPSON Perfection® 1650 típusú USB lapolvasót láthatunk.

7.6.2.2. Beállítás SCSI felületen

Ha a lapolvasónk SCSI felületen csatlakozik, fontos tisztában lennünk azzal, hogy pontosan milyen SCSI-vezérlőn keresztül is érhetjük el, ugyanis a rajta található SCSI chipkészletnek megfelelően kell majd hangolnunk a rendszermag beállításait. A `GENERIC` rendszermag alpból ismeri a leggyakrabban előforduló SCSI-vezérlőket. Mindenképpen olvassuk át a `NOTES` nevű állományt és adjuk hozzá a rendszermag beállításaihoz a megfelelő sort. A SCSI-kártya meghajtóján kívül még az alábbi beállításokat is meg kell adnunk a rendszermagunk számára:

```
device scbus
device pass
```

Ahogy sikerült a rendszermagot sikeresen lefordítani és telepíteni, a rendszer indulása során az üzenetpufferben már láthatjuk is a felismert eszközt:

```
pass2 at aic0 bus 0 target 2 lun 0
pass2: <AGFA SNAPSCAN 600 1.10> Fixed Scanner SCSI-2 device
pass2: 3.300MB/s transfers
```

Ha a rendszer indulásakor még nem kapcsoltuk volna be a lapolvasónkat, a `camcontrol(8)` parancs segítségével később külön kérhetjük a SCSI buszon található eszközök újbóli felderítését:

```
# camcontrol rescan all
Re-scan of bus 0 was successful
Re-scan of bus 1 was successful
Re-scan of bus 2 was successful
Re-scan of bus 3 was successful
```

Ekkor a lapolvasó megjelenik a SCSI eszközök felsorolásában:

```
# camcontrol devlist
<IBM DDRS-34560 S97B>          at scbus0 target 5 lun 0 (pass0,da0)
<IBM DDRS-34560 S97B>          at scbus0 target 6 lun 0 (pass1,da1)
<AGFA SNAPSCAN 600 1.10>      at scbus1 target 2 lun 0 (pass3)
<PHILIPS CDD3610 CD-R/RW 1.00> at scbus2 target 0 lun 0 (pass2,cd0)
```

A SCSI eszközökről további leírásokat a `scsi(4)` és `camcontrol(8)` man oldalakon találhatunk.

7.6.3. A SANE beállítása

A **SANE** rendszere két részre oszlik: a backendekre (`graphics/sane-backends`) és a frontendekre (`graphics/sane-frontends`). Ezek közül maguk a backendek szolgáltatják a lapolvasó hozzáférhetőségét. A **SANE** által ismert eszközeinek (<http://www.sane-project.org/sane-supported-devices.html>) listájából kifürkészhetjük, hogy lapolvasónkat melyik backenden keresztül érhetjük el. Az eszköz megfelelő használatához döntő fontosságú megállapítani a hozzá tartozó backendet. A frontendek között találjuk meg a lapolvasást felügyelő grafikus felületeket (mint például az **xscanimage**).

Elsőként telepítsük a `graphics/sane-backends` portot vagy csomagot. Ezután ellenőrizzük, hogy a **SANE** felismeri a lapolvasót, és ehhez adjuk ki a `sane-find-scanner` parancsot:

```
# sane-find-scanner -q
found SCSI scanner "AGFA SNAPSCAN 600 1.10" at /dev/pass3
```

A kimenetében jelzi a felületet, amin a lapolvasó csatlakozik, valamint a hozzá tartozó eszközeírót. A gyártó neve és a termék típusa nem minden esetben jelenik meg, de ez nem is annyira fontos.

Megjegyzés: Némely USB-s lapolvasók esetén még egy firmware-t is be kell töltenünk, amiről bővebben a backendhez tartozó man oldalakon olvashatunk. Ajánlott még elolvasni a `sane-find-scanner(1)` és `sane(7)` man oldalakat is.

Most pedig nézzük meg, hogy vajon a frontend is be tudja-e azonosítani a lapolvasónkat. Alapértelmezés szerint a **SANE** backendjéhez tartozik még egy `scanimage(1)` nevű segédprogram is, aminek segítségével listázni tudjuk a használható eszközöket és képeket tudunk beolvasni parancssorból. Közülük a `-L` kapcsoló listáz:

```
# scanimage -L
device 'snapscan:/dev/pass3' is a AGFA SNAPSCAN 600 flatbed scanner
```

Vagy ha a 7.6.2.1 Szakaszban szereplő USB lapolvasóval nézzük:

```
# scanimage -L
device 'epson2:libusb:/dev/usb:/dev/ugen0.2' is a Epson GT-8200 flatbed scanner
```

Ezt a kimenetet egy FreeBSD 8.x rendszeren kaptuk, ahol a `'epson2:libusb:/dev/usb:/dev/ugen0.2'` az eszközhöz tartozó backendet (`epson2`) és eszközeírót (`/dev/ugen0.2`) adja meg.

Megjegyzés: Ha ennek eredményeképpen semmi sem jelenik meg, vagy a `scanimage(1)` látszólag nem talált semmilyen eszközt, akkor a lapolvasó azonosítása nem sikerült. Ilyen esetekben valószínűleg módosítanunk kell a backend beállításait tartalmazó állományt a használni kívánt lapolvasó eszköz szerint. A backendek beállításait a `/usr/local/etc/sane.d/` könyvtárban találjuk. Ez a probléma bizonyos USB-s lapolvasók esetében jelentkezik.

Például, ha 7.6.2.1 Szakaszban használt USB-s lapolvasónkat FreeBSD 8.x alatt tökéletesen felismeri a rendszer, de a FreeBSD korábbi változatai esetén (ahol a `usscanner(4)` eszközmeghajtót használják) a `sane-find-scanner` parancs a következőket adja vissza:

```
# sane-find-scanner -q
found USB scanner (UNKNOWN vendor and product) at device /dev/usscanner0
```

Akkor a lapolvasót sikerült megtalálni, és láthatjuk, hogy USB-n keresztül csatlakozik és a `/dev/usscanner0` eszközeíró tartozik hozzá. Most már ellenőrizhetjük a lapolvasó helyes beazonosítását is:

```
# scanimage -L
```

```
No scanners were identified. If you were expecting something different,
check that the scanner is plugged in, turned on and detected by the
sane-find-scanner tool (if appropriate). Please read the documentation
which came with this software (README, FAQ, manpages).
```

Az üzenet fordítása:

Nincs azonosítható lapolvasó. Ha nem erre számítottunk, akkor ellenőrizzük, hogy az eszközt tényleg bekapcsoltuk, csatlakoztattuk és észlelte a `sane-find-scanner` segédprogram (amennyiben szükséges). Kérjük, olvassa el a szoftverhez tartozó dokumentációt (README, FAQ, man oldalak)!

Mivel a lapolvasót nem sikerült azonosítani, át kell írunk a `/usr/local/etc/sane.d/epson2.conf` állományt. A használt lapolvasó típusa EPSON Perfection 1650, ezért hozzá az `epson2` backendet fogjuk használni. Ehhez feltétlenül olvassuk el a konfigurációs állományban található megjegyzéseket is. A sorokat igen könnyű átírni: tegyük megjegyzésbe az összes olyat, ahol a lapolvasónk számára nem megfelelő felületek találhatók (a mi esetünkben tehát megjegyzésbe fogjuk tenni az összes `scsi` szóval kezdődő sort, hiszen nekünk USB-s eszközünk van), majd az állomány végére írjuk be a használni kívánt felületet és eszközeírót. Ez ebben a konkrét esetben ennyi lenne:

```
usb /dev/usbscanner0
```

A megfelelő formátum és a további részletek leírásához ne felejtsek el azonban elolvasni a backend konfigurációs állományában felbukkanó megjegyzéseket és az ide tartozó man oldalt sem. Most már megpróbálkozhatunk újra a lapolvasó azonosításával:

```
# scanimage -L
device 'epson:/dev/usbscanner0' is a Epson GT-8200 flatbed scanner
```

Láthatjuk, hogy az USB-s lapolvasónkat sikerült azonosítani. Nem számít, ha esetleg nem egyezne a valósággal a gyártó vagy a típus megjelölése. Itt a valóban lényeges elem az `'epson:/dev/usbscanner0'` mező lesz, melynek a backend és az eszközeíró nevét kell helyesen tartalmaznia.

A beállítást akkor zárhatjuk le, miután a `scanimage -L` parancs képes észlelni a lapolvasót. A eszköz ekkor már készen áll a beolvasásra.

Míg a `scanimage(1)` parancssorból teszi lehetővé számunkra a lapolvasást, addig érdemesebb a képek olvasását egy grafikus felületen keresztül végeznünk. A **SANE** egy egyszerű, ám hatékony grafikus felületet ajánl fel ehhez, ez az **xscanimage** (`graphics/sane-frontends`).

Az **Xsane** (`graphics/xsane`) egy másik népszerű grafikus frontend. Segítségével speciális lehetőségeket is kihasználhatunk, mint például többféle képolvasási mód (fénymásoló, fax stb.), színkorrekció, kötegetelt beolvasás, stb. Mind a két említett alkalmazás elérhető a **The GIMP** bővítményeként is.

7.6.4. A lapolvasó használatának engedélyezése más felhasználók számára

A korábban tárgyalt műveletek mindegyikét `root` felhasználóként tudjuk csak végrehajtani. Azonban előfordulhat, hogy más felhasználók számára is szeretnénk hozzáférést biztosítani a lapolvasóhoz. Ehhez az érintett felhasználóknak a lapolvasóhoz tartozó eszközeíróhoz olvasási és írás joggal kell rendelkezniük. Például az USB-s lapolvasónk a `/dev/ugen0.2` eszközeírót használja, amely valójában csak a `/dev/usb/lp0.2.0` eszközeíróra mutató szimbolikus link (ezt gyorsan le tudjuk ellenőrizni, ha megnézzük a `/dev` könyvtár tartalmát). Az eszközeíró és a rá mutató szimbolikus link rendre a `wheel` és `operator` csoportok birtokában van. Ha a `pgj` nevű felhasználót felvesszük ezekbe a csoportokba, akkor ezáltal hozzá tud majd férni a lapolvasóhoz. Nyilvánvaló biztonsági megfontolásokból azonban kétszer is javasolt meggondolni, mely felhasználókat mely csoportokba vesszük fel, különösen, ha `wheel` csoportról van szó. Ennél valamivel jobb megoldást kínál, ha létrehozunk külön az USB eszközök használatára vonatkozó csoportot és a lapolvasót ezen csoport tagjainak számára elérhetővé tesszük.

Tehát erre a célra például megalkotjuk a `usb` csoportot. Ehhez első lépésként a `pw(8)` parancs segítségével hozzuk létre magát a csoportot:

```
# pw groupadd usb
```

Ezután a `/dev/usb/0.2.0` eszközeírót és a rá mutató `/dev/ugen0.2` szimbolikus linket kell az `usb` csoport részére elérhetővé tennünk, a megfelelő írási engedélyekkel (0660 vagy 0664) együtt, mivel alapértelmezetten csak a tulajdonosuk (`root`) tudja írni ezeket. Mindezt úgy tudjuk megtenni, ha az `/etc/devfs.rules` állományhoz hozzáadjuk a megfelelő sorokat:

```
[system=5]
add path ugen0.2 mode 0660 group usb
add path usb/0.2.0 mode 0660 group usb
```

A FreeBSD 7.x változatok esetén valószínűleg a következő sorokra lesz szükségünk a `/dev/uscanner0` eszközeíróhoz:

```
[system=5]
add path usscanner0 mode 0660 group usb
```

Ezt követően az `/etc/rc.conf` állományba írjuk be az alábbi sort és utána indítsuk újra a számítógépet:

```
devfs_system_ruleset="system"
```

Az itt szereplő sorok pontos jelentéséről a `devfs(8)` man oldaláról tájékozódhatunk.

Ezután már csak fel kell vennünk azokat a felhasználókat a `usb` csoportba, amelyeknek engedélyezzük a lapolvasó használatát:

```
# pw groupmod usb -m pgj
```

A további részletekről a `pw(8)` man oldalon olvashatunk.

Fejezet 8. A FreeBSD rendszermag testreszabása

Frissítette és átdolgozta: Jim Mock. Eredetileg írta: Jake Hamby.

8.1. Áttekintés

A rendszermag a FreeBSD operációs rendszer lelke. Felelős a memória kezelésért, a biztonsági szabályozások betartatásáért, a hálózat működtetéséért, a lemezhozzáférésért és sok minden másért is. Miközben maga a FreeBSD egyre jobban konfigurálható dinamikusan, addig alkalmanként elegendhetetlen, hogy újrakonfiguráljuk és újrafordítsuk a rendszermagot.

A fejezet elolvasása során megismerjük:

- miért lehet szükségünk egy saját rendszermagra;
- hogyan készítsünk konfigurációs állományt a rendszermaghoz, vagy hogyan módosítsunk egy már létezőt;
- hogyan használjuk a rendszermag konfigurációs állományát egy új rendszermag lefordítására és létrehozására;
- hogyan telepítsük az új rendszermagot;
- hogyan orvosoljuk a felmerülő problémákat.

A fejezetben az összes példaként bemutatásra kerülő parancsot `root` felhasználóként kell kiadni a sikeres végrehajtásukhoz.

8.2. Miért készítsünk saját rendszermagot?

A FreeBSD eredetileg ún. “monolitikus” rendszermaggal rendelkezett. Ez azt jelenti, hogy a rendszermag egyetlen nagy program volt, ami előre rögzített eszközöket ismert, és ha meg akartuk változtatni a rendszermag működését, akkor új rendszermagot kellett fordítanunk, majd újra kellett indítanunk vele a számítógépet.

Manapság azonban a FreeBSD már inkább afelé a megközelítés felé halad, ahol a rendszermag funkcionalitásának nagy részét működés közben az igények szerint betölthető és eltávolítható modulok adják. Ezzel lehetővé válik, hogy a rendszermag gyorsan illeszkedjen az újonnan megjelenő hardvereszközökhöz (mint például a laptopok PCMCIA-kártyáihoz), vagy olyan új funkciókat tegyünk a rendszermaghoz, amelyek a fordításánál nem voltak feltétlenül szükségesek. Ezt a modellt nevezik moduláris rendszermagnak.

Ennek ellenére még mindig elkerülhetetlen, hogy esetenként ne legyen szükség a rendszermag statikus testreszabására. Ez a legtöbb esetben azzal magyarázható, hogy vannak olyan funkciók, amelyek túlságosan is mélyen helyezkednek el a rendszermagban, ezáltal nem tölthetők be dinamikusan. Máskor viszont egyszerűen azért nem lehetséges, mert még senki sem szánt időt az adott funkcióhoz tartozó, dinamikusan betölthető modul elkészítésére.

Egy saját rendszermag készítése azon legfontosabb próbatételek egyike, melyet egy haladó BSD felhasználónak ki kell állnia. Ez a folyamat, habár némileg időigényes, számos előnyt tartogat FreeBSD rendszerünk számára. Eltérően egy `GENERIC` (általános) rendszermagtól, amely rengeteg hardvert támogat, egy saját rendszermag csak a *saját* PC-nk hardverét ismeri. Ennek több előnye is van, például:

- A rendszerünk gyorsabban indul. Mivel a rendszermag csak azokat a hardvereket fogja keresni, melyek a rendszerünkben megtalálhatóak, jelentős mértékben le tud csökkenni az induláshoz szükséges idő.
- Kisebb memóriahasználat. Egy saját rendszermag a szükségtelen részek és eszközmeghajtók elhagyása miatt gyakran kevesebb memóriát emészt fel, mint a GENERIC rendszermag. Ez azért is fontos, mert a rendszermag mindig benn van a fizikai memóriában, és ezzel az alkalmazások elől veszi el a helyet. Emiatt egy saját rendszermag elkészítése különösen hasznos lehet egy kevés fizikai memóriával rendelkező rendszeren.
- További hardverek támogatása. A saját rendszermagunkba olyan eszközök támogatását is beletehetjük, amelyek nem szerepelnek a GENERIC rendszermagban, mint például a hangkártyákat.

8.3. A rendszerünkben levő hardverek összeszedése

Írta: Tom Rhodes.

Mielőtt belevetnénk magunkat a rendszermag beállításába, érdemes egy leltárt készíteni a gépünkben található különböző eszközökről. Ahol a FreeBSD nem elsődlegesen használt operációs rendszer, ott ehhez elegendő megnézni a jelenlegi rendszerben található elemeket. Például a Microsoft rendszerek **Eszközkezelőjében** (Device Manager) általában az összes eszköz fontosabb adatait megtaláljuk. Magát az **Eszközkezelőt** pedig a Vezérlőpultból (Control Panel) érhetjük el.

Megjegyzés: A Microsoft Windows egyes verzióiban a **Rendszer** (System) ikonjára kattintva megkapjuk azt a képernyőt, ahonnan közvetlenül el tudjuk érni az **Eszközkezelőt**.

Ha viszont nincs másik operációs rendszer a gépünkön, akkor magunknak kell mindezeknek utánanéznünk. Erre az egyik alkalmas módszer a `dmesg(8)` és a `man(1)` parancsok használata. A FreeBSD-ben található legtöbb meghajtónak van saját `man` oldala, ami tartalmazza az általuk kezelt eszközök listáját, illetve így a rendszerindítás során észlelt hardvereket nézhetjük vissza. Például az alábbi sorok arra utalnak, hogy a `psm` meghajtó megtalálta a gépünkhöz tartozó egeret:

```
psm0: <PS/2 Mouse> irq 12 on atkdbc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD]
psm0: model Generic PS/2 mouse, device ID 0
```

Ezután ezt a meghajtót vagy a rendszermagba kell beépítenünk, vagy pedig a `loader.conf(5)` állományon keresztül betöltenünk.

Bizonyos esetekben a `dmesg` az eszközök felkutatásának eredményei helyett csak a rendszer üzeneteit mutatja. Ilyen helyzetekben a teljes kimenet a `/var/run/dmesg.boot` állományban tekinthető meg.

A hardverek manuális felderítésének módja a `pciconf(8)` segédprogram kimenetének böngészése, ami valamivel részletesebb eredményt ad. Mint például:

```
ath0@pci0:3:0:0:      class=0x020000 card=0x058a1014 chip=0x1014168c rev=0x01 hdr=0x00
    vendor      = 'Atheros Communications Inc.'
    device      = 'AR5212 Atheros AR5212 802.11abg wireless'
    class       = network
    subclass    = ethernet
```

A `pciconf -lv` paranccsal kapott kimenet ezen része azt mutatja, hogy az `ath` meghajtó talált egy vezeték nélküli Ethernet eszközt. Innen a `man ath` paranccsal érhetjük el a `ath(4)` man oldalát.

A `man(1)` a `-k` paraméter megadásával további hasznos információkkal is tud szolgálni. A fentiekből kiindulva például a következő paranccsal:

```
# man -k Atheros
```

le tudjuk kérdezni azokat a man oldalakat, amelyek tartalmazzák az adott szót:

```
ath(4)                - Atheros IEEE 802.11 wireless network driver
ath_hal(4)            - Atheros Hardware Access Layer (HAL)
```

A hardvereszközaink listájával felvértelve most már egy saját rendszermag létrehozása sem lesz annyira ijesztő.

8.4. Meghajtók, alrendszerek és modulok

Mielőtt új rendszermagot készítenénk, érdemes megfontolnunk, hogy egyáltalán szükségünk lesz-e rá. Ha például valamilyen eszköz támogatásához kell, akkor könnyen előfordulhat, hogy azt modulként is be tudjuk tölteni.

A rendszermaghoz tartozó modulok a `/boot/kernel` könyvtárban találhatóak, és a `kldload(8)` segítségével a rendszer működése közben dinamikusan betölthetőek. Ha nem is az összes, de a legtöbb meghajtóhoz tartozik egy modul és egy man oldal. Például az előző szakaszban az `ath` vezeték nélküli Ethernet meghajtóval foglalkoztunk. A következő leírást találjuk a hozzá tartozó man oldalon:

```
Vagy ha modulként akarjuk betölteni ezt a meghajtót a rendszer indítása
során, akkor a loader.conf(5) állományba vegyük fel a következő
sort:
```

```
if_ath_load="YES"
```

A fentebb leírtak szerint tehát, ha az `if_ath_load="YES"` sort hozzáadjuk a `/boot/loader.conf` állományhoz, akkor a rendszer indulásakor ez a modul mindig dinamikusan betöltődik.

Némely esetben azonban nem áll rendelkezésünkre ilyen modul. Ez különösen igaz bizonyos alrendszerekre és a fontosabb meghajtókra, például az FFS állományrendszerre vonatkozóan, mivel ezeknek kötelezően a rendszermagban kell lenniük. Ugyanez elmondható a hálózati támogatásra is (INET). Csak úgy tudjuk megmondani, hogy valamelyik meghajtóra szükség van a rendszermagban, ha először megpróbáljuk megkeresni hozzá a megfelelő modult.

FigyelemA beépített meghajtók figyelmetlen eltávolításával könnyen lefordíthatatlan állapotba kerülhet a rendszermag. Például, ha az `ata(4)` meghajtót kivesszük a rendszermag konfigurációs állományából, az ATA alrendszert használó meghajtók csak abban az esetben fognak biztosan működni, ha egyúttal felvesszük a `loader.conf` állományba. Ha nem vagyunk benne biztosak, akkor először próbáljuk meg használni a modult, és csak utána hagyjuk el a rendszermagba épített változatát.

8.5. Saját rendszermag készítése és telepítése

Először is tegyünk egy rövidke sétát a rendszermag könyvtárában. A továbbiakban említendő összes könyvtár a `/usr/src/sys` könyvtáron belül található, amely `/sys` néven is elérhető. Itt rengeteg alkönyvtár található, mindegyikük a rendszermag különböző részeit testesíti meg. Ezek közül most számunkra a legfontosabb az *architektúra/conf* lesz, ahol majd létrehozzuk a saját rendszermagunk konfigurációs állományát, valamint a *compile*, ahol majd a rendszermagunk fordítása történik. Itt az *architektúra* lehet *i386*, *alpha*, *amd64*, *ia64*, *powerpc*, *sparc64* vagy *pc98* (a PC-k egyik, leginkább Japánban elterjedt változata). Az adott architektúra könyvtárában található összes állomány csak arra az architektúrára vonatkozik, a kód többi része pedig gépfüggetlen és közös az összes többi létező és leendő FreeBSD platformon. Érdemes megfigyelni a könyvtárak logikai elrendezését: minden egyes ismert eszköz, állományrendszer és bővítmény saját alkönyvtárral rendelkezik.

A példák során ez a fejezet feltételezi, hogy az *i386* architektúrát használjuk. Ha ez a mi esetünkben nem így lenne, ne felejtjük el átírni bennük az elérési útvonalakat a rendszerünk architektúrájának megfelelően.

Megjegyzés: Ha *nem lenne* `/usr/src/sys` könyvtár a rendszerünkben, valószínűleg még nem telepítettük a rendszermag forráskódját. Ezt a legkönnyebben úgy tudjuk megtenni, ha `root` felhasználóként elindítjuk a `sysinstall` programot és ott kiválasztjuk a *Configure (Beállítások)*, azon belül *Distributions (Terjesztések)* menüpontot, amiben válasszuk ki a *src*, *base* és *sys* terjesztéseket. Ha nem szeretnénk erre a célra a **sysinstall** programot használni, de rendelkezésünkre áll a "hivatalos" FreeBSD CD, akkor a forrásokat akár parancssorból is telepíthetjük:

```
# mount /cdrom
# mkdir -p /usr/src/sys
# ln -s /usr/src/sys /sys
# cat /cdrom/src/ssys.[a-d]* | tar -xzf -
# cat /cdrom/src/sbase.[a-d]* | tar -xzf -
```

Ezután lépünk be az *i386/conf* könyvtárba és másoljuk le a *GENERIC* konfigurációs állományt a kedvünk szerinti névre. Például:

```
# cd /usr/src/sys/i386/conf
# cp GENERIC SAJÁT
```

Általában a nevet végig nagybetűkkel írjuk, és ha több FreeBSD-s gépet is üzemeltetünk különböző hardverekkel, hasznosnak bizonyulhat megemlíteni benne az adott gép rendszerének nevét is. Ebben a példában ez most a *SAJÁT* lesz.

Tipp: A rendszermagunk konfigurációs állományát nem éppen a legjobb ötlet a `/usr/src` könyvtárban tárolni. Ugyanis könnyen előfordulhat, hogy egy rosszul sikerült fordítás után egyszerűen csak letöröljük az egész `/usr/src` könyvtárat és onnan kezdjük újra. Azonban csak ezután juthat eszünkbe, hogy vele együtt bizony letöröltük a saját rendszermagunk konfigurációs állományát is! Ehhez hasonlóan, közvetlenül a *GENERIC* konfigurációs állomány szerkesztése sem ajánlott, mivel a források egy esetleges frissítésénél könnyen felülíródhat és ezzel együtt elvesznek a módosításaink is.

Tehát érdemes inkább valahol máshol tárolnunk a rendszermagunk konfigurációs állományát, majd létrehozni rá egy szimbolikus linket a *i386* könyvtárban.

Valahogy így:

```
# cd /usr/src/sys/i386/conf
# mkdir /root/kernel
```

```
# cp GENERIC /root/kernel/SAJÁT
# ln -s /root/kernel/SAJÁT
```

Most pedig a kedvenc szövegszerkesztőnkkel lássunk neki a *SAJÁT* átírásának! Ha nemrég telepítettük csak a rendszerünket, az egyetlen elérhető szövegszerkesztőnk minden bizonnyal a **vi** lesz. Róla most túlságosan is bonyolult lenne leírást adnunk, de az Irodalomjegyzékben található könyvek közül sokban elég jól bemutatják. Ezen kívül a FreeBSD ajánl egy könnyebben megtanulható szövegszerkesztőt is az **ee** személyében, amely a kezdők számára az ideális választás. Nyugodtan átírhatjuk az elől található megjegyzéseket a saját konfigurációnknak megfelelően, vagy akár azt is rögzíthetjük, hogy miben térünk el a **GENERIC** beállításaitól.

Ha fordítottunk már rendszermagot SunOS vagy más BSD operációs rendszer alatt, ez az állomány ismerősnek tűnhet. Ha viszont más operációs rendszerek, mint például a DOS felől érkezünk, a **GENERIC** konfigurációs állomány egy kissé terebélyesnek tűnhet számunkra, ezért A konfigurációs állomány című részt figyelmesen és lassan olvassuk át.

Megjegyzés: Amennyiben a forrásfáinkat a FreeBSD projekt legfrissebb forrásaival szinkronizáljuk, mindig olvassuk el a `/usr/src/UPDATING` állományt, mielőtt bármilyen frissítéshez is kezdenénk. Itt megtalálhatóak azok a fontos érintett kérdések és területek, amely külön figyelmet igényelnek a frissített forráskód esetén. A `/usr/src/UPDATING` mindig a FreeBSD forrásának legfrissebb változatához igazodik, és ezért sokkal naprakészebb információkat tartalmaz, mint ez a kézikönyv.

Most pedig le kell fordítanunk a rendszermag forráskódját.

A rendszermag lefordítása

1. Lépjünk be a `/usr/src` könyvtárba:

```
# cd /usr/src
```

2. Fordítsuk le a rendszermagot:

```
# make buildkernel KERNCONF=SAJÁT
```

3. Telepítsük az új rendszermagot:

```
# make installkernel KERNCONF=SAJÁT
```

Megjegyzés: A FreeBSD teljes forrásfájára szükség van a rendszermag lefordításához.

Tipp: Amikor egy saját rendszermagot alapértelmezés szerint fordítunk, vele együtt az összes modul is lefordításra kerül. Ha viszont időt szeretnénk megtakarítani a rendszermag frissítése során, vagy csak a saját moduljainkat akarjuk lefordítani, érdemes átírunk az `/etc/make.conf` állományt a rendszermag fordításának megkezdése előtt:

```
MODULES_OVERRIDE = linux acpi sound/sound sound/driver/dsl ntfs
```

Ez a változó megadja a ténylegesen lefordítandó modulok listáját.

```
WITHOUT_MODULES = linux acpi sound ntfs
```

Ez a változó a fordításból kihagyandó felső szintű modulokat sorolja fel. A rendszermag fordításának folyamatában egyéb hasznosnak tekinthető változókról a `make.conf(5)` man oldalán olvashatunk.

Ezután az új rendszermag a `/boot/kernel` könyvtárba kerül `/boot/kernel/kernel` néven, a korábbi rendszermag pedig `/boot/kernel.old/kernel` néven őrződik meg. Most állítsuk le a rendszert és indítsuk újra az új rendszermag aktiválásához. Ha közben valamilyen hiba történt volna, nézzük meg a fejezet végén található, hibakeresésre vonatkozó utasításokat. Mindenképpen olvassuk el azt a részt, amely leírja, hogyan állítsuk helyre a rendszerünket abban az esetben, ha az új rendszermaggal nem indul.

Megjegyzés: A rendszerindítási folyamathoz tartozó további állományok, mint például a rendszerbetöltő (loader(8)) és annak konfigurációs állománya, a `/boot` könyvtárban találhatóak. A külső és saját modulok a `/boot/kernel` a könyvtárba kerülhetnek, azonban a felhasználóknak nagyon ügyelniük kell rá, hogy az itt található modulok szinkronban legyenek a lefordított rendszermaggal. Ellenkező esetben a rendszerben megbízhatatlanságot, hibákat észlelhetünk.

8.6. A konfigurációs állomány

A FreeBSD 6.X verziójához igazította: Joel Dahl.

A konfigurációs állomány általános formátuma igen egyszerű. Minden sor tartalmaz egy kulcsszót és egy vagy több paramétert. A további egyszerűsítés kedvéért a legtöbb sor csak egyetlen paramétert tartalmaz. Bármely, ami egy `#` (kettőskereszt) jelet követ, megjegyzésnek minősül és nem számít konfigurációs elemnek. A most következő részek bemutatják az egyes kulcsszavakat abban a sorrendben, ahogy azokat a `GENERIC` állományban is megtalálhatjuk. Az architektúrafüggő opciók és eszközök teljes listáját a `GENERIC` állománnyal egy könyvtárban levő `NOTES` állományban találhatjuk meg. Az architektúrától független opciókat a `/usr/src/sys/conf/NOTES` állományban találjuk.

A FreeBSD 5.0 megjelenése óta a konfigurációs állományokban használható az `include` direktíva. Ennek segítségével egy másik konfigurációs állomány tartalma logikailag beilleszthető az aktuálisba, így könnyebbé válik egy már meglévő állományhoz tartozó kisebb mennyiségű változtatás karbantartása. Például ha csupán pár egyszerű kiegészítést szeretnénk hozzáadni a `GENERIC` rendszermaghoz, akkor elegendő a hozzá vett eltéréseket nyilvántartanunk egy külön konfigurációs állományban:

```
include GENERIC
ident SAJAT

options      IPFIREWALL
options      DUMMYNET
options      IPFIREWALL_DEFAULT_TO_ACCEPT
options      IPDIVERT
```

Valószínűleg sok rendszergazda számára jelentős előnyt jelent ez a megoldás a konfigurációs állományok korábbról már megszokott újraírásával szemben: a helyi konfigurációs állomány csak a `GENERIC` rendszermag helyi rendszerre vonatkozó eltéréseit tartalmazza. Így amikor frissítjük a rendszerünket, a `GENERIC` rendszermag összes újítása elérhetővé válik, kivéve ha explicit módon le nem tiltottuk ezeket a `noptions` vagy a `nodevice` megadásával. A

fejezet további részében egy átlagos konfigurációs állománnyal fogunk foglalkozni, mind a beállítások, mind pedig az eszközök tekintetében.

Megjegyzés: Ha olyan állományt akarunk készíteni, amely tartalmazza az összes lehetséges opciót, például teszteléshez, futtassuk le `root` felhasználóként az alábbi parancsot:

```
# cd /usr/src/sys/i386/conf && make LINT
```

Itt a `GENERIC` rendszermag-konfigurációs állomány ismertetése következik, az érthetőség kedvéért helyenként megjegyzésekkel kibővítve. A bemutatott állománynak majdnem pontosan meg kell egyeznie a rendszerünkben található `/usr/src/sys/i386/conf/GENERIC` állománnyal.

```
machine      i386
```

A számítógépünk architektúráját adja meg. A következők valamelyikének kell lennie: `alpha`, `amd64`, `i386`, `ia64`, `pc98`, `powerpc`, vagy `sparc64`.

```
cpu          I486_CPU
cpu          I586_CPU
cpu          I686_CPU
```

A fenti beállítás segítségével megadhatjuk, milyen típusú processzor található a számítógépünkben. Több ilyen sorunk is lehet (ha például nem lennének biztosak benne, hogy az `I586_CPU` vagy `I686_CPU` értéket kellene megadnunk), de a saját rendszermagunk összeállításához érdemes csak egyet meghagynunk. Ha nem ismerjük pontosan a processzorunk típusát, vessünk egy pillantást a `/var/run/dmesg.boot` állományra és keressük ki belőle.

```
ident        GENERIC
```

Ez a rendszermag azonosítója. Változtassuk meg rendszermagunk nevére, legyen például *SAJAT*, ha a korábbi utasításokat követtük. Az `ident` után írt sztring fog megjelenni a rendszermag neve mellett a rendszer indítása során, ezért fontos, hogy az új rendszermagunknak más nevet adjunk, ha meg akarjuk különböztetni az általában használttól (például egy tesztelésre szánt rendszermagot akarunk készíteni).

```
# ha a /boot/device.hints használata helyett statikusan bele akarjuk fordítani
#hints          "GENERIC.hints"          # itt szerepelnek a device hintek
```

A `device.hints(5)` használható az eszközmeghajtók beállítására. A `loader(8)` a rendszer indítása során alapértelmezés szerint a `/boot/device.hints` állományt olvassa be erre a célra. A `hints` beállítás használatával ezeket a "hinteket" statikusan bele tudjuk építeni a rendszermagba. Ebben az esetben nincs szükségünk külön `device.hints` állomány létrehozására a `/boot` könyvtárban.

```
makeoptions   DEBUG=-g          # a nyomkövetéshez szükséges gdb(1) szimbólumok beépítése
```

A FreeBSD hagyományos fordításának folyamata során a rendszermagot a `-g` használatával készítjük el, aminek köszönhetően hibakeresési információkat tudunk átadni a `gcc(1)` fordítónak.

```
options       SCHED_ULE          # ULE ütemező
```

A FreeBSD alapértelmezett rendszerütemezője. Ne változtassuk meg!

```
options          PREEMPTION          # a rendszerszálak megszakíthatóságának engedélyezése
```

Ha engedélyezzük, a rendszermagban futó szálakat meg tudják szakítani más, magasabb prioritású szálak. Ez segít növelni a rendszer válaszadási sebességét és csökkenti a megszakításokat kezelő szálak várakozását.

```
options          INET                 # hálózatkézelés
```

A hálózatkézelés támogatása. Ne töröljük ki, még akkor sem, ha nem tervezzük hálózatra kapcsolni a rendszert. Sok programnak szüksége van legalább az ún. loopback típusú hálózat támogatására (vagyis a számítógépünkön belüli hálózati kapcsolatokra), ezért ez feltétlenül kötelező!

```
options          INET6                # IPv6 kommunikációs protokollok
```

Engedélyezi az IPv6 kommunikációs protokollok használatát.

```
options          FFS                  # Berkeley Fast Filesystem
```

Ez a legalapvetőbb merevlemez állományrendszer. Hagyjuk meg, ha merevlemezről akarjuk indítani a rendszerünket.

```
options          SOFTUPDATES          # az FFS Soft Updates támogatása
```

Ez a beállítás engedélyezi a rendszermagban a Soft Updates használatát, amely segít felgyorsítani a lemez írási sebességét. Ha már a rendszermag ezt a funkcionalitást ismeri, akkor még külön az egyes lemezeken is engedélyezni kell. Nézzük meg a mount(8) kimenetét, hogy lássuk, a rendszerünkben levő lemezek közül melyiken van ténylegesen engedélyezve a Soft Updates használata. Ha nem látjuk benne sehol sem a soft-updates opciót, akkor azt (meglevő állományrendszerek esetén) a tuneufs(8) vagy (új állományrendszerek esetén) a newfs(8) parancsokkal tudjuk bekapcsolni.

```
options          UFS_ACL              # a hozzáférés-vezérlési listák (ACL) támogatása
```

Ezzel a beállítással engedélyezhetjük a rendszermagban a hozzáférés-vezérlési listák támogatását. Ez a kiterjesztett attribútumok és az UFS2 használatára támaszkodik. Ezt a lehetőséget részleteiben a 14.12 Szakaszban tárgyaljuk. Az ACL alapértelmezés szerint támogatott, és ha korábban már használtuk, akkor semmiképpen se kapcsoljuk ki, mert ezzel az eddig létrehozott hozzáférés-vezérlési listáink érvénytelenné, az állományaink pedig védtelenné válnak.

```
options          UFS_DIRHASH          # nagyobb könyvtárak esetén gyorsulást hoz
```

Ezzel a beállítással némi memória feláldozása árán fel tudjuk gyorsítani a nagyobb könyvtárakon végzett lemezműveletek sebességét, ezért ezt a beállítást érdemes nagyobb szerverekre vagy interaktivitást igénylő munkaállomásokra tartogatni, és eltávolítani olyan esetekben, amikor a FreeBSD-t olyan kisebb számítógépeken használjuk, ahol a memória kevés és a lemezműveletek sebessége kevésbé fontos, például egy tűzfalon.

```
options          MD_ROOT              # tudunk memórialemezeiről is rendszert indítani
```

Ezzel az opcióval engedélyezni tudjuk a rendszer indítását memóriában tárolt virtuális lemezekről.

```
options          NFSCLIENT            # hálózati állományrendszer (NFS) kliens
```

```
options          NFSSERVER            # NFS szerver
```

```
options          NFS_ROOT              # NFS használható gyökérként is, kell hozzá az NFSCLIENT
```

A hálózati állományrendszer támogatása. Hacsak nem akarunk TCP/IP-n keresztül állományrendszereket csatlakoztatni egy UNIX állományszerverről, kivehetjük.

```
options          MSDOSFS          # MS-DOS állományrendszer
```

Az MS-DOS állományrendszer. Hacsak nem akarunk DOS-ra formázott merevlemez partíciót csatlakoztatni a rendszerindítás során, nyugodtan elhagyhatjuk. A fentebb leírtak szerint az első olyan alkalommal automatikusan betöltődik, amikor egy DOS partíciót csatlakoztatni akarunk. Sőt, a nagyszerű `emulators/mtools` szoftver segítségével külön csatlakoztatás és leválasztás nélkül tudunk DOS-os floppykat olvasni (és az MSDOSFS-re egyáltalán nincs is szüksége).

```
options          CD9660           # ISO 9660 állományrendszer
```

Az ISO 9660 állományrendszert a CD-k használják. Vegyük ki, ha nincs a számítógépben CD-ROM meghajtó, vagy csak ritkán fogunk CD-ket csatlakoztatni (mivel a hozzá tartozó modul magától betöltődik az első adat CD csatlakoztatása során). Az audio CD-k nem használják ezt az állományrendszert.

```
options          PROCFS           # a futó programok állományrendszere (szükséges hozzá a PSEUDOFS)
```

A futó programok állományrendszere. Ez csak a `/proc` könyvtárra csatlakoztatott "színtelt" állományrendszer, amelynek segítségével a `ps(1)` és hozzá hasonló programok képesek több információt adni a futó programokról. A PROCFS használata a legtöbb esetben nem indokolt, mivel a különféle nyomkövető és felügyeleti eszközök képesek a PROCFS használata nélkül is működni: alapértelmezés szerint a telepített rendszerek sem csatlakoztatják ezt az állományrendszert.

```
options          PSEUDofs         # pseudo állományrendszerek támogatása
```

A 6.X verziójú rendszermagokban a PROCFS használatához engedélyeznünk kell a PSEUDofs használatát is.

```
options          GEOM_GPT         # GUID típusú partíciós táblák használata
```

Ezzel a beállítással engedélyezni tudjuk nagy mennyiségű partíció támogatását egyetlen lemezen.

```
options          COMPAT_43        # kompatibilitás fenntartása a 4.3 BSD-vel [NE TÖRÖLD!]
```

Kompatibilitás a 4.3BSD-vel. Ne vegyük ki, mert bizonyos programok furcsán fognak viselkedni a hiánya esetén.

```
options          COMPAT_FREEBSD4  # kompatibilitás a FreeBSD4-el
```

Ez a beállítás szükséges a FreeBSD 5.X i386 és Alpha rendszerein a FreeBSD korábbi verzióihoz fordított alkalmazások támogatásához, melyek régebbi rendszerhívásokat használnak. Az összes i386 és Alpha típusú rendszeren ajánlott engedélyezni, mivel itt előfordulhatnak régebbi alkalmazások. A többi platform, mint például az ia64 vagy a SPARC64, támogatása csak az 5.X verzióban jelent meg, ezért ott nincs szükség erre.

```
options          COMPAT_FREEBSD5  # kompatibilitás a FreeBSD5-el
```

Ezt a beállítást a FreeBSD 6.X és afeletti verziókban kell használni az olyan FreeBSD 5.X verziókra fordított alkalmazások futtatásának támogatásához, melyek a FreeBSD 5.X rendszerhívásait használják.

```
options          SCSI_DELAY=5000  # a SCSI eszközök keresése előtt késleltetés (ezredmásodpercben)
```

Ezzel a beállítással a rendszermag 5 másodpercig várakozni fog a SCSI eszközök keresése előtt. Ha kizárólag csak IDE típusú merevlemezeink vannak, nyugodtan kihagyhatjuk, máskülönben érdemes a rendszerindítás gyorsítása érdekében csökkenteni ezt az értéket. Természetesen, ha így teszünk és a FreeBSD nem tudja felismerni a SCSI eszközeinket, akkor növeljük meg valamennyivel.

```
options          KTRACE          # a ktrace(1) támogatása
```

Engedélyezi a rendszermagban futó rutinok nyomonkövetését, ami hasznos lehet a hibák keresése során.

```
options          SYSVSHM          # SYSV-szerű osztott memória
```

Ezzel a beállítással engedélyezni tudjuk a rendszerben a System V típusú osztott memória használatát. Leggyakrabban az X rendszer XSHM kiterjesztése használja, amelyen keresztül számos műveletigényes grafikus program működését fel lehet gyorsítani. Ha X-et használunk, mindenképpen szükségünk lehet erre.

```
options          SYSVMSG          # SYSV-szerű üzenetsorok
```

A System V üzenetek támogatása. Ez a beállítás csupán néhány száz byte-tal növeli a rendszermagot.

```
options          SYSVSEM          # SYSV-szerű szemaforok
```

A System V szemaforok támogatása. Nem túl gyakran alkalmazzák ezeket, de ez csak néhány száz byte-ot tesz hozzá a rendszermaghoz.

Megjegyzés: A `ipcs(1)` parancs `-p` paraméterével ki tudjuk listáztatni azokat a futó programokat, amelyek ezen System V eszközöket használják.

```
options          _KPOSIX_PRIORITY_SCHEDULING # POSIX P1003_1B valósidejű kiterjesztések
```

A POSIX® 1993-as változatában megjelent valósidejű bővítések. A Portgyűjteményben megjelenő egyes alkalmazások használják ezeket (mint például a **StarOffice**).

```
options          KBD_INSTALL_CDEV # CDEV bejegyzés létrehozása a /dev könyvtárban
```

Ez a beállítás kell ahhoz, hogy a `/dev` könyvtárban létre tudjunk hozni eszközleírókat a billentyűzethez.

```
options          ADAPTIVE_GIANT    # adaptív Giant mutexek
```

A Giant annak a kölcsönös kizárási mechanizmusnak (blokkolt mutexnek) a neve, amely a rendszermag erőforrásainak jelentős részét védi. Manapság ez már egy elfogadhatatlanul szűk keresztmetszetet képez a teljesítményben, ezért a fejlesztésben fokozatosan felváltják az egyes erőforrásokat külön-külön védő zárolások. Az `ADAPTIVE_GIANT` beállítás hatására a Giant a helyzethez igazodóan forgó (spin) mutexek közé kerül. Ez azt jelenti, hogy amikor egy szál zárolni akarja a Giant mutexet, de ezt már megtette előtte egy másik processzorról futó szál, a szál tovább fut és várakozni fog a zárolás feloldására. Normális esetben ugyanis egy szál továbbra is blokkolt állapotban marad, várakozva a futásra. Ha nem tudunk dönteni, hagyjuk változatlanul.

Megjegyzés: Hozzáteesszük, hogy a FreeBSD 8.0-CURRENT és későbbi változataiban az összes mutex alapértelmezés szerint adaptív, hacsak meg nem adjuk a `NO_ADAPTIVE_MUTEXES` beállítást. Ennek eredményeképpen a Giant most már alapból adaptív, ezért esetükben az `ADAPTIVE_GIANT` nem szerepel a rendszermag beállításai között.

```
device           apic             # I/O APIC
```

Az `apic` nevű eszköz engedélyezésével használhatjuk a hardveres APIC-ot a megszakítások vezérlésére. Az `apic` alkalmazható egy- és többprocesszoros rendszerek esetén is egyaránt, de az SMP rendszermagoknál szükséges. Több processzor támogatásánál mindenképpen tegyük hozzá az `options SMP` beállítást is.

Megjegyzés: Az `apic` eszköz csak az i386 architektúrán létezik, ezért a többi architektúrán nem szabad használnunk ezt a beállítást.

```
device          eisa
```

Abban az esetben engedélyezzük, ha EISA-s alaplaponk van, ezzel aktiváljuk az EISA buszra csatlakoztatott eszközök automatikus felismerését és beállíthatóságát.

```
device          pci
```

Tegyük hozzá a konfigurációs állományhoz, ha PCI-os alaplaponk van. Ezzel engedélyezhetjük a PCI kártyák automatikus felismerését és a PCI és ISA buszok közti átírányítást.

```
# Hajlékonylemezes meghajtók
device          fd
```

Ez a hajlékonylemezes meghajtó vezérlője.

```
# ATA és ATAPI eszközök
device          ata
```

Ez az eszközmeghajtó felelős az összes ATA és ATAPI eszközért. A modern számítógépeken csak egyszer kell megadnunk a `device ata` sort a beállítások között az összes PCI-os ATA/ATAPI eszköz felismeréséhez.

```
device          atadisk          # ATA lemezmeghajtók
```

Az ATA lemezmeghajtók támogatásához erre van még szükség a `device ata` mellett.

```
device          ataraid          # ATA RAID-meghajtók
```

Az ATA RAID-meghajtók kezeléséhez erre a sorra van szükség a `device ata` mellett.

```
device          atapicd          # ATAPI CD-meghajtók
```

Az ATAPI CD-meghajtók használatához ezt is tegyük a konfigurációba a `device ata` mellé.

```
device          atapifd          # ATAPI floppy meghajtók
```

A `device ata` használata mellett erre van még szükségünk az ATAPI floppy meghajtók kezeléséhez.

```
device          atapist          # ATAPI szalagos meghajtók
```

Az ATAPI szalagos egységek használatához ezt a sort is tegyük a konfigurációba a `device ata` mellé.

```
options         ATA_STATIC_ID    # statikus eszközszámozás
```

Ezzel a beállítással a vezérlők számozása állandó lesz. Nélküle az eszközszámok dinamikusan kerülnek kiosztásra.

```
# SCSI vezérlők
```

```

device      ahb          # EISA AHA1742 család
device      ahc          # AHA2940 és integrált AIC7xxx eszközök
options     AHC_REG_PRETTY_PRINT  # a hibák kereséséhez kiíratja a regiszterek
                                # bitmezőit. Kb. 128 KB-al növeli a méretét.
device      ahd          # AHA39320/29320 és integrált AIC79xx eszközök
options     AHD_REG_PRETTY_PRINT  # a hibák kereséséhez kiíratja a regiszterek
                                # bitmezőit. Kb. 215 KB-al növeli a méretét.

device      amd          # AMD 53C974 (Teckram DC-390(T))
device      isp          # Qlogic család
#device     ispfw        # a QLogic HBA firmware-e, többnyire modul
device      mpt          # LSI-Logic MPT-Fusion
#device     ncr          # NCR/Symbios Logic
device      sym          # NCR/Symbios Logic (újabb chipsetek, illetve az 'ncr' típusúak)
device      trm          # Tekram DC395U/UW/F DC315U csatolók

device      adv          # Advansys SCSI-csatolók
device      adw          # Advansys wide SCSI-csatolók
device      aha          # Adaptec 154x SCSI-csatolók
device      aic          # Adaptec 15[012]x SCSI-csatolók, AIC-6[23]60.
device      bt           # Buslogic/Mylex MultiMaster SCSI-csatolók

device      ncv          # NCR 53C500
device      nsp          # Workbit Ninja SCSI-3
device      stg          # TMC 18C30/18C50

```

SCSI-vezérlők. Vegyük ki azokat, amelyekkel ténylegesen nem rendelkezünk. Ha csak IDE eszközeink vannak a rendszerünkben, az összeset eltávolíthatjuk. A `_REG_PRETTY_PRINT` végződésű sorok a megfelelő meghajtók hibakerési beállításait takarják.

```

# SCSI-perifériák
device      scbus        # SCSI-busz (kell a SCSI-hoz)
device      ch           # SCSI médiumváltók (media changer)
device      da           # közvetlen hozzáférés (lemezek)
device      sa           # soros hozzáférés (szalag stb.)
device      cd           # CD
device      pass         # áteresztő eszköz (közvetlen SCSI hozzáférés)
device      ses          # SCSI környezeti szolgáltatások (és SAF-TE)

```

SCSI-perifériák. Itt is érvényes, hogy kivethetjük azokat az eszközöket, amelyekkel nem rendelkezünk. De ha csak IDE hardvereink vannak, teljesen eltávolíthatjuk ezeket.

Megjegyzés: Annak ellenére, hogy valójában nem igazi SCSI-eszközök, az USB-s umass(4) és még néhány más egyéb meghajtó is használja a SCSI alrendszert. Emiatt semmiképpen se távolítsuk el a SCSI támogatást a rendszerünkől abban az esetben, ha ilyen meghajtókat is használni szándékozunk.

```

# a SCSI alrendszerhez kapcsolódó RAID-vezérlők
device      amr          # AMI MegaRAID
device      arcmsr       # Areca SATA II RAID
device      asr          # DPT SmartRAID V, VI és Adaptec SCSI RAID
device      ciiss        # Compaq Smart RAID 5*
device      dpt          # DPT Smartcache III, IV - lásd a NOTES állományt

```

```
device      hptmv      # Highpoint RocketRAID 182x
device      rr232x     # Highpoint RocketRAID 232x
device      iir        # Intel Integrated RAID
device      ips        # IBM (Adaptec) ServeRAID
device      mly        # Mylex AcceleRAID/eXtremeRAID
device      twa        # 3ware 9000 series PATA/SATA RAID

# RAID vezérlők
device      aac        # Adaptec FSA RAID
device      aacp       # SCSI áteresztő az aac-hez (kell hozzá a CAM)
device      ida        # Compaq Smart RAID
device      mfi        # LSI MegaRAID SAS
device      mlx        # Mylex DAC960 család
device      pst        # Promise Supertrak SX6000
device      twe        # 3ware ATA RAID
```

Az ismert RAID-vezérlők. Ha közülük egyikkel sem rendelkezünk, távolítsuk el ezeket a konfigurációból.

```
# az atkbd0 vezérli a billentyűzetet és a PS/2-es egeret
device      atkbd      # AT billentyűzet vezérlő
```

A billentyűzet vezérlője (atkbd) az AT-s billentyűzet és a PS/2 stílusú pozícionáló eszközök vezérléséhez szükséges I/O szolgáltatásokat biztosítja. Erre a vezérlőre a billentyűzet meghajtójának (atkbd) és a PS/2 pozícionáló eszközök eszközmeghajtójának (psm) is szüksége van.

```
device      atkbd      # AT billentyűzet
```

Az atkbd meghajtó, a atkbdc vezérlővel együtt, adja a hozzáférést az AT billentyűzet vezérlőre csatlakoztatott AT 84 és a fejlettebb AT billentyűzetek felé.

```
device      psm        # PS/2 egér
```

Használjuk ezt az eszközt, ha az egerünk a PS/2 portra csatlakozik.

```
device      kbdmux     # billentyűzet multiplexer
```

A billentyűzet multiplexer alapszintű támogatása. Ha nem kívánunk a jövőben egynél több billentyűzetet csatlakoztatni a rendszerünkre, nyugodt szívvel kivehetjük ezt a sort.

```
device      vga        # VGA videokártya meghajtó
```

Videokártya meghajtó.

```
device      splash     # üdvözlőképernyők és képernyőkímélők támogatása
```

Nyissunk egy üdvözlőképernyővel! A képernyőkímélőknek is szükségük van erre az eszközre.

```
# a syscons az alapértelmezett konzolmeghajtó, hasonlít a SCO konzolra
device      sc
```

Az sc az alapértelmezett meghajtó a konzolok számára, és sokban hasonlít a SCO konzolra. Mivel a legtöbb teljesképernyős program a termcap termináladatbázis könyvtáron keresztül éri el a konzolt, nem igazán számít, hogy ezt vagy a VT220-kompatibilis vt konzolmeghajtót használjuk. Ha bármilyen gondunk lenne a teljesképernyős

programok futtatásával ezen a konzolon, a bejelentkezéskor állítsuk a TERM környezeti változónkat a `scoansi` értékre.

```
# ezzel tudjuk engedélyezni a pcvt (VT220-kompatibilis) konzolmeghajtót
#device          vt
#options          XSERVER          # az X szerver támogatása vt konzolon
#options          FAT_CURSOR       # telt kurzor használata
```

Ez a VT220-kompatibilis konzolmeghajtó, amely visszafelé kompatibilis a VT100/102-vel is. Remekül működik olyan laptopokon, ahol a hardver nem használható az `sc` konzollal. Itt ugyanúgy érdemes egyébként a `vt100` értékre vagy a `vt220` értékre állítani a TERM környezeti változónkat. Hasznosnak bizonyulhat abban az esetben is, amikor hálózaton keresztül nagy mennyiségű és eltérő típusú számítógépekhez csatlakozunk, és ahol a `termcap` és `terminfo` adatbázisokban az `sc` bejegyzései gyakran nem is érhetőek el — a `vt100` viszont virtuálisan az összes platformon elérhető.

```
device          agp
```

Írjuk bele a konfigurációba, ha van AGP kártya a rendszerünkben. Ezzel engedélyezzük az AGP és az AGP GART támogatását az ezeket ismerő kártyák számára.

```
# energiagazdálkodás támogatása (bővebben lásd: NOTES)
#device          apm
```

A fejlett energiagazdálkodás támogatása. Laptopok esetén hasznos, habár ez alapértelmezés szerint nincs engedélyezve a `GENERIC` konfigurációban.

```
# az i8254 készenléti módjának támogatása
device          pmtimer
```

Az energiagazdálkodási események, mint például APM és ACPI időzítőjének eszközmeghajtója.

```
# PCCARD (PCMCIA) támogatás
# PCMCIA és cardbus támogatás
device          cbb          # cardbus (yenta) bridge
device          pccard       # PC Card (16 bites) busz
device          cardbus      # CardBus (32 bites) busz
```

A PCMCIA támogatása. Mindenképpen szükségünk lesz rá, ha laptopunk van.

```
# soros (COM) portok
device          sio          # 8250, 16[45]50 alapú soros portok
```

Ezek azok a soros portok, amelyek az MS-DOS/Windows világban csak COM portokként ismernek.

Megjegyzés: Ha van egy belső modemünk a `COM4`-en és egy soros portunk a `COM2`-n, a modem IRQ-ját meg kell változtatnunk 2-re (valamilyen homályos műszaki okból kifolyólag a `COM2 = IRQ9`), hogy hozzá tudjunk férni FreeBSD-ből. Ha többportos soros kártyánk lenne, lapozzuk fel a `sio(4)` man oldalát, és ott hozzá megtaláljuk a `/boot/device.hints` állományba írandó megfelelő értékeket. Egyes videokártyák (különösen az S3 chipekre épülők) az I/O címeket `0x*2e8` alakban használják, és mivel rengeteg olcsó soros kártya nem kódolja vissza egészében a 16 bites I/O címet, ütközni fognak ezekkel a kártyákkal, és ezáltal a `COM4` port gyakorlatilag elérhetetlenné válik.

Minden egyes soros portnak egyedi IRQ-jának kell legyen (hacsak nem használunk olyan többportos kártyát, amely támogatja a megosztott megszakításokat), ezért a COM3 és COM4 esetén alapértelmezett IRQ-k nem használhatóak.

```
# párhuzamos port
device          ppc
```

Ez az ISA busz párhuzamos portjának felülete.

```
device          ppbus      # a párhuzamos port busza (kell)
```

A párhuzamos porthoz tartozó busz támogatása.

```
device          lpt        # nyomtató
```

A párhuzamos portra csatlakozó nyomtatók támogatása.

Megjegyzés: A fentiek közül mind a három szükséges a párhuzamos porton csatlakozó nyomtatók használatához.

```
device          plip        # TCP/IP párhuzamos porton keresztül
```

Ez a párhuzamos port hálózati felületének meghajtója.

```
device          ppi         # a párhuzamos port felületének eszköze
```

Általános célú ("geek port") és IEEE1284 I/O.

```
#device         vpo         # az scbus és a da kell a használatához
```

Ez az Iomega Zip meghajtóihoz tartozó eszköz. A működéséhez szükség van az scbus és da engedélyezésére. A legjobb teljesítményt EPP 1.9 módban működő portokkal lehet kihozni belőle.

```
#device         puc
```

Tegyük bele a konfigurációba ezt az eszközt, ha egy olyan "buta" soros vagy párhuzamos PCI kártyánk van, amelyet a puc(4) segédmeghajtó ismer.

```
# PCI Ethernet kártyák
device          de          # DEC/Intel DC21x4x ("Tulip")
device          em          # Intel PRO/1000 Gigabit Ethernet kártya
device          ixgb        # Intel PRO/10GbE Ethernet kártya
device          txp         # 3Com 3cR990 ("Typhoon")
device          vx          # 3Com 3c590, 3c595 ("Vortex")
```

Különféle PCI hálózati kártyák meghajtói. Vegyük ki azokat, amelyek nem találhatók meg a rendszerünkben.

```
# PCI Ethernet kártyák, melyek az MII busz vezérlőkódját használják
# FIGYELEM: Ne töröljük ki a 'device miibus' sort, ha ilyen kártyánk van!
device          miibus      # az MII busz támogatása
```

Az MII busz engedélyezése elengedhetetlen bizonyos 10/100-as PCI Ethernet kártyák használatához, konkrétan azokéhoz, amelyek az MII-vel együttműködni képes adó-vevőt használnak vagy az MII-höz hasonló adó-vevő vezérlő felületet valósítanak meg. A `device miibus` hozzáadása a rendszermaghoz magával vonja az általános miibus API és az összes PHY meghajtó támogatását, beleértve azt az általános PHY eszközt is, amelyet az egyes eszközmeghajtók külön nem támogatnak.

```
device      bce      # Broadcom BCM5706/BCM5708 Gigabit Ethernet
device      bfe      # Broadcom BCM440x 10/100 Ethernet
device      bge      # Broadcom BCM570xx Gigabit Ethernet
device      dc        # DEC/Intel 21143 és egyéb hasonlóak
device      fxp      # Intel EtherExpress PRO/100B (82557, 82558)
device      lge      # Level 1 LXT1001 gigabit ethernet
device      msk      # Marvell/SysKonnect Yukon II Gigabit Ethernet
device      nge      # NatSemi DP83820 gigabit ethernet
device      nve      # nVidia nForce MCP integrált Ethernet hálózat
device      pcn      # AMD Am79C97x PCI 10/100 (az 'lnc' előtt)
device      re        # RealTek 8139C+/8169/8169S/8110S
device      rl        # RealTek 8129/8139
device      sf        # Adaptec AIC-6915 ("Starfire")
device      sis      # Silicon Integrated Systems SiS 900/SiS 7016
device      sk        # SysKonnect SK-984x & SK-982x gigabit Ethernet
device      ste      # Sundance ST201 (D-Link DFE-550TX)
device      stge     # Sundance/Tamarack TC9021 gigabit Ethernet
device      ti        # Alteon Networks Tigon I/II gigabit Ethernet
device      tl        # Texas Instruments ThunderLAN
device      tx        # SMC EtherPower II (83c170 "EPIC")
device      vge      # VIA VT612x gigabit ethernet
device      vr        # VIA Rhine, Rhine II
device      wb        # Winbond W89C840F
device      xl        # 3Com 3c90x ("Boomerang", "Cyclone")
```

Meghajtók, melyek az MII busz vezérlőkódját használják.

```
# ISA Ethernet és pccard hálózati kártyák.
device      cs        # Crystal Semiconductor CS89x0 NIC
# az 'device ed' eszközhöz kell a 'device miibus'
device      ed        # NE[12]000, SMC Ultra, 3c503, DS8390 cards
device      ex        # Intel EtherExpress Pro/10 és Pro/10+
device      ep        # Etherlink III alapú kártyák
device      fe        # Fujitsu MB8696x alapú kártyák
device      ie        # EtherExpress 8/16, 3C507, StarLAN 10 stb.
device      lnc      # NE2100, NE32-VL Lance Ethernet kártyák
device      sn        # az SMC 9000-res sorozatú Ethernet chipjei
device      xe        # Xircom pccard Ethernet
```

```
# ISA eszközök, melyek a régi ISA betétet használják
#device      le
```

ISA Ethernet meghajtók. A konkrétan támogatott kártyák teljes felsorolását lásd a `/usr/src/sys/i386/conf/NOTES` állományban.

```
# vezeték nélküli hálózati kártyák
device      wlan      # 802.11 támogatás
```

Általános 802.11 támogatás. Erre a sorra mindenképpen szükség van a vezeték nélküli hálózatok használatához.

```
device      wlan_wep      # 802.11 WEP támogatás
device      wlan_ccmp     # 802.11 CCMP támogatás
device      wlan_tkip     # 802.11 TKIP támogatás
```

A 802.11 eszközök esetén a titkosítás támogatása. Ezeket a sorokat akkor adjuk meg, ha titkosítást akarunk használni vagy a 802.11-i biztonsági protokolljait.

```
device      an            # Aironet 4500/4800 802.11 vezeték nélküli hálózati kártyák
device      ath           # Atheros pci/cardbus hálózati kártyák
device      ath_hal       # Atheros HAL (Hardware Access Layer)
device      ath_rate_sample # küldési mintavételi vezérlés az ath-hoz
device      awi           # BayStack 660 és mások
device      ral           # Ralink Technology RT2500 vezeték nélküli hálózati kártyák
device      wi            # WaveLAN/Intersil/Symbol 802.11 vezeték nélküli hálózati kártyák
#device     wl            # régebbi, nem 802.11 Wavelan vezeték nélküli hálózati kártyák
```

A különböző vezeték nélküli kártyák támogatása.

```
# Pszeudo eszközök
device      loop          # hálózati loopback
```

Ez a TCP/IP általános loopback eszköze. Ha telnettel vagy FTP-vel rácsatlakozunk a localhost címére (vagyis a 127.0.0.1-re), akkor rajta keresztül saját magunkhoz jutunk vissza. Ennek a megléte *kötelező*!

```
device      random        # álvéletlenszám eszköz
```

Kriptográfiai szempontból biztonságos álvéletlenszám generátor.

```
device      ether         # Ethernet támogatás
```

Az ether eszközre csak abban az esetben van szükség, ha Ethernet kártyánk van. Ez magában foglalja az általános Ethernet protokoll kódját.

```
device      sl            # belső SLIP
```

Az sl a SLIP használatát engedélyezi. Ez egy régi protokoll, amelyet azóta már szinte teljesen kiszorított a PPP, mivel azt könnyebb beállítani és sokkal jobban is illik a modem-modem kapcsolatokhoz, illetve sokkal erőteljesebb.

```
device      ppp           # belső PPP
```

Ez a tárcsázós kapcsolatok rendszermagon belüli PPP támogatását adja meg. Van a PPP-nek egy külső, a felhasználói programként megvalósított változata is, amely a tun eszközt használja és sokkal nagyobb rugalmasságot kínál fel, illetve olyan lehetőségeket, mint például az igény szerinti tárcsázás.

```
device      tun           # csomag alagút
```

Ezt a felhasználói PPP szoftver használja. A könyv PPP-ről szóló részében többet is megtudhatunk róla.

```
device      pty           # Pszeudo terminálok (telnet stb.)
```

Ezek a “pszeudo terminálok”, vagy más néven szimulált bejelentkezési portok. A bejövő telnet és rlogin munkamenetek használják, valamint az **xterm** és a hozzá hasonló alkalmazások, mint például az **Emacs**.

```
device    md                # "memórialemezek"
```

A memóriában levő pszeudo lemezes meghajtók.

```
device    gif                # IPv6 és IPv4 tunnelek használata
```

Megvalósítja az IPv6 IPv4 feletti, az IPv4 IPv6 feletti, az IPv4 IPv4 feletti és az IPv6 IPv6 feletti közvetítést. A gif eszköz "magától másolódik", vagyis szükség szerint hozza létre a megfelelő eszközeleírókat.

```
device    faith              # IPv6-IPv4 közti továbbítás (fordítás)
```

Ez a pszeudo eszköz elfogja a hozzá küldött csomagokat és átadja ezeket az IPv4/IPv6 fordítással foglalkozó démonnak.

```
# a 'bpf' eszköz használatával a Berkeley csomagszűrőt (Berkeley Packet Filter) engedélyezzük
# Legyünk rá tekintettel, hogy ennek komoly következményei lehetnek
# rendszeradminisztrációs szempontból!
# A 'bpf'-re szükség van a DHCP-hez.
device    bpf                # Berkeley csomagszűrő
```

A Berkeley csomagszűrője. Ez egy olyan pszeudo eszköz, amely lehetővé teszi, hogy a hálózati csatlók forgalmát megfigyeljük, mivel a (pl. Ethernet) hálózatunkon minden csomagot elkap. Ezek a csomagok lemezre is menthetők vagy kielemezhetők a tcpdump(1) program segítségével.

Megjegyzés: A bpf(4) eszközt a dhclient(8) is használja többek közt az alapértelmezett átjáró IP-címének megszerzéséhez. Ha DHCP-t akarunk használni, hagyjuk így.

```
# USB támogatás
device    uhci                # UHCI PCI->USB felület
device    ohci                # OHCI PCI->USB felület
device    ehci                # EHCI PCI->USB felület (USB 2.0)
device    usb                 # USB busz (kell)
#device   udbp                # USB Double Bulk Pipe eszközök
device    ugen                # általános
device    uhid                # "Human Interface Devices"
device    ukbd                # billentyűzet
device    ulpt                # nyomtató
device    umass               # lemez/háttértároló - kell hozzá az scbus és a da
device    ums                 # egér
device    ural                # Ralink Technology RT2500USB vezeték nélküli hálózati kártyák
device    urio                # Diamond Rio 500 MP3 lejátszó
device    uscanner            # lapolvasók
# USB Ethernet, kell hozzá az mii
device    aue                 # ADMtek USB Ethernet
device    axe                 # ASIX Electronics USB Ethernet
device    cdce                # általános USB, Etherneten keresztül
device    cue                 # CATC USB Ethernet
device    kue                 # Kawasaki LSI USB Ethernet
device    rue                 # RealTek RTL8150 USB Ethernet
```

A különféle USB eszközök támogatása.

```
# FireWire támogatás
device      firewire    # FireWire buszkód
device      sbp          # SCSI FireWire-ön keresztül (kell hozzá az scbus és a da)
device      fwe          # Ethernet FireWire-ön keresztül (nem szabványos!)
```

A különféle Firewire eszközök támogatása.

A FreeBSD által ismert további eszközökről a `/usr/src/sys/i386/conf/NOTES` állományból tájékozódhatunk.

8.6.1. Sok memória kezelése (PAE)

A sok memóriával rendelkező számítógépek esetén szükség lehet a felhasználói és rendszerszintű virtuális címek (Kernel Virtual Address, KVA) 4 gigabyte feletti használatára. Ennek a korlátozásnak a kiküszöbölésére az Intel külön támogatást épített be a Pentium Pro és az azt követő processzorok 36 bites fizikai címezésének kialakításához.

A Fizikai Címkiterjesztés (Physical Address Extension, PAE) az Intel Pentium Pro és későbbi processzoraiban található meg, és lehetővé teszi egészen 64 gigabyte-ig a memóriahasználatot. A FreeBSD is támogatja ezt a tulajdonságot a PAE rendszermag beállítás használatával, és megtalálható a FreeBSD összes jelenlegi verziójában. Az Intel architektúrájú processzorok memóriaszervezésének korlátai miatt nem különböztethető meg a 4 gigabyte alatti és feletti memória. A 4 gigabyte felett található memóriaterületek egyszerűen hozzáadódnak a rendelkezésre álló memóriához.

A rendszermagban a PAE támogatását egyszerűen az alábbi sor hozzáadásával tudjuk engedélyezni:

```
options          PAE
```

Megjegyzés: A FreeBSD-ben a PAE támogatása csak az Intel IA-32 architektúrájú processzoraihoz érhető el. Emellett meg kell említenünk, hogy a FreeBSD-ben található PAE támogatás nem lett szélesebb körben próbára téve, ezért a FreeBSD többi megbízható elemeihez képest csak béta állapotúnak tekinthető.

A FreeBSD PAE támogatásának van néhány hiányossága:

- Egy futó program a virtuális memóriában nem képes 4 gigabyte-nál többet elérni.
- A `bus_dma(9)` felületet nem használó eszközmeghajtók adathibákat okozhatnak a PAE-t támogató rendszermagokban, és emiatt nem ajánljuk a használatukat. Ebből a megfontolásból készítettünk egy `PAE` nevű konfigurációs állományt a FreeBSD-hez, amelyben nem szerepel egyetlen olyan meghajtó sem, amely ismereteink szerint nem működik együtt a PAE-t támogató rendszermagokkal.
- Bizonyos finomhangolási beállítások a memóriahasználatot a rendelkezésre álló fizikai memória mennyiségéből számítják ki. A PAE támogatással működő rendszerek esetében megjelenő sok memória miatt azonban az ilyen eszközök szükségtelenül több területet foglalhatnak le. Erre példa lehet a `kern.maxvnodes` `sysctl` változó, amely a rendszermag által maximálisan felhasználható virtuális csomópontok számát korlátozza. Ajánlott tehát az ilyen és ehhez hasonló beállítások értelmes értékre történő visszaállítása.
- Szükséges lehet a rendszermag virtuális címtérének (KVA) növelése vagy a rendszermag által túlságosan nagy méretűre foglalt címtérű különféle erőforrások (lásd fentebb) csökkentése a KVA kifogyásának elkerülésére. A KVA területének növelését a `KVA_PAGES` beállításával tehetjük meg.

Ha gondjaink lennének a teljesítménnyel vagy a megbízhatósággal, keressük fel a `tuning(7)` man oldalt. A `pae(4)` man oldalon pedig a FreeBSD PAE támogatásáról találhatunk naprakész információkat.

8.7. Ha valamilyen hiba történe

Négyféle probléma jelentkezhet egy saját rendszermag készítése során. Ezek:

A `config` hibát jelez:

Amikor a `config(8)` parancs hibát jelez vissza a rendszermagunk konfigurációs beállításainak feldolgozása során, akkor minden bizonnyal csak egy apró hibát vétettünk valahol. Szerencsére a `config(8)` kiírja a hibás sor számát, ezért gyorsan fel tudjuk kutatni a hibát tartalmazó sort. Például, ha ezt látjuk:

```
config: line 17: syntax error
```

Akkor győződjünk meg róla, hogy helyesen írtuk be az adott sorban szereplő kulcsszót. Ebben segítségünkre lehet, ha összevetjük a `GENERIC` konfigurációs állománnyal vagy más hivatkozásokkal.

A `make` hibát jelez:

Ha a `make` jelez hibát, az általában arra utal, hogy az általunk korábban megadott rendszermag konfigurációs állományt a `config(8)` nem értette meg rendesen. Megint azt tudjuk csak javasolni, hogy nézzük át a konfigurációs beállításainkat, és ha ezután sem sikerül megoldani a problémát, akkor mellékeljük egy levélben a rendszermagunk konfigurációs beállításait és küldjük el a FreeBSD general questions levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>) címére, ahol a hozzáértők gyorsan átnézik.

A rendszermag nem indul:

Ha az új rendszermagunk nem indul vagy nem képes felismerni az eszközeinket, ne essünk kétségbe! Szerencsére a FreeBSD tökéletes megoldással tud szolgálni az összeférhetetlen rendszermagok esetére: a FreeBSD rendszerbetöltőjében egyszerűen válasszuk ki az indítandó rendszermagot. Ezt akkor tudjuk előhívni, amikor a rendszerindító menü megjelenik. Válasszuk ki a hatos, vagyis az "Escape to a loader prompt" (a betöltő parancssorának előhívása) menüpontot. Mikor megjelenik a parancssor, írjuk be, hogy `unload kernel`, majd adjuk ki a `boot /boot/kernel.old/kernel`, parancsot, amiben bármilyen más olyan rendszermagot is megnevezhetünk, ami korábban már működött. Ezért amikor beállítunk egy új rendszermagot, mindig érdemes a kezünk ügyében tartani legalább egy olyan rendszermagot, amely működik.

Miután sikerült elindítanunk az egyik használható rendszermagot, nézzük át még egyszer a konfigurációs állományt és próbáljuk újra lefordítani a rendszermagot. A probléma megoldását segítheti a `/var/log/messages` állomány áttanulmányozása is, ami többek közt rögzíti a rendszermag sikeres indulása során keletkező üzeneteket. Ezenkívül a `dmesg(8)` parancs is meg tudja jeleníteni az aktuális rendszerindítás üzeneteit.

Megjegyzés: Ha gondok merülnének fel a rendszermag elkészítése során, mindenképpen tartsuk meg a `GENERIC`, vagy bármilyen másik olyan rendszermagot, amelyről tudjuk, hogy működik. Nevezzük át, így nem fog felülíródni a következő fordítás és telepítés során. A `kernel.old` állományra ugyanis nem minden esetben számíthatunk, mivel az új rendszermagok telepítésénél a `kernel.old` mindig felülíródik a legutóbb telepített rendszermaggal, amely azonban nem feltétlenül lesz működőképes. Sőt, amint csak lehetséges, rakjuk a működő rendszermagot a `/boot/kernel` könyvtárba vagy különben a `ps(1)` és a hozzá hasonló parancsok nem fognak rendesen működni. Mindezek elvégzéséhez egyszerűen nevezzük át a jó rendszermagot tartalmazó könyvtárt:

```
# mv /boot/kernel /boot/kernel.rossz
# mv /boot/kernel.jó /boot/kernel
```

A rendszermag működik, a ps(1) viszont nem:

Ha olyan rendszermagot telepítettünk, aminek a verziója nem egyezik meg a hozzá tartozó segédprogramokéval, tehát például -CURRENT rendszermagot raktunk egy -RELEASE rendszerhez, egyes rendszerállapotjelző parancsok, mint például a ps(1) vagy a vmstat(8) nem fognak működni. Ebben az esetben az egész rendszert újra kell fordítanunk és telepítenünk a rendszermagunkkal megegyező verziójú forrásból. Részben ezért sem különösen ajánlott, hogy az operációs rendszer többi részétől eltérő verziójú rendszermagot használjunk.

Fejezet 9. Nyomtatás

Írta: Sean Kelly. Átdolgozta és frissítette: Jim Mock.

9.1. Áttekintés

A FreeBSD a nyomtatók széles skálájával képes együttműködni, a legrégebbi vegyszeres nyomtatótól kezdve egészen napjaink lézernyomtatójáig, aminek köszönhetően alkalmazásainkkal nagyon jó minőségű nyomtatásokat tudunk készíteni.

A FreeBSD a helyi hálózaton nyomtatószervernek is beállítható. Ekkor a vele közös hálózatra csatlakozó többi, FreeBSD, Windows vagy Mac OS rendszerű számítógéptől képes nyomtatási kéréseket elfogadni. A FreeBSD gondoskodik róla, hogy egyszerre csak egy nyomtatás készüljön el, számon tartja, hogy mely felhasználók és számítógépek nyomtatnak a legtöbbit, és minden feladathoz “munkalapot” (banner page) készít, amiben többek közt megtalálhatjuk, hogy kihez tartozik.

A fejezet elolvasása során megismerjük:

- hogyan állítsuk be a FreeBSD nyomtatási sorát;
- hogyan telepítsünk nyomtatási szűrőket, hogyan kezeljünk különböző speciális nyomtatási feladatokat, tehát például miként alakítsuk át a beérkező dokumentumokat olyan nyomtatási formátumra, amelyet a nyomtatónk is megért;
- hogyan engedélyezzük a fejléc- vagy nyomtatási információk kinyomtatását;
- hogyan nyomtassunk más számítógépekhez csatlakoztatott nyomtatókkal;
- hogyan nyomtassunk a hálózatra közvetlenül kapcsolt nyomtatókkal;
- hogyan állítsuk be a nyomtatási korlátozásokat, például a nyomtatási feladatok méretét, amivel egyes felhasználók nyomtatását visszafoghatjuk;
- hogyan készítsünk nyomtatási kimutatásokat és nyilvántartást a nyomtató használatáról;
- hogyan keressük meg a nyomtatás során felmerülő problémák okait.

A fejezet elolvasásához ajánlott:

- egy új rendszermag beállításának és telepítésének ismerete (8 fejezet).

9.2. Bevezetés

A FreeBSD-ben a nyomtatók működéséhez be kell állítani az **LPD** nyomtatási rendszert. Ez a Berkeley sornyomtatási rendszere, amelyet ezentúl röviden csak **LPD**-nek fogunk hívni. Ez a FreeBSD alapértelmezett szabványos nyomtatásvezérlő rendszere. Ebben a fejezetben az **LPD** és annak konfigurációja kerül bemutatásra.

Ha már találkoztunk az **LPD**-vel vagy hozzá hasonló rendszerekkel, akkor innen nyugodtan ugorhatunk a Kezdeti beállítások című szakaszra.

Az **LPD** vezérli a számítógéphez csatlakoztatott nyomtató összes funkcióját. Számos feladata van:

- Felügyeli a lokálisan és hálózaton keresztül csatlakoztatott nyomtatók hozzáféréseit.
-

Lehetővé teszi az átküldött állományok kinyomtatását, amelyeket *nyomtatási feladatoknak* neveziünk.

- Minden nyomtatóhoz fenntart egy nyomtatási *sor*t, amivel meg tudja akadályozni, hogy egyszerre több felhasználó is hozzá tudjon férni az egyes nyomtatókhoz.
- A *fejléceket* (vagy más néven *munka-* vagy *elválasztó lapokat*) nyomtat, így a felhasználók könnyen megtalálják a saját nyomtatásaikat a többi közt.
- Felügyeli a soros portokon csatlakozó nyomtatók kommunikációs beállításait.
- A hálózaton keresztül átküldi a nyomtatási feladatokat egy másik számítógép **LPD** sorába.
- A nyomtatási feladatok formázásához lefuttatja az adott nyomtató nyelvéhez és képességeihez illeszkedő speciális szűrőket.
- Nyilvántartja a nyomtató kihasználtságát.

A beállításait tartalmazó állomány (`/etc/printcap`) és a speciális szűrőprogramok segítségével az **LPD** sokféle nyomtaton képes az összes említett feladatot vagy annak egy részét megvalósítani.

9.2.1. Amiért nyomtatási sort érdemes használni

Amikor csak egyedül vagyunk a rendszerben, felmerülhet bennünk a kérdés, hogy minek is kellene nekünk vesződni a nyomtatási sor beállításával, hiszen nincs szükségünk sem a hozzáférések vezérlésére, sem fejlécekre, sem pedig nyilvántartásra. Noha akár közvetlenül is el tudjuk érni a nyomtatót, néhány okból azért mégis érdemes nyomtatási sort használni:

- Az **LPD** a háttérben nyomtat, ezért ilyenkor nem kell megvárni, amíg az adat átmásolódik a nyomtatóra.
-

Az **LPD** tetszőlegesen tudja alakítani a nyomtatási feladatokat: hozzájuk tud tenni különböző adatokat (dátum és idő), vagy a speciális állományokat (például a \TeX DVI formátumát) képes megértetni a nyomtatóval, és nem nekünk kell mindezeket a lépéseket elvégeznünk.

- Számos nyomtatási lehetőséggel rendelkező szabad és kereskedelmi program arra számítt, hogy a rendszerünkben nyomtatási sor található, ezért annak beállításával sokkal könnyebb használni ezeket a szoftvereket.

9.3. Kezdeti beállítások

Úgy tudjuk használni a nyomtatókat az **LPD** nyomtatási rendszerével, ha egyaránt beállítjuk a nyomtatót és magát az **LPD**-t is. Itt a beállítás két szintjét tárgyaljuk:

- Az Alacsonyszintű nyomtatóbeállítás című szakaszból megtudhatjuk, hogyan tudunk csatlakoztatni egy nyomtatót, hogyan adjuk meg az **LPD**-nek, miként kommunikáljon vele, hogyan nyomtassunk ki egyszerű szöveges állományokat a nyomtatón.
- A Magasszintű nyomtatóbeállítás szakaszban bemutatjuk, hogyan nyomtassunk ki különféle speciális állományokat, hogyan készítsünk fejléceket, hogyan nyomtassunk hálózaton keresztül, hogyan vezéreljük a nyomtatók hozzáférését és hogyan tartsuk nyilván a nyomtató használatát.

9.3.1. Alacsonyszintű nyomtatóbeállítás

Ebben a szakaszban láthatjuk, miképpen kell beállítani a nyomtatónkat és az **LPD** hogyan lesz képes azt használatba venni. Az alapoktól kezdünk:

- A Hardveres beállítás című szakaszban abban kapunk segítséget, hogyan kell a nyomtatót a számítógéphez csatlakoztatni.
- A Szoftveres beállítás című szakaszban az **LPD** nyomtatási rendszer beállítását tartalmazó állományt (/etc/printcap) vesszük sorra.

Amennyiben olyan nyomtatót akarunk beállítani, amely nem helyileg, hanem valamilyen hálózati protokollon keresztül csatlakozik, nézzük meg a Nyomtatók hálózati adatcsatlakozással című szakaszt.

Habár ez a szakasz nevében csupán “Alacsonyszintű nyomtatóbeállításról” szól, meglehetősen szerteágazó tud lenni. A nyomtató hardveres és szoftveres életre keltése az egyik legnehezebb feladat. Ha van egy működő nyomtatónk, a fejlécek és a nyilvántartás beállítása tulajdonképpen már gyerekjáték.

9.3.1.1. Hardveres beállítás

Ebben a szakaszban a nyomtatók csatlakoztatásának lehetséges módozatairól esik szó. Beszélni fogunk mindenféle portokról és kábelekről, és a FreeBSD rendszermagjának az egyes nyomtatók használatához szükséges beállításairól is.

Ha korábban tudtuk csatlakoztatni a nyomtatónkat, és más operációs rendszerekkel már sikeresen nyomtattunk is vele, akkor rögtön ugorhatunk is a Szoftveres beállításokat tartalmazó szakaszra.

9.3.1.1.1. Portok és kábelek

A személyi számítógépekhez kapható nyomtatók általában a következő három csatolófelület egyikével rendelkeznek:

•

A *soros*, más néven RS-232-es vagy COM porton keresztül kommunikáló felületek a számítógép soros portján küldenek adatot a nyomtatónak. A soros csatolófelületek igen elterjedtek a számítógépiparban, könnyen tudunk ilyen kábelt szerezni, gyorsan is gyártható. Előfordulhat, hogy a soros csatolófelületek használatához valamilyen különleges kábelre, valamint bonyolult kommunikációs beállítások megadására van szükség. A legtöbb soros port által elérhető legnagyobb adatátviteli sebesség másodpercenként 115 200 bit, ami miatt azonban a komolyabb grafikai tartalmak nyomtatása szinte lehetetlen.

•

A *párhuzamos* csatolófelületek a számítógépünk párhuzamos portjával küldenek adatokat a nyomtatónak. A párhuzamos felületek gyorsabbak az RS-232 soros felületnél, és a számítógéppiacon is gyakran megtalálhatóak. Könnyen tudunk ilyen kábelt szerezni, azonban kézzel nehezebb elkészíteni. A párhuzamos csatolófelületekhez általában nem tartoznak kommunikációs beállítások, ezért rendkívül egyszerűen el lehet boldogulni velük.

A párhuzamos felületekre olykor “Centronics” csatolófelületként is hivatkoznak, amelyet egy nyomtatótípus után neveztek el.

•

A Universal Serial Bus (Univerzális soros busz) rövidítéseként használt USB elnevezésű csatolófelület a párhuzamos és a soros felületeknél jóval nagyobb sebességre képes. A hozzá tartozó kábelek felépítése egyszerű és az áruk olcsó. Habár a nyomtatás terén az USB hivatott leváltani az RS-232-es soros és a párhuzamos felületeket,

nem mindegyik UNIX rendszer támogatja kellőképpen. Ezt a problémát például úgy kerülhetjük el, ha olyan nyomtatót vásárolunk, amelyen a legtöbbhez hasonlóan a párhuzamos és az USB csatlakozás is megtalálható.

A párhuzamos felületeken általában csak egy irányban tudunk üzeneteket küldeni (a számítógéptől a nyomtatóhoz), miközben az USB és a soros felület használatával mind a két irányban is. FreeBSD alatt viszont már az újabb (EPP és ECP) párhuzamos portok egy IEEE 1284 szabványú kábellel képesek oda-vissza kommunikálni.

A párhuzamos nyomtatók kétirányú kommunikációját általában két mód közül az egyiket szokták megvalósítani. Az első esetben a FreeBSD a nyomtatóhoz egy speciális meghajtót használ, amely ismeri az általa beszélt nyelvet. Ilyenek a tintasugaras nyomtatók, amelyek más egyéb állapotinformációk mellett ezen keresztül képesek jelezni a tintapatronokban levő tinta mennyiségét. A második esetben a nyomtató ismeri a PostScript nyelvet.

A PostScript nyelvű nyomtatási feladatok valójában a nyomtatónak küldött programok. Használatukhoz még papírra sincs feltétlenül szükség, és előfordulhat, hogy közvetlenül a számítógépnek válaszolnak. A PostScript is kétirányú kommunikáción keresztül értesíti a számítógépet az olyan gondokról, mint például a PostScript programokban levő hibák vagy a papír beakadása, amely információnak a felhasználók szoktak örülni. Hovatovább ez a kétirányú kommunikáció a kulcsa a PostScript nyomtatók hatékony nyilvántartásának is: egyszerűen lekérdezzük a nyomtatótól a lapszámlálót (ami megadja, hogy a nyomtató eddig mennyi lapot nyomtatott ki), kiküldjük a felhasználóhoz tartozó feladatot és ismét lekérdezzük a lapszámlálót. A két érték kivonásából tájékozódhatunk a felhasználó által igényelt lapok mennyiségéről.

9.3.1.1.2. Párhuzamos portok

A párhuzamos csatolófelületen érintkező nyomtató használatához kapcsoljunk össze számítógépünket és nyomtatónkat egy párhuzamos kábellel. Az erre vonatkozó konkrét utasítások a nyomtató és/vagy a számítógép kézikönyvében olvashatóak.

Jegyezzük meg, hogy a számítógép melyik párhuzamos portjára csatlakoztattuk a kábelt. FreeBSD alatt az első ilyen port a `ppc0` eszköz, a második pedig a `ppc1` eszköz lesz és így tovább. A nyomtatóeszköz elnevezése ugyanezt a sémát követi: a `/dev/lpt0` lesz az első párhuzamos porton levő nyomtató stb.

9.3.1.1.3. Soros portok

A soros csatolófelületet használó nyomtatók beüzemeléséhez először egy soros kábel segítségével kapcsoljuk össze a számítógépünkkel. Ennek pontos részleteit a nyomtató és/vagy a számítógépünk kézikönyvében találhatjuk meg.

Ha nem vagyunk benne biztosak, hogy milyen a “megfelelő soros kábel”, próbáljunk az alábbiak alapján dönteni:

- A *modem* kábele a két oldalán levő, egymásnak megfelelő tüskéket közvetlenül összeköti. Ezt a típust nevezik “DTE-DCE” kábelnek.

•

A *null-modem* kábel bizonyos érintkezőket rendesen, másokat pedig fordítva köt össze (például a küldőt a fogadóval), illetve némelyeket rövidre zár közvetlenül a csatlakozón belül. Ez a típus a “DTE-DTE” kábel.

- Néhány speciális nyomtató esetén előfordul még a *soros nyomtatókábel*, amely leginkább a null-modem kábelekhez hasonlít, azonban az ott rövidre zárt csatornák itt a nekik megfelelő érintkezőknek továbbítanak jeleket.

Emellett még a nyomtató előlapján vagy az alján található kapcsolók segítségével be kell állítanunk a nyomtatóhoz tartozó kommunikációs paramétereket is. Itt válasszuk azt a `bps` (a bitek száma másodpercenként) értéket, amelyet még a számítógépünk és a nyomtatónk is egyaránt képes támogatni. Válasszunk 7 vagy 8 adatbitet, páros, páratlan

vagy kikapcsolt paritásbitet és 1 vagy 2 stopbitet. Ekkor tudjuk megadni a forgalomirányítási protokollt is: lehet kikapcsolt, XON/XOFF (ez az ún. “sávon belüli” vagy “szoftveres”) forgalomirányítás. Ne felejtjük el ezeket a beállításokat a most következő szoftveres beállítások elvégzése során sem.

9.3.1.2. Szoftveres beállítás

Ebben a fejezetben tárgyaljuk a FreeBSD-ben található **LPD** nyomtatási rendszer működéséhez és a nyomtatáshoz szükséges szoftveres beállításokat.

Íme az elvégzendő lépések rövid vázlata:

1. Amennyiben szükséges, állítsuk be a rendszermagunkat a nyomtató által használt portra. Ehhez A rendszermag beállítása szakaszban olvashatjuk el, mit is kell pontosan tenni.
2. Ha párhuzamos portot használunk, akkor állítsuk be, hogy a párhuzamos port miként fog kommunikálni. A párhuzamos port kommunikációs módjának beállítása című szakasz tárja fel ennek részleteit.
3. Próbáljuk ki, hogy ezek után az operációs rendszer képes-e adatot küldeni a nyomtatónak. A nyomtató kommunikációjának ellenőrzése szakaszban kapunk erre pár javaslatot.
4. Az `/etc/printcap` állomány felhasználásával állítsuk be a nyomtatónkhoz az **LPD**-t. Erről a fejezet további részei adnak majd felvilágosítást.

9.3.1.2.1. A rendszermag beállítása

Az operációs rendszer magja eszközök egy adott csoportjával képes együttműködni, amelyben a soros és párhuzamos felületen csatlakozó nyomtatók is megtalálhatóak. Azonban ha a rendszermag nem ismeri fel még valamelyiket, akkor a soros vagy párhuzamos portok használatához külön támogatásra van szükség.

Így tudjuk megnézni, hogy a jelenleg használt rendszermag támogatja-e a soros csatolófelületet:

```
# grep sion /var/run/dmesg.boot
```

Itt az *N* nullától kezdődően adja meg a soros port sorszámát. Amennyiben látunk valami ilyesmit:

```
sio2 at port 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
```

Ez azt jelenti, hogy a rendszermag sikeresen észlelte a portot.

A párhuzamos csatolófelület támogatásáról így győződhetünk meg:

```
# grep ppcN /var/run/dmesg.boot
```

Itt az *N* nullától kezdődően sorszámozza a párhuzamos portot. Ha eredményül valami hasonlót kapunk:

```
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/8 bytes threshold
```

Ez arra utal, hogy a rendszermagunk tud a portról.

Előfordulhat azonban, hogy az operációs rendszer csak akkor fogja észrevenni a nyomtatásra használt soros vagy párhuzamos portot, ha átállítjuk a rendszermagunkat.

A soros port támogatásának beállításához olvassuk el a rendszermag beállításáról szóló szakaszt. A párhuzamos port támogatásához szintén olvassuk el ugyanazt a szakaszt és a most következőt.

9.3.1.3. A párhuzamos port kommunikációs módjának beállítása

A párhuzamos csatolófelület használata esetén választhatunk, hogy a FreeBSD milyen módon tartsa a kapcsolatot a nyomtatóval: megszakításokkal vezérelje (interrupt-driven), vagy esetleg folyamatosan kérdezgesse (polled). A FreeBSD általános meghajtója (lpt(4)) a ppbus(4) alrendszert használja, ami a portot a ppc(4) meghajtón keresztül vezérli.

- A *megszakítás alapú* módszer a GENERIC rendszermagban alapértelmezés. Ilyenkor az operációs rendszer egy megszakításkérés felhasználásával értesül arról, hogy a nyomtató mikor áll készen adatok fogadására.
- A *lekérdezéses* módszer használata során az operációs rendszer folyamatosan érdeklődik a nyomtató rendelkezésre állásáról. Amikor erre pozitív megerősítést kap, akkor a rendszermag újabb adatokat küld.

A megszakításos módszer valamivel gyorsabb, azonban cserébe lefoglal egy értékes IRQ vonalat. A HP újabb nyomtatói állítólag nem működnek megfelelően ilyen módban, valamilyen (pillanatnyilag még nem teljesen tisztázott) időzítési probléma miatt. Ezért az ilyen nyomtatóknak is valószínűleg a lekérdezéses módszert kell használniuk. Más nyomtatók pedig, habár működnek mind a két módszerrel, hihetetlenül lassúak a megszakításokkal.

Kétféleképpen állíthatjuk be a kommunikációs módot: a rendszermagon keresztül, vagy az lptcontrol(8) segédprogrammal.

A rendszermagban így állíthatjuk be a kommunikációt:

1. Írjuk át a rendszermag beállításait tartalmazó állományt. Keressük meg benne a használt párhuzamos portnak megfelelően a ppc0, ppc1 (második párhuzamos port) vagy ppc2 (harmadik párhuzamos port) bejegyzést, és engedélyezzük.
 - A megszakításos mód használatához nyissuk meg a /boot/device.hints állományt, és az *N* helyére írjuk be a
`hint.ppc.0.irq="N"`
 sorba a megfelelő IRQ számát. A rendszermag beállításait tartalmazó állománynak tartalmaznia kell a ppc(4) meghajtót is:
`device ppc`
 - A lekérdezéses mód használatához a /boot/device.hints állományból távolítsuk el a következő sort:
`hint.ppc.0.irq="N"`
 Némely esetben azonban ennyi még nem lesz elég a port lekérdezéses beállításához. Ugyanis ha a hozzá tartozó meghajtó az acpi(4), akkor ez fogja felismerni, kezelni és a nyomtatóhoz tartozó portok hozzáférési módját vezérelni. A problémát ezért gyakran érdemes az acpi(4) beállításai között is keresni.
2. Mentsük el az állományt. Konfiguráljuk be, fordítsuk le és telepítsük az új rendszermagot. Ennek pontos részleteit a rendszermag beállításáról szóló fejezetben olvashatjuk.

A kommunikáció módjának beállítása az lptcontrol(8) programmal:

1. A megszakításos mód beállításához írjuk be:

```
# lptcontrol -i -d /dev/lptN
```

ahol az `lptN` a nyomtatóhoz tartozó eszköz neve.

2. A lekérdezéses mód beállításához írjuk be:

```
# lptcontrol -p -d /dev/lptN
```

ahol az `lptN` a nyomtatóhoz tartozó eszköz neve.

Ha ezeket a parancsokat berakjuk az `/etc/rc.local` állományunkba, akkor azzal a rendszer minden egyes indítása során beállítjuk a számunkra megfelelő módot. Erről többet az `lptcontrol(8)` man oldaláról tudhatunk meg.

9.3.1.4. A kommunikáció ellenőrzése

Még mielőtt nekilátnánk a nyomtatási rendszer beállításának, bizonyosodjunk meg róla, hogy az operációs rendszer képes adatokat továbbítani a nyomtatónak. Sokkal könnyebb egymástól függetlenül megvizsgálni a kommunikáció és a nyomtatási rendszer működését.

A nyomtatót úgy tudjuk kipróbálni, ha küldünk neki valamilyen szöveget. Az `lptest(1)` tökéletesen megfelelő akkor, ha olyan nyomtatónk van, amely azonnal kinyomtatja a kapott szöveget. Ez a program 96 sorban létrehozza mind a 96 kinyomtatható ASCII karaktert.

A PostScript (vagy más egyéb nyelvet ismerő) nyomtatóknak azonban ennél kifinomultabb próbára van szüksége. Erre a célra tökéletesen megfelel egy olyan kisebb PostScript programocska, mint például ez:

```
%!PS
100 100 moveto 300 300 lineto stroke
310 310 moveto /Helvetica findfont 12 scalefont setfont
(Remek! Ez mukodik!) show
showpage
```

Ezt a PostScript kódot nyugodtan elmenthetjük egy állományba, amelyet aztán a későbbi szakaszokban megjelenő példák szerint használni is tudunk majd.

Megjegyzés: A kézikönyvben a nyomtató nyelve alatt leginkább egy PostScript-szerű nyelvet értünk, nem pedig a Hewlett Packard PCL típusú nyelvét. Habár a PCL nagyon sokra képes, hiszen keverhetjük még benne akár a programokat és a nyers szövegeket is. Ezzel szemben a PostScript nem képes nyers szöveget kinyomtatni, ezért az ilyen típusú nyomtatók működtetéséhez külön támogatásra van szükségünk.

9.3.1.4.1. A párhuzamos nyomtató ellenőrzése

Ebben a szakaszban megtudhatjuk, hogy FreeBSD alatt miként ellenőrizzük a párhuzamos portra csatlakozó nyomtatók működését.

A párhuzamos porton levő nyomtató kipróbálásához:

1. A `su(1)` segítségével váljunk `root` felhasználóvá.
2. Küldjünk a nyomtatónak valamilyen adatot.
 - Ha a nyomtató képes nyers szöveget fogadni, akkor használjuk az `lptest(1)` programot. Ehhez gépeljük be:

```
# lptest > /dev/lptN
```

ahol az *N* nullától kezdődően a párhuzamos port sorszáma.

- Ha a nyomtató PostScript vagy más nyomtatási nyelvet ismer, akkor egy apró programot kell küldenünk neki. Ehhez írjuk be:

```
# cat > /dev/lptN
```

Ezután soronként írjuk be a programot, de *vigyázzunk*, mert az **Enter** vagy a **Return** lenyomása után már nem tudjuk kijavítani! A program begépelése után nyomjuk meg a **Ctrl+D** vagy bármely más olyan billentyűkombinációt, amivel ki tudunk lépni.

Ezt a programot belerakhatjuk egy állományba is, amire aztán adjuk ki az alábbi parancsot:

```
# cat állomány > /dev/lptN
```

ahol az *állomány* a nyomtatóra küldendő program neve lesz.

Ezután a nyomtató megkezdí a nyomtatást. Ne aggódjunk, ha netalán valami furcsán nézne ki, mert a későbbiekben ezt még úgyis rendbetesszük.

9.3.1.4.2. A soros nyomtató ellenőrzése

Ebben a szakaszban megtudhatjuk, hogyan ellenőrizzük a FreeBSD és soros portra kötött nyomtató kapcsolódását.

Így tudjuk kipróbálni a soros porton csatlakozó nyomtatónkat:

1. A `su(1)` paranccsal váljunk `root` felhasználóvá.
2. Nyissuk meg az `/etc/remote` állományt. Tegyük hozzá a következő sort:

```
printer:dv=/dev/port:br#bps:pa=paritás
```

ahol a *port* a soros porthoz tartozó eszközeírő neve (`ttyd0`, `ttyd1`, stb.), a *bps* a nyomtató által használt adatátviteli sebesség, végül a *paritás* a nyomtatóhoz használt paritás (ami lehet even (páros), odd (páratlan), none (nincs), vagy zero (nulla)).

Íme egy olyan soros nyomtató beállítása (`printer` néven), amely sebessége 19 200 bps, a harmadik portra csatlakozik és nem használ paritást:

```
printer:dv=/dev/ttyd2:br#19200:pa=none
```

3. Kapcsolódjunk a nyomtatóhoz a `tip(1)` segítségével. Ennek parancsa:

```
# tip printer
```

Ha az iménti lépés nem működne, próbálkozzunk az `/etc/remote` állomány újbóli módosításával, és a `/dev/cuaaN` eszköz helyett használjuk a `/dev/ttydN` eszközt!

4. Küldjünk adatot a nyomtatónak.

- Ha a nyomtató képes nyers szöveget nyomtatni, akkor használjuk az `lptest(1)` segédprogramot. Gépeljük be:

```
% $lptest
```

- Ha a nyomtató a PostScript vagy egy hozzá hasonló nyomtatási nyelven kommunikál, akkor a nyomtatónak egy rövid programot kell küldenünk. Soronként gépeljük be a programot, azonban *vigyázzunk* arra, hogy a törlés és minden más szerkesztésre használt billentyű a nyomtató számára is értelmes lehet. Az is

előfordulhat, hogy a program küldését egy speciális jelsorozattal tudjuk csak lezárni. A PostScript nyomtatók esetén ilyenkor elegendő a **Ctrl+D** billentyűk együttes lenyomása.

Vagy tehetjük az egész programot egy állományba, amihez aztán írjuk be ezt:

```
% >állomány
```

ahol az *állomány* a programot tartalmazó állomány neve. Miután a tip(1) elküldte az állományt, nyomjuk le a lezáráshoz szükséges billentyűkombinációt.

Most már meg kellene jelennie valaminek a nyomtatón. Az még nem számít, pontosan mi is lesz az — később még majd úgyis beállítjuk.

9.3.1.5. A nyomtatási rendszer aktiválása: a `/etc/printcap` állomány

Csatlakoztattuk a nyomtatónkat, a működtetéséhez beállítottuk a rendszermagot (amennyiben erre szükségünk volt), és tudtunk neki adatokat küldeni. Most már készen állunk arra, hogy **LDP** alkalmazáson keresztül beállítsuk a nyomtató hozzáféréseinek vezérlését.

Az **LPD** beállításait az `/etc/printcap` állományban találjuk. Az **LPD** nyomtatási rendszer minden egyes művelet előtt beolvassa ezt az állományt, ezért a benne végzett módosítások szinte azonnal életbe is lépnek.

A `printcap(5)` tartalma könnyen érthető, a `/etc/printcap` állományt egyszerűen módosíthatjuk a kedvenc szövegszerkesztőnkkel. A felépítése teljesen megegyezik a többi hozzá hasonló állományéval: ilyenek például a `/usr/share/misc/termcap` és a `/etc/remote`. Az itt alkalmazott formátum teljes leírását a `cgetent(3)` man oldalon találjuk.

A nyomtatási rendszer egyszerű beállítása az alábbi lépésekből áll:

1. Adjunk nevet (és még néhány álnevet) a nyomtatónak, írjuk ezeket az `/etc/printcap` állományba. A nevekről A nyomtató elnevezése című szakaszban kapunk felvilágosítást.
2. A(z) alpból bekapcsolt) fejléclapokat az `sh` tulajdonság megadásával kapcsolhatjuk ki. A részleteket A fejléclapok letiltása című szakaszban találjuk.
3. Hozzunk létre egy nyomtatási könyvtárat, és adjuk meg a helyét az `sd` tulajdonság beállításával. A nyomtatási könyvtár létrehozása című szakaszban fogunk erről többet mondani.
4. Állítsunk be egy nyomtató által használt `/dev` könyvtárbeli leíró, és az `lp` tulajdonsággal adjuk meg az `/etc/printcap` állományban. Erről részletesebben A nyomtatóeszköz azonosítása című szakaszban olvashatunk. Ha a nyomtató soros porton keresztül csatlakozik, az `ms#` tulajdonsággal még meg kell adnunk A nyomtatási rendszer kommunikációs paraméterei szakaszban tárgyaltakat is.
5. Helyezzünk el egy szűrőt a beérkező nyers szövegek számára. Erről A szövegszűrő telepítése című szakasz értekezik.
6. Az `lpr(1)` parancs segítségével próbáljuk ki a nyomtatást. Ennek pontos részleteit a Próbáljuk ki! és a Hibakeresés című fejezetekben találhatjuk meg.

Megjegyzés: A magasabb szintű nyomtatók, mint például a PostScript nyomtatók nem képesek közvetlenül nyers szöveget nyomtatni. Az imént felvázolt egyszerű beállítási séma feltételezi, hogy csak olyan állományokat fogunk nyomtatni a nyomtatón, amelyeket meg is ért.

A felhasználók gyakran arra számítanak, hogy bármelyik általuk elérhető nyomtatón képesek nyers szöveget kinyomtatni. Az **LPD** alkalmazással kapcsolatban álló programok is általában ugyanezt az elgondolást követik. Ha egy saját nyelvvel rendelkező nyomtatót akarunk telepíteni, de a nyomtató saját nyelvén és a nyers szöveg formájában érkező nyomtatási feladatok is rendesen ki akarjuk nyomtatni, akkor mindenképpen javasoljuk, hogy illesszünk még egy további lépést is ebbe a sorba: illesszünk a rendszerbe egy nyers szövegről automatikusan PostScript (vagy más egyéb) nyelvre tolmácsoló programot. Erről a Szöveges nyomtatási feladatok PostScript nyomtatókon című fejezetben olvashatunk.

9.3.1.5.1. A nyomtató elnevezése

Az első (egyszerű) lépés a nyomtatónk nevének kiválasztása. Igazából nem számít, mennyire kifejező vagy éppen hóbortos nevet adunk neki, hiszen emellett még számos álnévvel is illethetjük.

Az `/etc/printcap` állományban megtalálható nyomtatók egyikének legalább az `lp` álnévvel rendelkeznie kell, mivel ez lesz az alapértelmezett nyomtató neve. Tehát ha a felhasználó nem adja meg sem a `PRINTER` környezeti változót, sem pedig az **LPD**-vel kapcsolatban álló aktuális parancsban a használni kívánt nyomtató nevét, akkor a rendszer az `lp` nevet fogja keresni.

Ezenkívül általában még gyakran adnak egy olyan álnevet is a nyomtatónak, ahol annak teljes leírása, többek közt a gyártmánya és a típusa szerepel.

Ahogy sikerült nevet és álneveket adni a nyomtatónak, írjuk is be ezeket az `/etc/printcap` állományba. Itt a nyomtató neveit balról kezdjük felsorolni, mindegyik álnevet egy függőleges vonallal válasszunk el, és az utolsó után tegyünk pontosvesszőt.

A most következő példában egy olyan vázlat mutatunk be az `/etc/printcap` állományhoz, amelyben két nyomtatót (egy Diablo 630 márkájú sornyomtatót és egy Panasonic KX-P4455 típusú PostScript lézernyomtatót) adunk meg:

```
#
# /etc/printcap (rose)
#
rattan|line|diablo|lp|Diablo 630 Line Printer:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:
```

Ebben a példában az első nyomtató neve `rattan`, és ehhez tartozik még a `line`, `diablo`, `lp`, és `Diablo 630 Line Printer` álnév. Mivel itt soroltuk fel az `lp` álnevet is, ezért a rendszerben ez lesz az alapértelmezett nyomtató. A második nyomtató neve `bamboo`, és álnevei többek közt a `ps`, `PS`, `S`, `panasonic`, valamint a `Panasonic KX-P4455 PostScript v51.4`.

9.3.1.5.2. A fejléclapok letiltása

Az **LPD** nyomtatási rendszer alapértelmezés szerint minden egyes feladathoz *fejléclapot* készít. Ez a lap szép nagy betűkkel tartalmazza a nyomtatási feladatot kiadó felhasználó nevét, a gépet, amiről küldték, és a feladat nevét. Sajnálatos módon ez azonban inkább akadályozza a hibakeresést a nyomtató beállításában, ezért most inkább kapcsoljuk ki ezeket.

Ha le akarjuk tiltani a fejléclapokat, az `/etc/printcap` állományban adjuk meg az `sh` (úgy mint “suppress header pages”) tulajdonságot. Íme egy példa az `sh` tulajdonsággal bővített `/etc/printcap` állományra:

```
#
# /etc/printcap (rose) - sehol sem lesznek fejléclapok
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:
```

Ebben a példában megfigyelhetjük a helyes felírási módot: az első sor a bal szélső oszlopban kezdődik, az azt követő sorok pedig bentebb. Minden bejegyzésben az utolsó kivételével mindegyik sor egy visszaper (backslash) karakterrel zárul.

9.3.1.5.3. A nyomtatási könyvtár létrehozása

A nyomtatási rendszerünk beállításának következő lépése a *nyomtatási könyvtár* létrehozása. Ez egy olyan könyvtár, ahová a különböző nyomtatási feladatok kerülnek a feldolgozásuk előtt, valamint ahol a nyomtatási rendszer többi állománya lakozik.

A nyomtatási rendszer adatait tároló könyvtárakat tartalmuk gyakori változása miatt általában a `/var/spool` könyvtárba szokás tenni. Ezen könyvtárak tartalmát nem szükséges menteni sem. Az `mkdir(1)` parancs futtatásával egyszerűen újra létre tudjuk hozni.

Általában minden nyomtatóhoz külön létre szoktak hozni egy könyvtárat az adott nyomtató nevéen. Erre példa:

```
# mkdir /var/spool/nyomtatónév
```

Azonban ha a hálózatunkon rengeteg nyomtató található, akkor érdemes inkább egyetlen könyvtárat használni, amelyet az **LPD** számára tartunk fenn.

```
# mkdir /var/spool/lpd
# mkdir /var/spool/lpd/rattan
# mkdir /var/spool/lpd/bamboo
```

Megjegyzés: Amennyiben fontos nekünk a felhasználói nyomtatások titkosságának megóvása, érdemes levédnünk a nyomtatási könyvtárat, így az nem lesz mindenki által elérhető. A nyomtatási könyvtárak tulajdonosa egyedül és kizárólag a `daemon` felhasználó és a `daemon` csoport legyen, és hozzá olvasási, írási és keresési engedélyekkel rendelkezzen. Ezt fogjuk most beállítani a példáinkban szereplő nyomtatóinkhoz is:

```
# chown daemon:daemon /var/spool/lpd/rattan
# chown daemon:daemon /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan
# chmod 770 /var/spool/lpd/bamboo
```

Végezetül az `/etc/printcap` állományban ezeket a könyvtárakat se felejtjük el megadni az **LPD**-nek. Itt a nyomtatási könyvtár nevét az `sd` tulajdonsággal írjuk le:

```
#
# /etc/printcap (rose) - a nyomtatási könyvtárak hozzáadása
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
```

```

:sh:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:

```

Vegyük észre, hogy a nyomtató neve ugyan a sor elején kezdődik, azonban a hozzá tartozó összes többi sor mind bentebb kezdődik és egy visszaper (backslash) karakterrel választjuk le.

Ha az `sd` tulajdonsággal nem adunk meg semmilyen nyomtatási könyvtárat, akkor ennek az értéke alapértelmezés szerint a `/var/spool/lpd` lesz.

9.3.1.5.4. A nyomtatóeszköz azonosítása

A Hardveres beállítás című szakaszban már beazonosítottuk, hogy a FreeBSD a `/dev` könyvtárban melyik eszközeleíron keresztül fogja megszólítani a nyomtatót. Most ideje ugyanezt tudatni az **LPD** démonnal is. Így amikor a nyomtatási rendszer végre szeretne hajtani egy nyomtatási feladatot, a szűrőprogram nevében ezt az eszközt nyitja meg (ahol a szűrőn keresztül továbbítjuk az adatokat a nyomtató felé).

Az `lp` tulajdonság segítségével a `/etc/printcap` állományban soroljuk fel a nyomtatók `/dev` könyvtárban található leíróit.

Az eddig használt példánkban most tételezzük fel, hogy a `rattan` nevű nyomtató az első párhuzamos porton található, míg a `bamboo` nevű a hatodik soros porton. Ebben a helyzetben így kellene kiegészítenünk az `/etc/printcap` állományunkat:

```

#
# /etc/printcap (rose) - a használni kívánt eszközök
# beazonosítása
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
:sh:sd=/var/spool/lpd/rattan:\
:lp=/dev/lpt0:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:\
:lp=/dev/ttyd5:

```

Az **LPD** alapértelmezés szerint a `/dev/lp` eszközt fogja használni, ha nem adjuk meg az `lp` tulajdonságot az `/etc/printcap` állományban. A `/dev/lp` azonban a FreeBSD-ben jelenleg nem létezik.

Ha a telepítendő nyomtatónk valamelyik párhuzamos portra csatlakozik, akkor innen akár tovább is léphetünk A szövegszűrő telepítése című szakaszra. Ha viszont nem, kövessük a most következő szakaszban szereplő utasításokat.

9.3.1.5.5. A nyomtatási rendszer kommunikációs paraméterei

A soros portra csatlakozó nyomtatóknál az **LPD** képes beállítani az adatátviteli sebességet, a paritást, valamint más egyéb olyan kommunikációs paramétereket, amelyekkel a szűrőprogram adatokat tud továbbítani a nyomtató felé. Ez több szempontból is előnyös, mivel:

- Egyszerűen az `/etc/printcap` állomány átírásával ki tudunk próbálni több kommunikációs beállítást, nem kell magát a szűrőprogramot újrafordítanunk.

- A nyomtatási rendszer képes ugyanazt a szűrőt több, különböző kommunikációs beállítást alkalmazó nyomtatóhoz is használni.

Az `/etc/printcap` állományban az `lp` tulajdonsággal megadott eszközök soros kommunikációjának beállításait az alábbi tulajdonságok határozzák meg:

`br#sebesség`

Beállítja az eszköz adatátviteli sebességét a *sebesség* értékre, ahol a *sebesség* lehet 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19 200, 38 400, 57 600 vagy 115 200 bit másodpercenként (bps).

`ms#stty-mód`

Beállítja az eszköz megnyitása után használt termináleszköz működésének paramétereit. Az `stty(1)` man oldalon többet is megtudhatunk róluk.

Miután az **LPD** megnyitja az `lp` tulajdonsággal megadott eszközt, beállítja az `ms#` tulajdonság értéke szerint annak jellemzőit. Itt a `parenb`, `parodd`, `cs5`, `cs6`, `cs7`, `cs8`, `cstopb`, `crtsets`, és `ixon` módok lehetnek lényegesek, melyekről az `stty(1)` man oldalon többet is megtudhatunk.

Állítsuk most be az egyik képzeletbeli nyomtatónkat a hatodik soros portra. Az adatátviteli sebessége 38 400 bps lesz. A kommunikáció módjánál kapcsoljuk ki a paritást (`-parenb`), 8 bites karakterek legyenek (`cs8`), ne legyen modemes vezérlés (`clocal`) és a hardveres forgalomirányítás legyen `crtsets`:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:\
:lp=/dev/ttyd5:ms#-parenb cs8 clocal crtsets:
```

9.3.1.5.6. A szövegszűrő telepítése

Most már utasíthatjuk az **LPD**-t, hogy milyen szövegszűrőt használjon a nyomtatási feladatok eszközre küldéséhez. A *szövegszűrő* (text filter), vagy más néven *bemeneti szűrő* (input filter) egy olyan program, amelyet az **LPD** egy nyomtatási feladat elvégzésekor lefuttat. Amikor az **LPD** lefuttatja a nyomtatóhoz tartozó szövegszűrőt, a szűrő szabványos bemenetére elküldi az elvégzendő nyomtatási feladatot, és a szabványos kimenetét pedig átirányítja az `lp` tulajdonság által megadott nyomtatóeszközre. Ennek megfelelően a szűrőnek a szabványos bemenetről kell olvasnia az elvégzendő feladatot, a szabványos kimenetre pedig a ténylegesen nyomtatandót kell kiírnia. A szövegszűrők részleteiről a *Hogyan működnek a szűrők?* szakasz szól.

A mi esetünkben most szövegszűrőnek tökéletesen megfelel egy olyan rövid szkript, ami a nyomtatóra a nyomtatási feladatot a `/bin/cat` paranccsal küldi ki. A FreeBSD-ben még találhatunk egy másik szűrőt is, amelynek a neve `lpf`. Ez képes a törlést és aláhúzást jelző karaktereket érthetővé tenni bizonyos nyomtatók számára. Természetesen itt használhatunk kedvünk szerinti szűrőt is. Az `lpf` szűrő működésének részleteit Az `lpf` szövegszűrő című szakaszban fejtjük ki bővebben.

Először is készítsünk egy `/usr/local/libexec/if-simple` nevű egyszerű szövegszűrő szkriptet. A kedvenc szövegszerkesztőnkkel írjuk bele a következő sorokat:

```
#!/bin/sh
#
# if-simple - egyszerű szövegszűrő szkript az lpd-hez
# Helye: /usr/local/libexec/if-simple
#
# Egyszerűen átmásolja a kimenetére a bemenetéről érkező adatokat; nem
```

```
# fogad el semmilyen paramétert.

/bin/cat && exit 0
exit 2
```

Tegyük indíthatóvá:

```
# chmod 555 /usr/local/libexec/if-simple
```

Ezután tájékoztassuk róla az **LPD**-t az `/etc/printcap` állományban található `if` tulajdonság megadásával. Itt most a példánkban szereplő mind a két nyomtatóhoz beillesztjük:

```
#
# /etc/printcap (rose) - a szövegszűrő hozzáadása
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:\
    :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:\
    :if=/usr/local/libexec/if-simple:
```

Megjegyzés: Az `if-simple` szkript megtalálható a `/usr/share/examples/printing` könyvtárban.

9.3.1.5.7. Az **LPD** elindítása

Az `lpd(8)` az `/etc/rc` szkriptből, az `lpd_enable` változó értékének megfelelően indul el. Ennek értéke alapból `NO`, vagyis nem. Ha eddig még nem tettük volna meg, akkor az `/etc/rc.conf` állományba most vegyük fel a következő sort:

```
lpd_enable="YES"
```

Ezután vagy indítsuk újra a számítógépünket, vagy pedig adjuk ki az `lpd(8)` parancsot:

```
# lpd
```

9.3.1.5.8. Próbáljuk ki!

Elérkeztünk az **LPD** egyszerű beállításának utolsó lépéséhez. Sajnos azonban még nem gratulálhatunk, hiszen hátra van még a nyomtató kipróbálása és az esetlegesen előforduló hibák kijavítása. A beállítást úgy tudjuk a legegyszerűbben letesztelni, ha megpróbálunk valamit kinyomtatni. Az **LPD** rendszerben az `lpr(1)` parancs használatával tudunk nyomtatási feladatokat kiadni.

A kommunikáció ellenőrzése című szakaszban megtalálhatjuk, hogy hozzunk létre tesztelésre alkalmas szövegeket az `lpr(1)` és az `lpctest(1)` programok segítségével.

Az **LPD** beállításainak egyszerű tesztelése:

Írjuk be:

```
# lptest 20 5 | lpr -Pnyomtatónév
```

ahol a *nyomtatónév* az `/etc/printcap` állományban megadott egyik nyomtató neve (vagy álneve) lehet. Az alapértelmezett nyomtató kipróbálásához ne adjunk meg az `lpr(1)` parancsnak semmilyen `-P` paramétert. Még egyszer megemlítenénk, hogy amennyiben PostScript nyomtatót tesztelünk, az előbbi helyett az `lptest(1)` parancssal küldjünk ki egy PostScript programot. Ehhez tegyük a tesztelő programunkat egy állományba, majd írjuk be az `lpr állománynév` parancsot.

A PostScript nyomtató esetén a kiküldött program eredményét kell látnunk. Amennyiben az `lptest(1)` parancsot használjuk, valami ilyesmire kell számítanunk:

```
! "$%&' ( ) * + , - . / 0 1 2 3 4
" "$%&' ( ) * + , - . / 0 1 2 3 4 5
# "$%&' ( ) * + , - . / 0 1 2 3 4 5 6
$ "%&' ( ) * + , - . / 0 1 2 3 4 5 6 7
% "&' ( ) * + , - . / 0 1 2 3 4 5 6 7 8
```

A nyomtató kimerítőbb teszteléséhez próbáljunk meg nagyobb programokat keríteni valahonnan (ha a nyomtatónk valamilyen nyelven kommunikál) vagy adjunk meg az `lptest(1)` parancsnak más paramétereket. Például az `lptest 80 60` soronként 80 karaktert írat ki 60 sorban.

Amennyiben a nyomtató nem működne, nézzük meg a Hibakereséshez tartozó szakaszt.

9.4. Magasszintű nyomtatóbeállítás

Ebben a szakaszban olyan szűrőket mutatunk be, amelyek speciálisan formázott állományok, fejléclapok, hálózati nyomtatás, nyomtatási nyilvántartás vagy szabályozás esetén használhatóak.

9.4.1. Szűrők

Noha az **LPD** képes hálózati protokollokat, nyomtatási sorokat, hozzáférést és sok minden más nyomtatási feladatot kezelni, a *tényleges* munka legnagyobb része a *szűrőkben* (filter) történik. A szűrők olyan programok, amelyek tartják a kapcsolatot a nyomtatóval és megbirkóznak annak eszközfüggőségeivel és különleges igényeivel. Az egyszerű beállítás során egy primitív szövegszűrőt állítottunk be (lásd A szövegszűrő telepítése) — ami annyira egyszerű, hogy szinte minden nyomtatón működnie kell.

Azonban mindahhoz, hogy ki tudjuk használni a különböző átalakítási, nyilvántartási lehetőségeket, valamint a nyomtatók különlegességeit és egybeikt, meg kell értenünk a szűrők pontos működését. Az előbb említett feladatok ugyanis teljesen a szűrő kezében vannak. Ezzel kapcsolatban azonban rossz hír, hogy ezeket a szűrőket *nekiünk* kell megírunk. A jó hír ellenben az, hogy könnyen találunk ilyen szűrőket, vagy ha éppen nem lennének valamelyiket, akkor is gyorsan meg tudjuk ezeket írni.

Sőt, a FreeBSD alaptól tartalmaz is egyet, amit a `/usr/libexec/lpr/lpf` helyen találunk meg, és sok olyan nyomtatóval képes együttműködni, amelyek nyers szöveget tudnak nyomtatni. (Kezeli az állományokban felbukkanó

törléseket és tabulálásokat, valamint képes nyilvántartást vezetni, de semmi többet.) Rajta kívül még számos szűrőt és szűrőelemet is találhatunk a FreeBSD Portgyűjteményében.

Lássuk, mit tartogat számunkra ez a rész:

- A *Hogyan működnek a szűrők?* című szakaszban megpróbálunk egyfajta áttekintést adni a szűrők nyomtatási folyamatban betöltött szerepéről. Mindenképpen érdemes elolvasnunk ezt a szakaszt, mivel ebben derül ki, hogy valójában mi is történik a “függöny mögött”, vagyis amikor az **LPD** használja ezeket a szűrőket. Ezzel a tudással el tudjuk kerülni vagy éppen nyakon tudjuk csípni azokat a problémákat, amelyek a nyomtatóinkhoz telepített szűrők hozzáadása során adódhatnak.
- Az **LPD** alapból arra számít, hogy minden nyomtató képes nyers szöveget nyomtatni. Ez gondot okoz a PostScript (és minden más nyelv alapú) nyomtatók esetén, mivel azok nem képesek nyers szöveget nyomtatni. Szöveges nyomtatási feladatok PostScript nyomtatókon című szakaszban viszont fény derül rá, hogyan kerekedjünk felül ezen. Feltétlenül olvassuk el, ha PostScript nyomtatónk van.
- A PostScript számos program közkedvelt kimeneti formátuma, sőt gyakran maguk a felhasználók is szeretnek ilyen programokat írni. Sajnos azonban a PostScript nyomtatók egyáltalán nem olcsók. A PostScript szimulációja nem PostScript nyomtatókon című szakaszban megtudhatjuk, miképp tudjuk úgy módosítani a szűrőt, hogy *nem PostScript* nyomtatókon is tudjunk PostScript programokkal nyomtatni. Ezt a szakaszt akkor érdemes elolvasni, ha nincs PostScript nyomtatónk.
- A Konverziós szűrők című szakaszban eláruljuk, miként lehetséges automatizálni a különböző állományformátumok és a nyomtatók által érthető formátumok közti konverziókat, legyen az grafikus vagy betűszedésre vonatkozó adat. A szakasz elolvasása során megismerjük, hogyan tudjuk a nyomtatónkat képessé tenni az `lpr -t` paranccsal troff adatok, vagy a `lpr -d` paranccsal a \TeX DVI állományainak, esetleg az `lpr -v` paranccsal raszteres képek nyomtatására és így tovább. Csak ajánlani tudjuk ennek elolvasását.
- A Kimeneti szűrők című szakaszban kivesézzük az **LPD** egyik kevésbé használt lehetőségét is, a kimeneti szűrőket. Hacsak nem fejléclapokat akarunk készíteni (lásd Fejléclapok), akkor ezt a szakaszt nyugodtan kihagyhatjuk.
- Az *lpf* szövegszűrő szakaszban bemutatásra kerül a FreeBSD-ben alapból megtalálható *lpf* szűrő, amely egy sornyomtatóknál (vagy az így viselkedő lézernyomtatóknál) használható egyszerű szövegszűrő. Ha nyers szövegek nyomtatásánál meg akarjuk oldani a nyomtatási feladatok nyilvántartását, vagy a törlés karakter láttán a nyomtatónk füstölni kezdene, akkor mindenképpen érdemes belemerülnünk az *lpf* titkaiba.

Megjegyzés: A most következő szkriptek mindegyike megtalálható a `/usr/share/examples/printing` könyvtárban.

9.4.1.1. Hogyan működnek a szűrők?

Ahogy már korábban is jeleztük, a szűrő egy olyan végrehajtható program, amelyet az **LPD** indít el, amikor a nyomtatóval eszközfüggetlen módon kommunikál.

Amikor az **LPD** egy feladat elvégzése során ki akar nyomtatni egy állományt, akkor elindít egy ilyen szűrőprogramot. A szűrő szabványos bemenetére elküldi a kinyomtatandó állományt, a szabványos kimenetét a nyomtatóra, a szabványos hibajelzéseit pedig egy naplóállományba irányítja (ez utóbbit az `/etc/printcap` állományban az `lf` tulajdonsággal adhatjuk meg, vagy alapértelmezés szerinti a `/dev/console` állományba kerül).

Az **LPD** a használni kívánt szűrőt és annak paramétereit az `/etc/printcap` állományban felsoroltak vagy az `lpr(1)` parancssorában megadottak szerint választja ki. Például, ha a felhasználó a `lpr -t` parancsot adja ki, akkor az **LPD** a célként megadott nyomtatonál szereplő `tf` tulajdonság által megadott troff szűrőt kezdi el használni. Amennyiben a felhasználó egyszerűen csak nyers szöveget akar nyomtatni, akkor az `if` szűrőnek kellene elindulnia (ez viszont csak részben igaz: lásd Kimeneti szűrők).

Háromfajta szűrő jelenhet meg az `/etc/printcap` állományban:

- A *szövegszűrő* (text filter), ami a hagyományos szöveges nyomtatásért felelős, és amit az **LPD** dokumentációjában érdekes módon *bemeneti szűrőnek* (input filter) hívnak. Mivel az **LPD** arra számít, hogy minden nyomtató alából képes kinyomtatni bármilyen nyers szöveget, ezért a szövegszűrő feladata, hogy a nyomtató számára gondoskodjon a tabulátorok, törlések és más egyéb speciális karakterek megfelelő kezeléséről. Emellett ha olyan helyen vagyunk, ahol szükség van a nyomtatási feladatok nyilvántartására is, a szövegszűrő ennek megoldására is képes, még hozzá úgy, hogy összeszámolja a kinyomtatott sorokat, és elosztja ezeket a nyomtató által oldalanként nyomtatott sorok számával. Egy szövegszűrő a következő paraméterekkel indulhat:

szűrőnév [-c] -w szélesség -l hossz -i behúzás -n hozzáférés -h gépnév nyilvántartás

ahol a

-c

akkor jelenik meg, ha egy nyomtatási feladatot az `lpr -l` paranccsal adunk át

szélesség

az `/etc/printcap` állományban definiált `pw` (page width, avagy oldalszélesség) tulajdonság értéke, ami alapbeállítás szerint 132

hossz

a `pl` (page length, avagy oldalhossz) tulajdonság értéke, amely az alapbeállítás szerint 66

behúzás

az `lpr -i` parancs megadása során használt behúzás mértéke, ami alából 0

hozzáférés

a nyomtatást végző felhasználó hozzáféréseinek megnevezése

gépnév

a gép neve, amiről a nyomtatási feladat érkezett

nyilvántartás

ez a nyilvántartást tároló állomány `af` tulajdonsággal definiált neve

•

A *konverziós szűrők* (conversion filter) egy adott állományformátumot hoznak a nyomtató számára értelmes formára. Például ditroff adatok közvetlenül ugyan nem nyomtathatóak, azonban a ditroff állományokhoz tudunk telepíteni egy olyan szűrőt, amely a ditroff adatokat a nyomtató számára is emészthető és nyomtatható formájává

teszi. A Konverziós szűrők című szakasz tud ezekről többet mondani. Ilyen esetekben kérhetünk nyilvántartást. A konverziós szűrők az alábbi paraméterekkel indulhatnak:

szűrőnév -x pixelszélesség -y pixelmagasság -n hozzáférés -h gépnév nyilvántartás

ahol a *pixelszélesség* a *px* tulajdonság értékéből (ami alaphól 0), a *pixelmagasság* a *py* tulajdonság értékéből (ami alaphól szintén 0) származik.

- A *kimeneti szűrő* (output filter), ami csak akkor aktív, ha a szövegszűrő nem, vagy ha engedélyeztük fejléclapok nyomtatását. Tapasztalatom szerint az ilyen szűrőket ritkán használják. A *Kimeneti szűrők* című szakasz mutatja be a működésüket. Ekkor csupán két paraméterünk van:

szűrőnév -w szélesség -l hosszúság

amik rendre megegyeznek a szövegszűrők *-w* és *-l* paramétereivel.

A szűrők *ki is tudnak lépni* a következő kódokkal (exit status):

0

A szűrő sikeresen kinyomtatta az állományt.

1

A szűrőnek nem sikerült kinyomtatnia az állományt, azonban szeretné, ha az **LPD** újból megpróbálna vele. Az **LPD** tehát ebben az esetben újraindítja a szűrőt.

2

A szűrőnek nem sikerült kinyomtatnia az állományt, és nem is kívánja újra megpróbálni. Ekkor az **LPD** eldobja az állományt.

A FreeBSD kiadásokban megtalálható `/usr/libexec/lpr/lpf` szövegszűrő képes a kapott szélesség és hossz paraméterekkel megállapítani az oldaltöréseket és a nyomtató használatát nyilvántartani, amihez a hozzáférés, gépnév és nyilvántartás adatait használja fel.

Amikor majd igyekszünk mellé újabb szűrőket beszerezni, ne felejtsük el ellenőrizni, hogy együtt tudnak-e működni az **LPD**-vel. Ha a válasz igen, akkor a fentebb említett paraméterek mindegyikét ismerniük kell. Az általános használatra készült szűrők készítése során mi magunknak is be kell tartanunk ezeket az elvárásokat.

9.4.1.2. Szöveges nyomtatási feladatok PostScript® nyomtatókon

Ha csak egyedül dolgozunk a számítógépen és PostScript (vagy bármilyen más nyelvet ismerő) nyomtatónk van, valamint megígérjük, hogy soha nem küldünk sem mi, sem pedig nem küldetünk semmilyen más programmal nyers szöveget a nyomtatóra, akkor átléphetjük ezt a szakaszt.

Ha viszont egyaránt akarunk küldeni PostScript programot és nyers szöveget tartalmazó nyomtatási feladatot a nyomtatónak, akkor ehhez kénytelenek vagyunk a rendszerünket beállítani. Először is szükségünk van szövegszűrőre, ami megállapítja, hogy a frissen érkezett nyomtatási feladat nyers szöveget vagy PostScript programot tartalmaz-e. Minden PostScript-alapú feladat a `%!` karaktersorozattal kezdődik (a többi esetben olvassuk a nyomtató leírását). Szóval, ha a nyomtatandó állomány első két karaktere ilyen, akkor egy PostScript programmal van dolgunk és közvetlenül továbbküldhetjük a nyomtatási feladatot a nyomtatónak. Minden más esetben a szűrőnek előbb át kell alakítania a szöveget PostScript nyelvre.

Hogyan érhetjük el mindezt?

Ha soros nyomtatónk van, akkor erre a feladatra az `lprps` parancs tökéletes. Az `lprps` egy olyan PostScript szűrő, amely mind a két irányban képes közvetíteni. Folyamatosan rögzíti egy állományba a nyomtató állapotát, így a felhasználók és rendszergazdák pontosan látják a nyomtató jelenlegi állapotát (például `toner low` (a toner hamarosan kifogy) vagy `paper jam` (a papír beragadt)). Ami viszont sokkal lényegesebb, hogy a `psif` nevű program képes megmondani az érkező nyomtatási feladat valódi típusát, és ennek megfelelően meg tudja hívni nyers szöveg átalakítására a `textps` (egy másik program, amit a `lprps` mellé kapunk) parancsot. Ezután az `lprps` elküldi a feladatot a nyomtatónak.

Az `lprps` a FreeBSD Portgyűjteményének része (lásd [A Portgyűjtemény](#)), ezért a használni kívánt papír méretétől függően pillanatok alatt magunk is letölthetjük, fordíthatjuk és telepíthetjük a `print/lprps-a4` és `print/lprps-letter` csomagok valamelyikét. Az `lprps` telepítése után egyszerűen csak adjuk meg a `psif` elérési útvonalát. Ha tehát telepítettük a Portgyűjteményből az `lprps` csomagot, akkor egy soros portra csatlakozó PostScript nyomtató esetén ezt kell beírunk az `/etc/printcap` állományba:

```
:if=/usr/local/libexec/psif:
```

Ezenkívül még az `rw` tulajdonsággal meg kell mondanunk az **LPD**-nek, hogy a nyomtatót írásra és olvasásra nyissa meg.

Amennyiben a PostScript nyomtatónk a párhuzamos porton csatlakozik (és amiért a nyomtatónk nem képes az `lprps` által igényelt kétirányú kommunikációra), szövegszűrőként a következő szkriptet fogjuk használni:

```
#!/bin/sh
#
#  psif - PostScript vagy nyers szöveg nyomtatása PostScript nyomtaton
#  Ez a szkriptes változat, NEM pedig az lprps-hez mellékelt szűrő
#  (a /usr/local/libexec/psif állomány)!
#

IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
    #
    #  PostScript: nyomtassuk ki.
    #
    echo "$first_line" && cat && printf "\004" && exit 0
    exit 2
else
    #
    #  Nyers szöveg: alakítsuk át, majd nyomtassuk ki.
    #
    ( echo "$first_line"; cat ) | /usr/local/bin/textps && printf "\004" && exit 0
    exit 2
fi
```

A fentebb szereplő szkriptben a `textps` programot használjuk a nyers szövegek PostScript programokra alakításához, de helyette bármilyen más konvertáló programot is igénybe vehetünk. A FreeBSD Portgyűjteményében (lásd [A Portgyűjtemény](#)) találhatunk erre a célra egy `a2ps` nevű programot is, amit esetleg érdemes lehet közelebbről megnéznünk.

9.4.1.3. PostScript szimulációja nem PostScript nyomtatókon

A PostScript a magas színvonalú betűszedés és nyomtatás *de facto* szabványa. Emellett azonban a PostScript egy *költséges* szabvány is. Az Aladdin Enterprises-nak hála azonban létezik egy hozzá hasonló szabad szoftver, a **Ghostscript**, amely képes FreeBSD-n is futni. A **Ghostscript** képes a legtöbb PostScript állomány olvasására, megjelenítésére mindenféle eszközön, beleértve a PostScriptet nem ismerő nyomtatókat is. A **Ghostscript** és egy speciális szövegszűrő telepítésével el tudjuk érni, hogy egy nem PostScript nyomtató valódi PostScript nyomtatóként viselkedjen.

Ha telepíteni szeretnénk, a **Ghostscript** megtalálható a FreeBSD Portgyűjteményében. Innen tehát magunk is könnyedén le tudjuk tölteni, fordítani és telepíteni.

A PostScript nyomtatás szimulációjához először egy szűrő segítségével észre kell vennünk, hogy egy PostScript formátumú állományt készülünk kinyomtatni. Ha nem ilyen a nyomtatási feladat, akkor egyenesen a nyomtatóra küldjük, azonban minden más esetben először a **Ghostscript** segítségével átalakítjuk egy olyan formátumba, amit a nyomtató is képes feldolgozni.

Nézzünk erre egy példát: a most következő szövegszűrő a Hewlett Packard DeskJet 500-as nyomtatóihoz használható. Más nyomtató esetén cseréljük ki a `gs` (**Ghostscript**) parancs `-sDEVICE` paraméterét a neki megfelelőre. (A telepített **Ghostscript** által ismert nyomtatók listáját a `gs -h` paranccsal kérdezhetjük le.)

```
#!/bin/sh
#
# ifhp - Ghostscripttel szimulált Postscript nyomtatás DeskJet 500-on
# Helye: /usr/local/libexec/ifhp

#
# LF karaktereket CR+LF-ként kezeljük (elkerülve ezzel a HP/PCL
# nyomtatókon a "lépcsőzést"):
#
printf "\033&k2G" || exit 2

#
# Az állomány első két karakterének beolvasása
#
IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
    #
    # Ez PostScript: küldjük át a Ghostscripen és nyomtassuk ki.
    #
    /usr/local/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=djet500 \
        -sOutputFile=- - && exit 0
else
    #
    # Nyers szöveg vagy HP/PCL, ezért küldjük át közvetlenül. Az utolsó
    # lap kidobásához küldünk még egy lapdobást is.
    #
    echo "$first_line" && cat && printf "\033&l0H" &&
exit 0
fi

exit 2
```

Befejezésül az `if` tulajdonságon keresztül értesítenünk kell erről a szűrőről az **LPD**-t is:

```
:if=/usr/local/libexec/ifhp:
```

Készen is vagyunk! Most már nyugodtan beírhatjuk, hogy `lpr sima.szöveg` vagy `lpr akarmi.ps`, mind a kettőnek ki kell tudnia nyomtatódnia.

9.4.1.4. Konverziós szűrők

Miután elvégeztük az Alacsonyszintű nyomtatóbeállítás című szakaszban leírt beállításokat, a (nyers ASCII szöveg mellett) kedvenc állományformátumainkhoz is minden bizonnyal szeretnénk telepíteni néhány konverziós szűrőt.

9.4.1.4.1. Miért használjunk konverziós szűrőket?

A konverziós szűrők segítségével állományok mindenféle formátumait könnyen ki tudjuk nyomtatni. Például tegyük fel, hogy sokat dolgozunk a \TeX betűszedő rendszerrel és egy PostScript nyomtatónk van. Minden alkalommal, amikor egy DVI állományt hozunk létre a \TeX forrásból, azt közvetlenül még nem tudjuk a nyomtatóra küldeni. Ehhez a következő parancsokat kell kiadnunk:

```
% dvips hínár-elemzés.dvi
% lpr hínár-elemzés.ps
```

Ha telepítünk egy konverziós szűrőt a DVI állományokhoz, meg tudjuk spórolni ezt a manuális átalakítási lépést azzal, hogy átadjuk ezt a feladatot az **LPD**-nek. Így ezután mindig, amikor egy DVI állományt akarunk kinyomtatni, csupán egyetlen lépésre lesz szükségünk:

```
% lpr -d hínár-elemzés.dvi
```

Az **LPD**-nek a `-d` paraméterrel adjuk meg, hogy a nyomtatás előtt hajtsa végre a DVI átalakítását. A Formázási és konverziós beállítások című szakaszban találjuk meg a többi konverziós opciót.

Minden olyan konverziós beállításhoz, amit használni szeretnénk a nyomtatóval, telepítenünk kell egy *konverziós szűrőt* (conversion filter) és meg kell adnunk a nevét az `/etc/printcap` állományban. A konverziós szűrők az egyszerű nyomtatóbeállításnál szereplő szövegszűrőkhöz hasonlítanak (lásd A szövegszűrő telepítése szakasz) azzal a kivétellel, hogy a nyers szövegek kinyomtatása helyett ezek a szűrők a nyomtató számára értelmes formátumra alakítják az állományokat.

9.4.1.4.2. Milyen konverziós szűrőket érdemes telepíteni?

Olyan konverziós szűrőket telepítsünk, amelyekre gyakran szükségünk lehet. Ha például sok DVI adatot szeretnénk nyomtatni a jövőben, akkor használjunk DVI konverziós szűrőt, vagy ha sok troff formátumú adatot nyomtatunk, akkor minden bizonnyal jól fog jönni egy troff szűrő.

A következő táblázat foglalja össze azokat a szűrőket, amelyekkel az **LPD** képes együttműködni. Megtudhatjuk, hogy az `/etc/printcap` állományban melyik tulajdonság tartozik hozzájuk és hogyan hívjuk meg ezeket az `lpr` paranccsal:

Állománytípus	Tulajdonság az <code>/etc/printcap</code> állományban	Az <code>lpr</code> kapcsolója
cifplot	cf	-c

Állománytípus	Tulajdonság az <code>/etc/printcap</code> állományban	Az <code>lpr</code> kapcsolója
DVI	<code>df</code>	<code>-d</code>
plot	<code>gf</code>	<code>-g</code>
ditroff	<code>nf</code>	<code>-n</code>
FORTTRAN forrás	<code>rf</code>	<code>-f</code>
troff	<code>tf</code>	<code>-f</code>
raster	<code>vf</code>	<code>-v</code>
nyers szöveg	<code>if</code>	nincs, <code>-p</code> , vagy <code>-l</code>

A példánkban tehát a `lpr -d` parancs használata arra utal, hogy a nyomtatónak az `/etc/printcap` állományból a `df` tulajdonságára van szüksége.

Minden hadakozás ellenére állíthatjuk, hogy a FORTRAN források és a plot által használt szövegek formátuma napjainkra már elavultnak tekinthető. Ezért ezekhez az opciókhoz a saját szűrőinkkel tetszőleges formázási lehetőségeket rendelhetünk. Például, ha Printerleaf (az Interleaf asztali kiadványszerkesztő formátuma) állományokat szeretnénk közvetlenül nyomtatni, akkor valószínűleg nem lesz szükségünk plot állományokra. Ezért a `gf` tulajdonságnak megadhatunk egy Printerleaf konverziós szűrőt, amelyen keresztül aztán a felhasználók az `lpr -g` parancssal Printerleaf állományokat tudnak nyomtatni.

9.4.1.4.3. Konverziós szűrők telepítése

Mivel a konverziós szűrők az alap FreeBSD rendszeren kívülre kerülnek, ezért ezeket minden valószínűség szerint valahol a `/usr/local` könyvtárban találjuk meg. Ezen belül is általában a `/usr/local/libexec` könyvtárban fordulnak elő, mivel ezeket csak az **LPD** futtatja, senki másnak nincs rájuk szüksége.

A konverziós szűrők aktiválásához az `/etc/printcap` állományban egyszerűen adjuk meg az alkalmas tulajdonságoknak megfelelő szűrők elérési útjait.

A példánkban most felveszünk egy DVI konverziós szűrőt a bamboo nevű nyomtatóhoz. Itt ismét láthatjuk a korábban használt `/etc/printcap` állományt, ahol most azonban a bamboo nevű nyomtatónál hozzáadtunk egy `df` tulajdonságot:

```
#
# /etc/printcap (rose) - egy df szűrő hozzáadása a bamboo
# nevű nyomtatóhoz
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
:sh:sd=/var/spool/lpd/rattan:\
:lp=/dev/lpt0:\
:if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:\
:lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
:if=/usr/local/libexec/psif:\
:df=/usr/local/libexec/psdf:
```

A DVI szűrő ebben az esetben a `/usr/local/libexec/psdf` néven elérhető aprócska szkript. Ezt találhatjuk benne:

```
#!/bin/sh
#
# psdf - DVI szűrő PostScript nyomtatóhoz
# Helye: /usr/local/libexec/psdf
#
# Az lpr -d parancs hatására hívódik meg
#
exec /usr/local/bin/dvips -f | /usr/local/libexec/lprps "$@"
```

A szkript a dvips parancsot szűrőként futtatja (az -f paraméterrel) a szabványos bemenetről, ahova a nyomtatási feladatot is kapja. Ezután elindítja az lprps PostScript szűrőt (lásd a Szöveges nyomtatási feladatok PostScript nyomtatókon című szakaszt) az **LPD** által átadott paraméterekkel. Az lprps parancs ezekkel a paraméterekkel tartja nyilván az így kinyomtatott lapokat.

9.4.1.4.4. További példák konverziós szűrőkre

A konverziós szűrők telepítésének nincs bevált receptje, ezért ebben a szakaszban bemutatunk rájuk néhány működő illusztrációt. Ezeket tudjuk felhasználni saját szűrők elkészítésére. Vagy ha megtehetjük, használjuk közvetlenül ezeket.

Ebben a példa szkriptben Hewlett Packard LaserJet III-Si nyomtatókhoz hozunk létre raszteres (pontosabban GIF formátumú) konverziós szűrőt:

```
#!/bin/sh
#
# hpvf - GIF állományokat konvertál át HP/PCL-be, majd kinyomtatja
# Helye: /usr/local/libexec/hpvf

PATH=/usr/X11R6/bin:$PATH; export PATH
giftopnm | ppmtopgm | pgmtopbm | pbmtolj -resolution 300 \
    && exit 0 \
    || exit 2
```

Úgy működik, hogy a GIF állományt először PNM (portable anymap), utána PGM (portable graymap), majd PBM (portable bitmap) formátumúra alakítja, amiből végül LaserJet/PCL-kompatibilis adat lesz.

Ez lesz a hozzá tartozó /etc/printcap állomány:

```
#
# /etc/printcap (orchid)
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:\
    :vf=/usr/local/libexec/hpvf:
```

A most következő szkript a groff betűszedű rendszerből érkező troff adatokat alakítja át a bamboo nevű PostScript nyomtató számára:

```
#!/bin/sh
#
# pstf - a groff troff adatait alakítja PS-re, majd kinyomtatja
# Helye: /usr/local/libexec/pstf
```

```
#
exec grops | /usr/local/libexec/lprps "$@"
```

A szkript az `lprps` parancs segítségével kommunikál a nyomtatóval. Ha a nyomtatónk párhuzamos porton csatlakozik, akkor helyette ezt a szkriptet használjuk:

```
#!/bin/sh
#
# pstf - a groff troff adatait alakítja PS-re, majd kinyomtatja
# Helye: /usr/local/libexec/pstf
#
exec grops
```

Kész is! A szűrő életrekeléséhez mindössze ennyit kell beillesztenünk az `/etc/printcap` állományba:

```
:tf=/usr/local/libexec/pstf:
```

Most pedig jöjjön a FORTRAN szerelmeseinek szívét megmelengető szkript. Ez egy olyan szövegszűrő, amely bármelyik nyers szöveget közvetlenül kezelni tudó nyomtató esetén működik. A `teak` nevű nyomtatóhoz helyezzük be:

```
#!/bin/sh
#
# hprf - FORTRAN szövegszűrő LaserJet 3si-hez
# Helye: /usr/local/libexec/hprf
#

printf "\033&k2G" && fpr && printf "\033&l0H" &&
exit 0
exit 2
```

Az `/etc/printcap` állományban a `teak` nyomtatóhoz a következő sor beírásával tudjuk engedélyezni ezt a szűrőt:

```
:rf=/usr/local/libexec/hprf:
```

Most pedig következzen egy utolsó, de az eddigieknél valamivel összetettebb példa. Ebben a korábban bemutatott `teak` nevű LaserJet nyomtatóhoz fogunk hozzáadni egy DVI szűrőt. Először is következzen a művelet egyszerűbb része: bővítsük ki az `/etc/printcap` állományt a DVI szűrő helyének megadásával:

```
:df=/usr/local/libexec/hpdf:
```

Ezután következzen a nehezebb rész: a szűrő elkészítése. Ehhez szükségünk lesz egy DVI-ről LaserJet/PCL-re alakító programra. A FreeBSD Portgyűjteményében (lásd A Portgyűjtemény) találunk is egyet: a csomag neve `print/dvi2xx`. A csomag telepítésével megkapjuk a nekünk kellő `dvi1j2p` programot, ami képes DVI-t LaserJet IIp, LaserJet III és a LaserJet 2000 típusok által ismert kódokra fordítani.

A `dvi1j2p` felhasználásától függetlenül a `hpdf` néven létrehozni kívánt szűrőnk még így is bonyolult lesz, hiszen a `dvi1j2p` nem tud olvasni a szabványos bemenetről, hanem mindenáron egy állománnyal akar dolgozni. Sőt, olyan állománnyal, amelynek `.dvi` kiterjesztése van, ezért még a `/dev/fd/0` (vagyis a szabványos bemenethez tartozó eszközeleíró) használata is akadályokba ütközik.

Üröm még az örömünkben, hogy a `/tmp` könyvtárat sem tudjuk felhasználni ideiglenes link létrehozására: a szimbolikus linkeket a `bin` felhasználó és csoport birtokolja, a szűrőt pedig a `daemon` felhasználó futtatja. A `/tmp`

könyvtárban ráadásul csak a tulajdonosaik képesek állományokat átnevezni vagy törölni (sticky bit). Ezért a szűrő ugyan létre tudna hozni egy linket, azonban ezt a feladata végeztével nem lesz majd képes törölni, mivel a link egy másik felhasználóhoz tartozik.

Ezért a szűrő az aktuális könyvtárban fogja létrehozni ezt a szimbolikus linket, ami jelen esetünkben a nyomtatási rendszer által használt könyvtár lesz (ezt az `/etc/printcap` állomány `sd` tulajdonságával adjuk meg). Itt remekül el tudják végezni a feladataikat a szűrők, különösen mivel (néha) több hely van itt, mint a `/tmp` könyvtárban.

Végül lássuk magát a szűrőt:

```
#!/bin/sh
#
# hpdf - DVI adat nyomtatása HP/PCL nyomtatón
# Helye: /usr/local/libexec/hpdf

PATH=/usr/local/bin:$PATH; export PATH

#
# Létrehozunk egy függvényt az átmeneti állományok törlésére. Ezek
# az aktuális könyvtárban jönnek létre, ami pedig a nyomtatási
# rendszer adott nyomtatóhoz tartozó könyvtára lesz.
#
cleanup() {
    rm -f hpdf$$dvi
}

#
# Létrehozunk egy függvényt a súlyos hibák kezelésére: írassunk ki
# egy adott üzenetet és lépünk ki a 2-es hibakóddal. Ezzel üzenünk
# az LPD-nek, hogy ne hajtsa végre újra a nyomtatási feladatot.
#
fatal() {
    echo "$@" 1>&2
    cleanup
    exit 2
}

#
# Ha a felhasználó eltávolítja a nyomtatási feladatot a sorból, akkor az
# LPD egy SIGINT jelzést fog küldeni, ezért próbáljuk meg azt elkapni
# (néhány más egyéb jelzéssel együtt), így még tudjuk törölni az
# ideiglenesen # létrehozott állományokat.
#
trap cleanup 1 2 15

#
# Gondoskodjunk róla, hogy a feladat megkezdésekor még egyetlen
# használt állomány sem létezik.
#
cleanup

#
# Kössük össze a szabványos bemenetet egy DVI állománnyal (amit
# majd nyomtatni akarunk).
```

```
#
ln -s /dev/fd/0 hpdf$$ .dvi || fatal "Cannot symlink /dev/fd/0"

#
# LF = CR+LF
#
printf "\033&k2G" || fatal "Cannot initialize printer"

#
# Alakítsuk át az adatot és nyomtassunk. A dvilj2p által visszaadott érték
# nem túlságosan megbízható, ezért ne is foglalkozzunk vele.
#
dvilj2p -M1 -q -e- dfhp$$ .dvi

#
# Takarítsunk el magunk után és lépjünk ki szabályosan
#
cleanup
exit 0
```

9.4.1.4.5. Automatikus konverziók: a konverziós szűrők helyett

A konverziós szűrők sokat segítenek egy kényelmes nyomtatási környezet kialakításában, azonban a használatukhoz a felhasználóknak (az `lpr(1)` parancson keresztül) egyenként hivatkozniuk kell rájuk. Ha a rendszerünk felhasználói nem eléggé műveltek számítástechnikai téren, akkor még egy szűrő megadása is zavaró lehet számukra. Ami még ennél is rosszabb, hogy egy rosszul megadott szűrő hatására a nyomtató sem fogja jól kezelni az adott állomány formátumát és erre válaszul akár többszáz lapot is pillanatok alatt kiköphet magából.

A konverziós szűrők telepítése helyett gyakran csak egy (alapértelmezett) szövegszűrőre van szükségünk, amely kideríti a nyomtatandó állomány pontos formátumát és magától elindítja a neki megfelelő konverziós szűrőt. Ilyen esetekben például a `file` parancs pont a hasznunkra válhat. Persze *bizonyos* állománytípusok közt nagyon nehéz különbséget tenni — de ezekre továbbra is adhatunk még külön konverziós szűrőket.

A FreeBSD Portgyűjteményében találhatunk egy `apsfilter` elnevezésű szövegszűrőt (`print/apsfilter`), ami képes ilyen automatikus konverzióra. Képes felismerni a nyers szöveget, PostScript programokat, DVI és szinte bármilyen formátumú állományokat, lefuttatni rájuk a megfelelő átalakításokat, majd kinyomtatni ezeket.

9.4.1.5. Kimeneti szűrők

Az **LPD** nyomtatási rendszer kezel egy eddig még nem tárgyalt szűrőtípust is: ez a kimeneti szűrő. A kimeneti szűrő a szövegszűrőhöz hasonlóan csak nyers szöveg nyomtatására használatos, de tartalmaz néhány egyszerűsítést. Ha kizárólag csak kimeneti szűrőket alkalmazunk, akkor:

- Az **LPD** az egész nyomtatási feladathoz egyetlen kimeneti szűrőt fog használni, nem pedig minden állományhoz külön.
- Az **LPD** a kimeneti szűrő számára nem nyújt semmilyen segítséget a nyomtatási feladaton belül szereplő állományok kezdetének vagy végének megállapításában.

- Az **LPD** a szűrőnek nem adja át sem a felhasználó hozzáférését, sem pedig gépnevét, ezért nyilvántartásra nem alkalmas. Mindent összegezve lényegében csak két paramétert kap meg:

`szűrőnév -wszélesség -lhossz`

ahol a *szélesség* a kérdéses nyomtató `pw` tulajdonságából, a *hossz* pedig a `p1` tulajdonságából származik.

Ne bővíljön el minket a szűrő egyszerűsége! Ha például a nyomtatási feladatban minden állományt újabb lapon szeretnénk kezdeni, akkor azt kimeneti szűrővel *nem tudjuk megoldani*. Erre a célra használunk szövegszűrőt (másik nevén bemeneti szűrőt), lásd A szövegszűrő telepítése szakaszt. Továbbá, a kimeneti szűrő valójában *sokkal bonyolultabb* abban a tekintetben, hogy a beérkező adatok közül neki kell kikeresnie a speciális jelentéssel bíró karaktereket ugyanúgy, ahogy az **LPD** helyett saját magának kell küldenie a jelzéseket.

Azonban a kimeneti szűrők használata *elkerülhetetlen*, ha például fejléclapokat akarunk nyomtatni, és esetleg még különböző inicializálásra használatos speciális kódokat vagy karakterláncokat akarunk ez előtt kiküldeni. (Ellenben *badarság* a fejléclapoktól követelni a felhasználó adatait, hiszen az **LPD** a kimeneti szűrőnek nem ad semmilyen erre vonatkozó információt.)

Egyetlen nyomtató esetén az **LPD** egyaránt lehetővé teszi kimeneti, szöveg- és más egyéb szűrők használatát. Ilyenkor az **LPD** a kimeneti szűrőn keresztül csak a fejléct tartalmazó oldal (lásd a Fejléclapok szakaszt) nyomtatását indítja el. Ezt követően az **LPD** arra számít, hogy a kimeneti szűrő két karakter, az ASCII 031 és az ezt követő ASCII 001, hatására *leállítja magát*. Amikor tehát a kimeneti szűrő érzékeli ezt a két karaktert (031, 001), akkor a SIGSTOP jelzéssel le kell állnia. Miután az **LPD** lefuttatta a többi szűrőt, a SIGCONT jelzéssel újraindítja a kimeneti szűrőt.

Ha van kimeneti szűrőnk, de *nincs* szövegszűrőnk, akkor az **LPD** minden további feldolgozás nélkül továbbadja a nyomtatási feladatot a kimeneti szűrőnek. Ahogy már korábban is említettük, a kimeneti szűrő a nyomtatási feladatban levő összes állományt egymás után nyomtatja ki, lapdobások vagy bármilyen más papírmozgatás nélkül, ezért valószínűleg *nem* ez kell nekünk. Az esetek túlnyomó részében ehhez elég egy szövegszűrő.

A korábban szövegszűrőként beharangozott `lpf` program kimeneti szűrőként is képes funkcionálni. Ha szükségünk lenne egy gyorsan összezsapható kimeneti szűrőre, és nem akarunk a speciális karakterek, valamint a jelzések küldésével elidőzni, akkor próbálkozzunk az `lpf` használatával. Az `lpf` parancsot mellesleg becsomagolhatjuk egy olyan szkriptbe is, amely elvégzi a nyomtató számára szükséges inicializálást.

9.4.1.6. Az `lpf` szövegszűrő

A FreeBSD bináris terjesztéséhez mellékelt `/usr/libexec/lpr/lpf` program egy szövegszűrő (bemeneti szűrő), amely képes (az `lpr -i` parancssal hozzáadott nyomtatási feladatokat) tabulálni, (az `lpr -l` parancssal felvett nyomtatási feladatokban) a vezérlőkaraktereket figyelmen kívül hagyni, a nyomtatási feladatban előforduló törlések és behúzások nyomtatási pozícióját igazítani és nyilvántartani a kinyomtatott lapokat. Kimeneti szűrőként is tud viselkedni.

Az `lpf` szűrő rengeteg nyomtatási környezetben felhasználható. Habár nem képes a nyomtatónak inicializáló jelsorozatokat küldeni, mégis könnyű olyan szkriptet írni, amely elvégzi ezeket a hiányzó kezdeti beállításokat, majd lefuttatja az `lpf` szűrőt.

Az `lpf` akkor lesz képes helyesen számolni a kinyomtatott lapokat, ha ehhez az `/etc/printcap` állományban jól töltjük ki a `pw` és `p1` tulajdonságokat. Ezen értékek segítségével határozható meg ugyanis, hogy mennyi szöveg fért rá egy lapra és így mennyi lapot emésztett fel az adott felhasználó által küldött nyomtatási feladat. A nyomtatás nyilvántartásával kapcsolatban A nyomtató használatának nyilvántartása című szakaszt érdemes elolvasni.

9.4.2. Fejléclapok

Ha *nagyon sok* felhasználónk van, és sok különböző nyomtatót is használnak, akkor előbb vagy utóbb minden bizonnyal elkerülhetetlenné fog válni a *fejléclapok* használata.

A fejléc-, vagy más néven *munka-* vagy *elválasztó lapok* segítik elő az elvégzett nyomtatási feladatok azonosítását. A többi dokumentumtól eltérő módon, általában dekoratív keretben, nagy, vastag betűkkel nyomtatódnak ki, hogy a halomnyi papír között a felhasználók könnyedén megtalálhassák az elküldött nyomtatási feladataik eredményét. Természetesen a fejléclapok nyilvánvaló hátulütője, hogy így minden nyomtatási feladathoz még egy lappal többet kell elhasználni és mivel gyakorlatilag néhány percnél tovább nincs is rájuk szükség, meglehetősen hamar a kukába kerülnek. (A fejléclapok nyomtatási feladatonként jönnek létre, nem pedig a nyomtatási feladatokban levő állományokhoz egyenként, ezért nem is akkora pazarlás ez.)

Az **LPD** rendszer képes magától fejléclapokat készíteni a nyomtatásokhoz, *amennyiben* a nyomtatónk képes közvetlenül nyers szöveget nyomtatni. Ha PostScript nyomtatónk van, akkor ennek legyártásához egy külső programra van szükségünk, lásd a Fejléclapok PostScript nyomtatókon szakaszt.

9.4.2.1. A fejléclapok engedélyezése

Az Alacsonyszintű nyomtatóbeállítás című szakaszban az `/etc/printcap` állományban a `sh` (úgy mint “suppress header”) tulajdonsággal kikapcsoltuk a fejléclapokat. A fejléclapok engedélyezéséhez mindössze el kell távolítanunk ezt az `sh` tulajdonságot.

Ez túl egyszerű, nemde?

Igen, ez így van. *Előfordulhat*, hogy szükségünk van még egy olyan kimeneti szűrőre is, amely inicializáló karaktereket küld a nyomtatónak. Íme egy példa ehhez a Hewlett Packard PCL-kompatibilis nyomtatói esetére:

```
#!/bin/sh
#
# hpof - Kimeneti szűrő Hewlett Packard PCL-kompatibilis nyomtatókhoz
# Helye: /usr/local/libexec/hpof

printf "\033&k2G" || exit 2
exec /usr/libexec/lpr/lpf
```

Az `of` tulajdonsággal adjuk meg a kimeneti szűrőt. A **Kimeneti szűrők** szakaszban erről részletesebben is olvashatunk.

A korábban ismertetett `teak` nevű nyomtatóhoz most az alábbi minta `/etc/printcap` állományt mellékeljük. Itt engedélyeztük a fejléclapokat és hozzátettük az iménti kimeneti szűrőt:

```
#
# /etc/printcap (orchid)
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:\
    :vf=/usr/local/libexec/hpvf:\
    :of=/usr/local/libexec/hpof:
```

Mostantól kezdve, amikor a felhasználók a `teak` nyomtatón akarnak nyomtatni, minden nyomtatási feladathoz kapni fognak egy fejléclapot. Amennyiben a kedves felhasználók mégis keresgetni akarják a nyomtatásaikat, az `lpr -h`

paranccsal tetszőleges módon letilthatják azokat. Az `lpr(1)` többi hasonló opcióját A fejléclapokhoz tartozó beállítások szakaszban találjuk.

Megjegyzés: Az **LPD** minden fejléclap után egy lapdobást küld. Ha erre a célra a nyomtatónk egy eltérő karaktert vagy karaktersorozatot használ, akkor azt az `/etc/printcap` állomány `ff` tulajdonságával határozhatjuk meg.

9.4.2.2. A fejléclapok vezérlése

A fejléclapok engedélyezésével az **LPD** egy ún. *hosszú fejléctet* fog készíteni, vagyis a felhasználót, a gépet és a nyomtatási feladatot jól azonosító, egész lapot kitöltő óriási betűket. Erre egy példa (amiben a `rose` nevű gépről `kelly` küldte az “outline” elnevezésű nyomtatási feladatot):

```

k          ll      ll
k          l       l
k          l       l
k  k      eeee    l       l       y       y
k  k      e   e   l       l       y       y
k k      eeeee   l       l       y       y
kk k      e       l       l       y       y
k  k      e   e   l       l       y      yy
k  k      eeee   ll      ll      yyy y
          y
          y       y
          yyy
          yyy

          ll
          t       l       i
          t       l
o o o o  u   u   ttttt  l       ii      n nnn      eeee
o   o   u   u   t       l       i       nn      n   e   e
o   o   u   u   t       l       i       n       n   eeeee
o   o   u   uu   t  t   l       i       n       n   e   e
o o o o  uuu u   tt      ll      iii      n       n   eeee

r rrr      oooo      ssss      eeee
rr  r      o   o   s   s   e   e
r          o   o   ss      eeeee
r          o   o   ss      e
r          o   o   s   s   e   e

```

```
r          oooo      ssss      eeee
```

Job: outline

Date: Sun Sep 17 11:04:58 1995

Ezt követően az **LPD** elküld még egy lapdobást is, ezért maga a nyomtatási feladat eredménye egy új oldalon fog kezdődni (kivéve, ha az `/etc/printcap` állományban az adott nyomtatóhoz tartozó bejegyzésben megadtuk az `sf` (úgy mint “suppress form feeds”, vagyis a lapdobások letiltása) tulajdonságot.

Ha úgy jobban tetszik, akkor az `/etc/printcap` állományban a `sb` tulajdonsággal az **LPD** utasítható *rövid fejlécek* készítésére is. Ilyenkor a fejléclap tartalma mindössze ennyi lesz:

```
rose:kelly Job: outline Date: Sun Sep 17 11:07:51 1995
```

Alapértelmezés szerint az **LPD** először a fejléclapot, majd a nyomtatási feladatot végzi el. Ezt a sorrendet az `/etc/printcap` állományban a `hl` (header last) tulajdonsággal meg tudjuk fordítani.

9.4.2.3. A nyomtató használatának nyilvántartása

Az **LPD** által felkínált fejléclapok használata során egyetlen irányelv érvényesül a nyilvántartásukban: a fejléclapok *költségmentesek*.

De miért?

Azért, mert kizárólag csak a kimeneti szűrő képes a fejléclapok viselkedését irányítani, ami viszont nem képes semmiféle nyilvántartásra, hiszen nem kapja meg az ehhez szükséges *felhasználói- vagy gépnév* információkat, illetve nyilvántartásokat. Emiatt fogalma sincs róla, hogy kit terhel az adott nyomtató használata. Úgy sem tudjuk megoldani a problémát, ha a szöveg- vagy konverziós szűrőkben (ahol már rendelkezésünkre állnak a felhasználó és a gépének adatai) “növeljük a lapok számát eggyel” a nyomtatási feladatban, mivel a felhasználók az `lpr -h` parancs használatával kedvük szerint letilthatják a fejléclapokat. Ezt ugyan alapvetően a természetet óvni kívánó felhasználók részesítik előnyben, de ettől függetlenül sem erőszakolhatjuk rá mindenkire.

Az *sem elég*, ha minden szűrő létrehozza a saját fejlécét (amiért aztán pénzt kérhetnénk). Mivel ha a felhasználók az `lpr -h` paranccsal le akarják tiltani a fejlécek használatát, attól a szűrőkhöz még mindig létrejönnek, hiszen az **LPD** a `-h` opcióról semmilyen értesítést nem küld át a szűrőknek.

Nos, ilyenkor mitévők legyünk?

A lehetőségeink:

- Elfogadjuk az **LPD** elvét, és nem számítunk fel költséget a fejléclapokra.
- Az **LPD** helyett egy másik nyomtatási rendszert használunk, például az **LPRng** rendszert. A Más nyomtatási rendszerek című szakaszban kiderül, milyen alternatívák érhetőek el az **LPD** kiváltására.
- Írjunk mi magunk egy *intelligens* kimeneti szűrőt. Normális esetben a kimeneti szűrők nem valók másra, csupán a nyomtató alaphelyzetbe hozására vagy egyszerűbb karakterkonverziók elvégzésére. Fejléclapokhoz és nyers szöveget tartalmazó nyomtatási feladathoz remekül használható (ahol nincs szöveg- (avagy bemeneti) szűrő).

Azonban ha a nyers szövegekhez van szövegszűrőnk, akkor az **LPD** a kimeneti szűrőt csak a fejléclapokhoz indítja el. Emellett a kimeneti szűrő az **LPD** által generált fejléc szövegéből képes megmondani, melyik felhasználóhoz és géphez tartozik a szóbanforgó fejléc. A módszer egyetlen bökkenője, hogy a nyilvántartásokat tároló állományról viszont még így se tudunk semmilyen információt szerezni (mivel nem kapjuk meg az `af` tulajdonsággal beállított állomány nevét). Ha azonban egy rendszerszinten elérhető állományba mentjük ezeket az adatokat, akkor akár bele is drótozhatjuk ezt a kimeneti szűrőbe. A kimeneti szűrőnek az adatok megtalálásában ilyenkor úgy tudunk segíteni, ha az `/etc/printcap` állományban az `sh` (rövid fejléc) tulajdonságot állítjuk be. De ez igazából sok hűhó semmiért, és a felhasználók is jobban megbecsülik az olyan nagylelkű rendszergazdát, aki nem számítja fel nekik a fejléclapokat.

9.4.2.4. Fejléclapok PostScript nyomtatókon

Ahogy arról már korábban is szó esett, az **LPD** képes többféle nyomtató számára is megfelelő, nyers szövegű fejléclapokat készíteni. Persze a PostScript közvetlenül nem képes nyers szövegek nyomtatására, ezért az **LPD** ezen lehetősége lényegében használhatatlan — többnyire.

Ilyen helyzetben a fejléclapok használatának nyilvánvaló módja, hogy minden szövegszűrőt fejlécek gyártására utasítunk. Ezek a szűrők a felhasználóról és a gépről kapott információkból össze tudják állítani a megfelelő fejléclapot. A megoldás hátránya, hogy ez még olyankor is megtörténik, amikor a felhasználók az `lpr -h` paranccsal küldik a nyomtatási feladataikat.

Kísérletezzünk egy kicsit ezzel a módszerrel! A most következő szkript három paramétert fogad el (a felhasználó hozzáférést, a gép és a nyomtatási feladat nevét), majd ezekből létrehoz egy egyszerű PostScript formátumú fejléct:

```
#!/bin/sh
#
# make-ps-header - PostScript fejléc létrehozása a szabvány kimenetre
# Helye: /usr/local/libexec/make-ps-header
#
#
# Ezek itt a PostScript által használt egységekben vannak megadva
# (72/col vagy 28/cm). Írjuk át az általunk használt papírméretre,
# A4-re vagy amit éppen használunk:
#
page_width=612
page_height=792
border=72
#
# A paraméterek ellenőrzése.
#
if [ $# -ne 3 ]; then
    echo "Usage: `basename $0` <user> <host> <job>" 1>&2
    exit 1
fi
#
# Mentsük el ezeket, leginkább az olvashatóság miatt.
#
user=$1
host=$2
```

```

job=$3
date='date'

#
# Küldjük el a PostScript-kódot a szabványos kimenetre.
#
exec cat <<EOF
%!PS

%
% Gondoskodjunk róla, hogy ne zavarjuk az utánunk következő
% felhasználó nyomtatási feladatának végrehajtását.
%
save

%
% Csináljunk egy csúf vastag szegélyt, körbe a papíron.
%
$border $border moveto
$page_width $border 2 mul sub 0 rlineto
0 $page_height $border 2 mul sub rlineto
currentscreen 3 -1 roll pop 100 3 1 roll setscreen
$border 2 mul $page_width sub 0 rlineto closepath
0.8 setgray 10 setlinewidth stroke 0 setgray

%
% Jelenítsük meg a felhasználó azonosítóját szép, feltűnő
% betűkkel.
%
/Helvetica-Bold findfont 64 scalefont setfont
$page_width ($user) stringwidth pop sub 2 div $page_height 200 sub moveto
($user) show

%
% Most pedig mutassuk az unalmas részleteket.
%
/Helvetica findfont 14 scalefont setfont
/y 200 def
[ (Job:) (Host:) (Date:) ] {
200 y moveto show /y y 18 sub def }
forall

/Helvetica-Bold findfont 14 scalefont setfont
/y 200 def
[ ($job) ($host) ($date) ] {
270 y moveto show /y y 18 sub def
} forall

%
% Ennyi lett volna.
%
restore
showpage

```

EOF

Ezzel a szkripttel pedig mindegyik konverziós és szövegszűrő először létrehoz egy fejléclapot, majd elvégzi a felhasználó nyomtatási feladatát. Íme egy korábban már bemutatott DVI szűrő, amit most kiegészítünk a fejléclapok használatával:

```
#!/bin/sh
#
# psdf - DVI szűrő PostScript nyomtatóhoz
# Helye: /usr/local/libexec/psdf
#
# Az lpr -d parancs hatására hívódik meg.
#

orig_args="$@"

fail() {
    echo "$@" 1>&2
    exit 2
}

while getopts "x:y:n:h:" option; do
    case $option in
        x|y) ;; # Ignore
        n)    login=$OPTARG ;;
        h)    host=$OPTARG ;;
        *)    echo "LPD started `basename $0` wrong." 1>&2
              exit 2
              ;;
    esac
done

[ "$login" ] || fail "No login name"
[ "$host" ] || fail "No host name"

( /usr/local/libexec/make-ps-header $login $host "DVI File"
  /usr/local/bin/dvips -f ) | eval /usr/local/libexec/lprps $orig_args
```

Láthatjuk, hogy a szűrőnek a felhasználói- és a gépnév megállapításához végig kell néznie a paraméterek listáját. Ez lényegében minden más konverziós szűrőnél ugyanígy néz ki. Ez a lista azonban a szövegszűrők esetén némileg eltér (lásd a *Hogyan működnek a szűrők?* szakaszt).

Már az előbbieken is tárgyaltuk, hogy ez a megoldás, habár eléggé egyszerű, az lpr számára nem teszi lehetővé a fejléclapok letiltását (a -h opció). Ha a felhasználóink kímélni akarják a fákat (vagy meg akarják úszni a fejléclapok égbeszökő költségeit), akkor ezt nem tudják megtenni, hiszen a szűrők minden nyomtatási feladathoz készíteni fognak fejléceket.

Ezt a korlátozást csak úgy tudjuk elsöpörni, ha bevetjük a A nyomtató használatának nyilvántartása szakaszban leírt cselt, tehát készítünk egy olyan kimeneti szűrőt, amely megkeresi az LPD-vel generált fejléceket és létrehozza azok PostScript változatát. Ha valaki az lpr -h parancssal küld nyomtatnivalót, akkor **LPD** nem készít hozzá fejléclapot, ahogy a kimeneti szűrőnk sem. A kimeneti szűrő minden más esetben beolvassa az **LPD** által küldött szöveget és átküldi a neki megfelelő PostScript kódot a nyomtatóra.

Ha soros PostScript nyomtatónk van, akkor használhatjuk a `psof` kimeneti szűrőhöz tartozó `lprps` parancsot is, ami pontosan az előbbit végzi el. Hozzátennénk azonban, hogy a `psof` nem számolja a fejléclapokat.

9.4.3. Hálózati nyomtatás

A FreeBSD tud hálózaton is nyomtatni, vagyis tud távoli számítógépeknek is nyomtatási feladatot küldeni. A hálózati nyomtatás kifejezés általánosságban véve két különböző dologra utalhat:

- Egy távoli számítógéphez kapcsolt nyomtató hozzáférését. A géphez a nyomtató a hagyományos soros vagy párhuzamos csatlófelületen keresztül kapcsolódik, amit aztán az **LPD** alkalmas beállításával a hálózaton mindenki számára elérhetővé teszünk. A Távoli számítógépekre csatlakoztatott nyomtatók című szakasz erről szól.
- Egy közvetlenül a hálózatra kapcsolt nyomtató hozzáférését. A nyomtató tehát rendelkezik még egy hálózati csatlakozással is a hagyományos soros vagy párhuzamos felület mellett (vagy éppen helyett). Egy ilyen nyomtató a következőképpen működhet:
 - Elfogadja az **LPD** kéréseit, és még képes nyomtatási feladatokat is tárolni. Ebben az esetben teljesen egyenértékű egy **LPD** alkalmazást futtató számítógéppel. Ekkor nincs más teendőnk, csak követnünk kell a Távoli számítógépeken telepített nyomtatók című szakasz utasításait.
 - Hálózati adatfolyamokkal dolgozik. Ebben az esetben a nyomtatót “hozzá kell kapcsolnunk” a hálózaton található egyik számítógéphez, ami majd a nyomtatási feladatok tárolásáért és folyamatos küldéséért lesz felelős. A Nyomtatók hálózati adatcsatlakozással szakasz az ilyen fajtájú nyomtatók telepítésére tesz néhány javaslatot.

9.4.3.1. Távoli számítógépekre csatlakoztatott nyomtatók

Az **LPD** nyomtatási rendszer alpból képes más, szintén **LPD**-t (vagy vele kompatibilis rendszert) futtató számítógépekre nyomtatási feladatokat küldeni. Ezzel lényegében az egyik géphez hozzá tudunk kapcsolni egy nyomtatót, amit aztán a többiek számára elérhetővé teszünk. Ez olyan nyomtatók esetében is működik, amelyek ismerik az **LPD** által alkalmazott protokollt.

A távoli nyomtatáshoz először telepítsük a nyomtatót valamelyik számítógépre az Alacsonyszintű nyomtatóbeállítás szakaszban leírtak szerint, és ezzel az lesz a *nyomtatószerverünk*. Ezután, amennyiben szükségesnek találjuk, végezzünk magasabb szintű nyomtatóbeállításokat is. Ne felejtjük el kipróbálni a nyomtatón, hogy rendesen működik az **LPD** mindegyik olyan beállításával, amit engedélyeztünk. Emellett gondoskodjunk minden olyan jogosultságról is, amivel a *helyi számítógépről* el tudjuk érni a *távoli számítógép* által felkínált **LPD** szolgáltatást (lásd Távoli számítógépekről érkező kérések szabályozása).

Ha olyan nyomtatót használunk, aminek a hálózati felülete kompatibilis az **LPD** rendszerrel, akkor az előbb említett *nyomtatószerver* lényegében maga lesz a nyomtató, valamint a *nyomtató neve* a rajta beállított név. Ezzel kapcsolatban olvassuk el a nyomtatóhoz és/vagy a hálózati csatlójához mellékelte dokumentációt.

Tipp: Amikor a Hewlett Packard Laserjet típusú nyomtatóit használjuk, a `text` nevű nyomtatónév magától elvégzi a LF és CRLF formátumú sortörések közti átalakítást, ezért ilyenkor nincs szükségünk a `hpif` szkriptre.

Ezután ha szeretnénk más gépek részére is elérhetővé tenni a frissen telepített nyomtatónkat, adjuk meg mindegyikük `/etc/printcap` állományában a következőket:

1. Tetszőlegesen választott nevet, álneveket. Az egyszerűség kedvéért azonban itt érdemes ugyanazokat a neveket választani, mint amit a nyomtatószerveren is használunk.
2. Szándékosan hagyjuk az `lp` tulajdonságot üresen (`:lp=:`).
3. Hozzunk létre egy nyomtatási könyvtárat, és jelöljük meg a helyét az `sd` tulajdonsággal. Az **LPD** itt fogja összegyűjteni a nyomtatási feladatokat, mielőtt elküldené azokat a nyomtatószervernek.
4. Adjuk meg a nyomtatószerver nevét az `rm` tulajdonság segítségével.
5. Az `rp` tulajdonsággal adjuk meg a *nyomtatószerverre* csatlakoztatott nyomtató nevét.

Kész! Az `/etc/printcap` állományban már nem kell megadni konverziós szűrőket, oldalbeállításokat és semmi más egyebet.

Lássunk mindezekre egy példát. A *rose* nevű számítógéphez két nyomtató csatlakozik, a *bamboo* és a *rattan*. Most pedig beállítjuk, hogy az *orchid* nevű gép felhasználói képesek legyenek ezekkel a nyomtatókkal dolgozni. Ekkor a most következők szerint fog kinézni az *orchid* (a *Fejléclapok engedélyezése* szakaszban bemutatott) `/etc/printcap` állománya. Tartalmazza a *teak* nevű nyomtató beállításait is, és ehhez fogjuk hozzáadni a *rose* másik két nyomtatóját:

```
#
# /etc/printcap (orchid) - a rose két (távoli) nyomtatójának
# hozzáadása
#

#
# A "teak" egy helyi nyomtató, közvetlenül az orchidhoz
# csatlakozik:
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/lfhp:\
    :vf=/usr/local/libexec/vfhp:\
    :of=/usr/local/libexec/ofhp:

#
# A "rattan" rose-hoz csatlakozik, így küldhetünk neki nyomtatási
# feladatot:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

#
# A "bamboo" is a rose-hoz tartozik:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:
```

Ezután már csak létre kell hoznunk a megfelelő nyomtatási könyvtárakat az *orchid* nevű gépen:

```
# mkdir -p /var/spool/lpd/rattan /var/spool/lpd/bamboo
```

```
# chmod 770 /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chown daemon:daemon /var/spool/lpd/rattan /var/spool/lpd/bamboo
```

Mostantól kezdve az orchid felhasználói képesek lesznek nyomtatni a rattan és bamboo nevű nyomtatókon is. Ezért, ha az orchid egyik felhasználója beírja, hogy:

```
% lpr -P bamboo -d sushi-leírás.dvi
```

Az orchid gépen működő **LPD** rendszer ezt a nyomtatási feladatot a bemásolja a /var/spool/lpd/bamboo nevű nyomtatási könyvtárba és feljegyzi róla, hogy a nyomtatásához DVI szűrőre lesz szükség. Ahogy rose gépen található bamboo nyomtatási könyvtárában elegendő hely keletkezik, a két **LPD** átküldi egymás közt a rose nevű gépre az állományt. Ezután az állomány egészen addig várakozik a rose nyomtatási sorában, amíg végezetül kinyomtatásra nem kerül. A rose fogja átalakítani DVI-ről PostScript formátumra átalakítani (mivel a bamboo egy PostScript nyomtató).

9.4.3.2. Nyomtatók hálózati adatcsatlakozással

Amikor hálózati kártyát vásárolunk a nyomtatónkhoz, általában két változatukkal találkozhatunk: az egyikük nyomtatási rendszerként működik (ez a drágább), a másikuk pedig egyszerűen csak soros vagy párhuzamos csatlakozón továbbítandó adatként közvetíti az adatokat a nyomtató felé (az olcsóbb). A drágábbik változatot az előző, Távoli számítógépekre csatlakoztatott nyomtatók című szakaszban leírtak szerint tudjuk használni.

Az /etc/printcap állományban ugyan meg tudjuk adni, hogy a nyomtató soros vagy párhuzamos portra csatlakozik, és azon keresztül milyen adatátviteli sebességgel (amennyiben soros), forgalomirányítással, tabulálással, sortörési konvenció szerint stb. kommunikáljunk vele. Azonban TCP/IP vagy más hálózati porton ülő nyomtatók adatait itt nem tudjuk kifejteni.

A hálózatra kötött nyomtatók használatához lényegében egy olyan külön kifejlesztett kommunikációs programra van szükségünk, amely a szöveg- vagy konverziós szűrőkhöz hasonló módon hívható meg. Erre rögtön adunk is egy példát: a netprint szkript a szabványos bemenetről beolvassa az összes kinyomtatandó adatot és átküldi azokat a hálózatra csatlakoztatott nyomtatónak. A szkript első paramétereiként a nyomtató hálózati nevét adjuk meg, másodiknak pedig portot. Azonban megjegyezzünk, hogy ez csak egyirányú kommunikációt tesz lehetővé (a FreeBSD-től a nyomtatóig). Sok hálózati nyomtató viszont két irányban is képes kommunikálni, ezért érdemes lehet ezt kihasználni (a nyomtató állapotának lekérdezésére, nyilvántartások készítésére stb).

```
#!/usr/bin/perl
#
# netprint - A hálózatra csatlakoztatott nyomtató szövegszűrője
# Helye: /usr/local/libexec/netprint
#
$#ARGV eq 1 || die "Usage: $0 <printer-hostname> <port-number>";

$printer_host = $ARGV[0];
$printer_port = $ARGV[1];

require 'sys/socket.ph';

($ignore, $ignore, $protocol) = getprotobyname('tcp');
($ignore, $ignore, $ignore, $ignore, $address)
    = gethostbyname($printer_host);
```

```
$sockaddr = pack('S n a4 x8', &AF_INET, $printer_port, $address);

socket(PRINTER, &PF_INET, &SOCK_STREAM, $protocol)
|| die "Can't create TCP/IP stream socket: $!";
connect(PRINTER, $sockaddr) || die "Can't contact $printer_host: $!";
while (<STDIN>) { print PRINTER; }
exit 0;
```

Rengeteg szűrőben fel tudjuk használni ezt a szkriptet. Például tegyük fel, hogy egy Diablo 750-N típusú sornyomtatót csatlakoztattunk a hálózatra, amely az 5100-as porton várja a nyomtatandó adatokat. A hálózati neve most scrivener lesz. Íme a hozzá tartozó szövegszűrő:

```
#!/bin/sh
#
# diablo-if-net - Az 5100-as porton figyelő 'scrivener' nevű Diablo
# nyomtató szövegszűrője. Helye: /usr/local/libexec/diablo-if-net
#
exec /usr/libexec/lpr/lpf "$@" | /usr/local/libexec/netprint scrivener 5100
```

9.4.4. A nyomtató használatának szabályozása

Ebben a szakaszban a nyomtató használatának korláztásáról írunk. Az **LPD** rendszeren keresztül meghatározhatjuk, hogy ki képes helyben vagy távolról hozzáférni a nyomtatóhoz, mennyi másolatot nyomtathat, mennyi és egyenként mekkora nyomtatási feladatokat küldhet.

9.4.4.1. A másolatok számának szabályozása

Az **LPD** segítségével a felhasználók egy állományt könnyen ki tudnak nyomtatni akár többször is. Ha (például) a felhasználó egy nyomtatási feladat kiküldéséhez az `lpr -#5` parancsot használja, akkor a nyomtatási feladatban levő összes állományból öt példányt kap. Ennek létjogosultságát azonban nekünk kell megítélni.

Amennyiben úgy érezzük, hogy a további példányok készítése csupán felesleges papír- és tintapazarlás, akkor az `sc` tulajdonság megadásával az `/etc/printcap` állományban kikapcsolhatjuk az `lpr(1) -#` lehetőség használatát. Így amikor a felhasználók a `-#` kapcsolóval küldenek el feladatokat a nyomtatóra, a következőt fogják tapasztalni:

```
lpr: multiple copies are not allowed
```

Fordítása:

```
lpr: másolatok nyomtatása nem engedélyezett
```

Vigyázzunk arra, hogy ha távoli számítógépen zajlik a nyomtatás (lásd

Távoli számítógépekre csatlakoztatott nyomtatók), akkor az `sc` tulajdonságot a távoli számítógép `/etc/printcap` állományában is be kell állítani, máskülönben a felhasználók egy másik számítógépről mindig képesek lesznek több példány nyomtatására.

Nézzünk erre egy példát. Itt most a `rose` nevű számítógép `/etc/printcap` állományát vesszük szemügyre. Ebben a rattan egy nagyon szívélyes nyomtató lesz, ezért engedélyezi a másolatok nyomtatását, azonban a `bamboo` nevű lézernyomtatónál már sokkal válogatósabb lesz, ezért a beállításai közt az `sc` tulajdonsággal kikapcsoljuk a másodpéldányok nyomtatását:

```
#
# /etc/printcap (rose) - A másolatok korlátozása a "bamboo"
# nevű nyomtatón
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:\
    :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

Az `sc` tulajdonságot az `orchid` `/etc/printcap` állományában is meg kell adni (és ha már itt vagyunk, akkor tegyük meg ugyanezt a `teak` esetében is):

```
#
# /etc/printcap (orchid) - Nincsenek másodpéldányok sem a helyi
# "teak" nyomtatón, sem pedig a távoli "bamboo" nyomtatón
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:sc:\
    :if=/usr/local/libexec/ifhp:\
    :vf=/usr/local/libexec/vfhp:\
    :of=/usr/local/libexec/ofhp:

rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:sc:
```

Az `sc` tulajdonság használatával ugyan megakadályozzuk az `lpr -#` parancs teljesítését, azonban ez még mindig nem óv meg minket attól, hogy a felhasználók képesek legyenek többször egymás után lefuttatni az `lpr(1)` parancsot, vagy éppen egyetlen nyomtatási feladatban több állományt is elküldeni:

```
% lpr forsale.sign forsale.sign forsale.sign forsale.sign forsale.sign
```

Számos módszer kínálkozik az effajta visszaélések kivédésére (beleértve a figyelmen kívül hagyást is), lehet velük kísérletezni!

9.4.4.2. A nyomtatók hozzáféréseinek szabályozása

A UNIX csoportkezelésével és az `/etc/printcap` állományban található `rg` tulajdonság felhasználásával korlátozni tudjuk, ki milyen nyomtatón dolgozhat. Ehhez mindössze annyit kell tennünk, hogy besoroljuk egy csoportba azokat a felhasználókat, amelyek hozzáférhetnek a nyomtatóhoz, és az `rg` tulajdonsággal megnevezzük azt.

A csoporton kívüli felhasználókat (köztük magát a `root` felhasználót is) pedig ezután így üdvözlí a rendszer, ha megpróbálnak valamit kinyomtatni egy korlátozott felhasználású nyomtatón:

```
lpr: Not a member of the restricted group
```

Az üzenet fordítása:

```
lpr: Nem jogosult felhasználó
```

Ha erre a távoli számítógépek esetén szükségünk lenne (lásd Távoli számítógépekre csatlakoztatott nyomtatók), akkor tegyük ugyanazt, mint amit az `sc` (a másodpéldányok letiltása, “suppress multiple copies”) tulajdonság esetén is, vagyis az `rg` tulajdonságot adjuk meg azokon a távoli számítógépeken is, amelyek hozzá tudnak férni a megosztott nyomtatóhoz.

Például megengedjük, hogy a `rattan` nevű nyomtatót bárki használhassa, azonban a `bamboo` nyomtatót csak az `artists` nevű csoport használhatja. Következzen hát akkor a `rose` korábbiól már ismert `/etc/printcap` állománya:

```
#
# /etc/printcap (rose) - A bamboo hozzáféréseinek korlátozása
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:\
    :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

Most ne bántsuk a másik (az `orchid` nevű gépen levő) `/etc/printcap` állományt. Így persze az `orchid` bármelyik felhasználója nyomtathat a `bamboo` nyomtatón. De ez most egy olyan eset, ahol egyébként lekorlátozzuk a `orchid` elérését is, ezért az ott beengedett felhasználók már akár használhatják is a nyomtatót. Vagy sem.

Megjegyzés: Minden nyomtatóhoz csak egy ilyen csoportot adhatunk meg.

9.4.4.3. A beküldött nyomtatási feladatok méretének szabályozása

Ha sok felhasználó szeretne a nyomtatóinkhoz hozzáférni, akkor minden bizonnyal meg akarunk adni egy felső határt a felhasználók által beküldhető nyomtatások méretére vonatkozóan. Mivel a nyomtatási könyvtáraknak otthont adó állományrendszer is egyszer betelhet, ezért mindenképpen érdemes gondoskodni arról, hogy mindenki nyomtatási feladatát el tudjuk rendesen tárolni.

Az **LPD** az `mx` tulajdonsággal lehetőséget ad arra, hogy lekorlátozzuk a nyomtatási feladatokban található egyes állományok méretét. Ennek mértékegysége egy `BUFSIZ` blokk, ami pedig 1024 byte. Ha értékül nullát adunk meg, akkor nincs korlátozás, viszont ha semmit sem rögzítünk, akkor az `mx` tulajdonság alapértéke, vagyis 1000 blokk lesz a határ.

Megjegyzés: Ez az érték a nyomtatási feladatokban levő egyes állományok méretére vonatkozik, nem pedig a nyomtatási feladatok teljes méretére.

Fontos tudni, hogy az **LPD** nem dobja vissza a méreten felüli állományokat. Ehelyett a méret alatti részt szépen berakja a sorba és kinyomtatja, a többi pedig elhagyja. Lehetne rajta vitázni, hogy ez mennyire helyes cselekedet.

Példaképpen definiáljunk a korábban használt `rattan` és `bamboo` nyomtatóinkhoz ilyen korlátokat. Mivel az `artists` csoport tagjai hajlamosak nagy PostScript állományokat küldeni, ezért most lekorlátozzuk ezt öt megabyte-ra. A szöveges nyomtatónk esetén azonban nem lesz semmilyen határ:

```
#
# /etc/printcap (rose)
#

#
# Itt nincs korlát a nyomtatási feladatokra:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
      :sh:mx#0:sd=/var/spool/lpd/rattan:\
      :lp=/dev/lpt0:\
      :if=/usr/local/libexec/if-simple:

#
# Öt megabyte a PostScript:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
      :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
      :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
      :if=/usr/local/libexec/psif:\
      :df=/usr/local/libexec/psdf:
```

Ismét hozzátesszük, hogy ezek a korlátok csak a helyi felhasználókra vonatkoznak. Amennyiben távolról is el lehet érni ezt a nyomtatót, a távoli felhasználókat nem fog semmilyen korlátozás érinteni. Azokon a számítógépeken is meg kell adnunk az `/etc/printcap` állományban az `mx` tulajdonságot. Ehhez a Távoli számítógépekre csatlakoztatott nyomtatók című szakaszban találunk segítséget.

Van még egy speciális módszer, amivel képesek vagyunk szabályozni a távolról érkező kérések méretét. Erről a Távoli számítógépekről érkező kérések szabályozása szakaszban olvashatunk.

9.4.4.4. Távoli számítógépekről érkező kérések szabályozása

Az **LPD** nyomtatási rendszer több módot is szolgáltat a távolról érkező nyomtatási feladatok szabályozására:

Az elérés szabályozása

Az `/etc/hosts.equiv` és `/etc/hosts.lpd` állományok segítségével beállíthatjuk, hogy mely távoli számítógépektől fogadjon el kéréseket az **LPD**. Az **LPD** minden kérés elfogadásakor ellenőrzi, hogy a küldő számítógép címe szerepel-e az említett állományok valamelyikében. Ha nem, akkor az **LPD** visszautasítja a kérést.

A két állomány felépítése egyszerű, mert bennük minden sorban egy-egy hálózati nevet adunk meg. Hozzátennénk azonban, hogy legyünk óvatosak, mivel az `/etc/hosts.equiv` állományt az `ruserok(3)` protokoll is használja, ezért ennek módosítása hatással van az `rsh(1)` és `rcp(1)` programok működésére.

Például most nézzük meg a `rose /etc/hosts.lpd` állományát:

```
orchid
violet
madrigal.fishbaum.de
```

Ennek megfelelően tehát a `rose` elfogadja az `orchid`, `violet` és `madrigal.fishbaum.de` nevű távoli számítógépek kéréseit. Ha bármilyen más gép próbál hozzáférni a `rose` által felkínált **LPD** szolgáltatáshoz, visszautasítja.

A méret szabályozása

Szabályozhatjuk többek közt azt is, hogy mennyi szabad területnek kell fennmaradnia a nyomtatási könyvtárnak otthont adó állományrendszeren. A helyi nyomtató könyvtárában ehhez hozzunk létre egy `minfree` nevű állományt. Ide írjuk be, mennyi szabad lemezblokk (512 byte-os egység a lemezen) szükségeltetik egy távolról beérkező nyomtatási feladat fogadásához.

Így gondoskodhatunk róla, hogy a távoli felhasználók nem fogják eltömíteni az állományrendszerünket, illetve ezzel egyúttal adhatunk némi előnyt a helyi felhasználóknak is: ők ugyanis még azután is képesek lesznek nyomtatási feladatokat küldeni a nyomtatónak, miután az állományrendszeren található szabad terület mennyisége már rég a `minfree` állományban szereplő érték alá csökkent.

Példaként most a `bamboo` nevű nyomtatónkhoz adjunk meg egy ilyen `minfree` állományt. Ehhez az `/etc/printcap` állományból tudjuk kideríteni a hozzá tartozó nyomtatási könyvtárat. Lássuk tehát belőle a `bamboo` bejegyzését:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
:lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:mx#5000:\
:if=/usr/local/libexec/psif:\
:df=/usr/local/libexec/psdf:
```

A nyomtatási könyvtárat az `sd` tulajdonság határozza meg. Úgy állítjuk most be, hogy az **LPD** számára a távoli nyomtatási feladatok fogadásához ebben a könyvtárban legalább három megabyte (6144 blokk) szabad területnek mindig lennie kell:

```
# echo 6144 > /var/spool/lpd/bamboo/minfree
```

A felhasználók szabályozása

Az `/etc/printcap` állományban megadható `rs` tulajdonság segítségével korlátozhatjuk a helyi nyomtatókhoz hozzáférni képes távoli felhasználókat. Amikor az `rs` tulajdonság szerepel egy helyben csatlakozó nyomtató leírásánál, akkor az **LPD** csak abban az esetben fogad el távoli felhasználóktól nyomtatási feladatot, ha az adott feladatot küldő felhasználónak ugyanazon a néven van a helyi gépen is hozzáférése. Máskülönben az **LPD** vissza fogja utasítani a kérést.

Ez a tulajdonság különösen fontos olyan környezetben, ahol (például) több szervezeti egység használ egyetlen közös hálózatot és bizonyos felhasználók képesek átlépni szervezeti egységük határait, mivel ha a hozzáférést adunk nekik a rendszereinkhez, akkor képesek a saját helyükről használni ezeket. Ha ehelyett *csupán* a nyomtatóinkat és a számítógépünk összes erőforrását akarjuk megosztani, akkor létrehozhatunk a számukra olyan “token” hozzáféréseket is, amikhez nem tartozik sem felhasználói könyvtár, sem pedig parancsértelmező (pontosabban a `/usr/bin/false`).

9.4.5. A nyomtató használatának nyilvántartása

Tehát szükségünk lenne a nyomtatások költségének elszámolására. Miért is ne tennénk ilyet? A papír és a tinta bizony pénzbe kerül, amihez még hozzájárulnak más egyéb karbantartási költségek is — a nyomtatók dugig vannak mindenféle mozgó alkatrészszel, amelyek előbb-utóbbi el is romlanak. Tegyük fel, hogy a nyomtatóink kapacitása, kihasználtsága és karbantartási költsége alapján már megállapítottunk egy elszámolási egységet (oldalanként, méterenként, akárminként). De hogyan lássunk hozzá a nyomtatások költségének tényleges nyilvántartásához?

Van egy rossz hírünk: az **LPD** nyomtatási rendszer önmaga nem tud segíteni ebben a feladatban. A nyilvántartás nagyban függ a használt nyomtatóktól, a nyomtatott formátumoktól és nyomtató *általunk* kiszabott költségeitől.

A nyilvántartás létrehozásához át kell írunk a nyomtatóhoz tartozó szűrőt (a nyers szövegek költségének felszámításához) és konverziós szűrőket (a különféle formátumok költségei miatt), amikkel aztán számolhatjuk vagy lekérdezhetjük a kinyomtatott lapokat. Egyetlen kimeneti szűrő használatával szinte semmire se megyünk, mivel az nem képes nyilvántartás vezetésére. Erről bővebb útmutatást a Szűrők szakaszban találhatunk.

Általánosságban véve két módon vezethetünk nyilvántartást:

- Az *időszakos elszámolás* a gyakoribb, mivel ez az egyszerűbb. Amikor valaki végrehajt egy nyomtatási feladatot, a szűrő a nyilvántartást tároló állományba feljegyzi a felhasználó azonosítóját, a gépének nevét és a kinyomtatott oldalakat. Ezután minden hónapban, félévben, évben vagy akár tetszőleges időközönként összegyűjtjük a nyomtatók nyilvántartásait és külön feljegyezzük az egyes felhasználók nyomtatásait, majd benyújtjuk róla a számlát. Töröljük az összes naplóállományt, és tiszta lappal kezdjük a következő időszakot.
- Az *azonnali elszámolás* már nem annyira népszerű, mivel nehezebb megvalósítani. Ekkor a felhasználók már közvetlenül a nyomtatás után megkapják a számlát, hasonlóan a lemezkvótákhoz. Meg tudjuk akadályozni ezzel azt is, hogy a felhasználók túlléphessék az előre kiszabott “nyomtatási kvótájukat”, amit persze menet közben lehet ellenőrizni és állíthatni. A felhasználók és kvótájuk nyomonkövetéséhez viszont szükségünk lesz egy kis adatbáziskezelésre is.

Az **LPD** nyomtatási rendszer mind a két módszer kivitelezéséhez tud segítséget nyújtani, hiszen amikor szűrőket állítunk be (vagyis szinte mindig), lehetőségünk van a nyilvántartást végző programrészeket is beilleszteni. És ami feltétlenül előnyös: óriási mértékű rugalmasságot ajánl fel a nyilvántartás megvalósításához. Például magunk választhatjuk ki, hogy időszakos vagy azonnali elszámolást alkalmazunk. Meg tudjuk adni, milyen információkat rögzítsünk: felhasználói neveket, számítógépek neveit, a nyomtatási feladatok típusát, vagy a kinyomtatott oldalakat, a használt lapok területét, a nyomtatások időbeli igényeit és így tovább. Ehhez mindössze csak a szűrőket kell módosítani.

9.4.5.1. Nyilvántartás gyorsan és egyszerűen

A FreeBSD-ben egyből találunk is két programot, amivel pillanatok alatt ki tudunk alakítani egy egyszerű időszakos elszámolási rendszert. Ezek Az `lpf` szövegszűrő című szakaszban ismertetett `lpf` és a nyomtatók nyilvántartásait tartalmazó állományok adatainak összegyűjtését és kiértékelését végző `pac(8)`.

Ahogy korábban már leírtuk a szűrőkről szóló szakaszban (Szűrők), az **LPD** a szöveg- és konverziós szűrőket parancssorból a nyilvántartást tároló állomány nevével indítja el. Ezt a paramétert a szűrők aztán fel tudják használni a nyilvántartások feljegyzéséhez. Az állomány nevét az `/etc/printcap` állományban szereplő `af` tulajdonsággal tudjuk megadni, vagy teljes elérési úttal, vagy pedig a nyomtatási könyvtárhoz viszonyítva.

Az **LPD** az `lpf` szűrőt a lap szélességének és hosszának megadásával indítja el (ezeket az értékeket a `pw` és `p1` tulajdonságokból származtatja). Az `lpf` ezek felhasználásával meg tudja mondani, mennyi papírt használtunk el. Miután kiküldte az állományt a nyomtatóra, nyilvántartásba is veszi. Ezek a típusú bejegyzések valahogy így néznek ki:

```

2.00 rose:andy
3.00 rose:kelly
3.00 orchid:mary
5.00 orchid:mary
2.00 orchid:zhang

```

Minden nyomtatóhoz érdemes külön nyilvántartást vezetni, mivel az `lpf` nem tartalmaz semmilyen beépített zárolási megoldást, ezért két `lpf` párhuzamos futtatása könnyen összezagyválhatja a közösen használt nyilvántartások tartalmát. Az `/etc/printcap` állományban az `af=acct` tulajdonság megadásával könnyen létre tudunk hozni minden nyomtatóhoz külön nyilvántartást. Ilyenkor minden nyomtató könyvtárában megjelenik egy `acct` nevű állomány.

Amikor elérkezünk a nyomtatások kiszámlázásához, futtassuk le a `pac(8)` programot. Ehhez mindössze annyit kell tennünk, hogy átlépünk az elszámlolni kívánt nyomtató könyvtárba és begépeljük a `pac` parancsot. Ekkor kapunk egy ehhez hasonló, dollár alapú kimutatást:

Login	pages/feet	runs	price
orchid:kelly	5.00	1	\$ 0.10
orchid:mary	31.00	3	\$ 0.62
orchid:zhang	9.00	1	\$ 0.18
rose:andy	2.00	1	\$ 0.04
rose:kelly	177.00	104	\$ 3.54
rose:mary	87.00	32	\$ 1.74
rose:root	26.00	12	\$ 0.52
total	337.00	154	\$ 6.74

A `pac(8)` a következő paramétereket várja:

`-Pnyomtató`

Az kiértékelendő *nyomtató* neve. Ez a paraméter csak akkor használható, ha az `/etc/printcap` állományban az `af` tulajdonságnak teljes elérési utat adtunk meg.

`-c`

A felhasználók nevei helyett a fizetendő összeg szerint rendezze a listát.

`-m`

Hagyja figyelmen kívül a nyilvántartásban szereplő gépek hálózati neveit. Ennek hatására az `alpha` gépről nyomtató `smith` meg fog egyezni a `gamma` gépről nyomtatóval. A beállítás nélkül ez a két felhasználó el fog térni.

`-pár`

A paraméterként megadott *ár* dollár értékkel számol oldalanként vagy lábanként az `/etc/printcap` állományban megadott `pc` tulajdonság értéke helyett (ami alaphoz két cent). Az *ár* lebegőpontos (valós) számként is megadható.

`-r`

A rendezési sorrend megfordítása.

-s

Hozzon létre egy elszámolást, majd törölje a hozzá kapcsolódó nyilvántartási adatokat.

nev . . .

Csak az adott *nev* felhasználók adatait értékelje ki.

A `pac(8)` által alapértelmezés szerint generált kimutatásban láthatjuk az egyes gépekről származó egyes felhasználók kinyomtatott oldalait. Ha nekünk viszont nem számít, hogy honnan küldték a kéréseket (mivel bárhonnan lehet küldeni), akkor a `pac -m` paranccsal az alábbi táblázatot készíttethetjük el:

Login	pages/feet	runs	price
andy	2.00	1	\$ 0.04
kelly	182.00	105	\$ 3.64
mary	118.00	35	\$ 2.36
root	26.00	12	\$ 0.52
zhang	9.00	1	\$ 0.18
total	337.00	154	\$ 6.74

Itt megtaláljuk a ténylegesen kifizetendő összegeket is, amik kiszámításához a `pac(8)` az `/etc/printcap` állomány `pc` tulajdonságát használja (ez alapból 200, avagy 2 cent oldalanként). Ezzel a tulajdonsággal tehát egy cent századrészeben mérve tudjuk megadni az oldalankénti vagy lábankénti árakat. Ezt a beállítást természetesen a `pac(8)` `-p` opciójával felül tudjuk bíráltni. Arra azonban vigyázzunk, hogy a `-p` után dollárban kell megadnunk az árat. Emiatt tehát a

```
# pac -p1.50
```

parancs szerint minden egyes oldal másfél dollárba fog kerülni. Ezzel az opcióval aztán alaposan megdönthetjük az árakat.

Végezetül megemlítjük, hogy a `pac -s` parancs az általa létrehozott elszámolást egy külön állományba menti, amelynek a neve nagyjából megegyezik a nyilvántartást végzőével, de `_sum-ra` (mint "summary", azaz elszámolás) végződik. Ezután nullázza a nyilvántartást. Amikor a `pac(8)` programot újra lefuttatjuk, újból beolvassa a korábban elmentett elszámolásokat, majd hozzászámolja a többi a hagyományos nyilvántartási adatokból.

9.4.5.2. Hogyan tudjuk számolni a kinyomtatott lapokat?

A nyilvántartás pontos vezetéséhez még távolról is valamilyen módon meg kell tudnunk mondani, hogy mennyi lapot használt egy nyomtatási feladat végrehajtása. Ez a nyomtatás nyilvántartásának egyik alapvető problémája.

A nyers szövegek esetében ez nem is annyira bonyolult: egyszerűen számoljuk össze, hogy a nyomtatási feladatban mennyi sor kinyomtatására lesz szükség és vessük össze ezt a nyomtató által lapoként kinyomtatott sorok számával. Ne felejtjük el számításba venni a szövegben felbukkanó törlések hatását, vagy az olyan hosszú sorokat, amelyek a valóságban több sorban fognak megjelenni.

Viszont (Az `lpf` szövegszűrő című szakaszban bemutatott) `lpf` program ezeket mind lekezeli a nyilvántartások készítése során. Ezért ha szintén egy nyilvántartást vezetni képes szövegszűrőt akarunk írni, akkor mindenképpen érdemes megnéznünk az `lpf` forráskódját.

De hogyan bányunk el a többi formátummal?

Nos, a DVI-Laserjet és DVI-PostScript közti átalakítások esetén a kinyomtatott lapok számának megállapításához meg kell tanítanunk a szűrőnket értelmezni a `dvi1j` vagy `dvijs` parancsok kimenetét. Ugyanezt meg tudjuk tenni más formátumok és más konverziós programok használata során is.

Azonban ezek a módszerek nem veszik számításba, hogy a nyomtató egyáltalán kinyomtatta-e az összes elküldött oldalt. Sok minden történhet még addig, például beragadhat a papír, kifogyhat a tinta vagy akár felrobbanhat a nyomtató — a felhasználónak ettől függetlenül még fizetnie kell.

Mit lehet ilyenkor tenni?

A *precíz* nyilvántartásnak csak egyetlen *biztos* módja létezik. Olyan nyomtatót szerezzünk be, amely képes megmondani, mennyi lapot használt el a nyomtatás során, majd egy ilyen csatlakoztassunk soros porton vagy hálózaton keresztül. Szinte majdnem az összes PostScript nyomtató támogatja ezt a lehetőséget, ahogy sok más gyártmány és típus is (például a hálózati Imagen lézernyomtatók). A nyomtatóhoz tartozó szűrőt ehhez úgy kell módosítani, hogy lekérdezzük a kinyomtatott lapok számát a nyomtatás után és *kizárólag* erre az értékre alapozva készítünk nyilvántartást. Itt nincs szükség sem a sorok számolására, sem pedig az állományok (könnyen elhibázható) átvizsgálására.

Természetesen lehetünk nagylelkűek és ne számítsunk fel semmit a nyomtatásért.

9.5. A nyomtatók használata

Ebből a szakaszból megtudhatjuk, hogyan használjuk a FreeBSD-n beállított nyomtatónkat. Röviden most itt foglaljuk össze az ide tartozó felhasználói parancsokat:

`lpr(1)`

Nyomtatási feladatokat hajt végre.

`lpq(1)`

Ellenőrzi a nyomtatási sorokat.

`lprm(1)`

Feladatokat vesz ki a nyomtatási sorokból.

Ezek mellett létezik még a nyomtatók és a hozzájuk tartozó sorok irányítására alkalmas parancs is, az `lpc(8)`, amelyre a *A nyomtatók vezérlése* című szakaszban fogunk részleteiben kitérni.

A nyomtatók/sorok `/etc/printcap` állományban szereplő nevük szerinti megadásához az `lpr(1)`, `lprm(1)` és `lpq(1)` parancsok közül mindegyik elfogadja a `-P nyomtatónév` paramétert. Ennek köszönhetően képesek vagyunk nyomtatási feladatokat küldeni, eltávolítani vagy felügyelni az egyes nyomtatók soraiban. Ha nem használjuk a `-P` kapcsolót, akkor az érintett nyomtató a `PRINTER` környezeti változó által meghatározott lesz. Végül, ha a `PRINTER` nevű környezeti változót sem állítottuk be, akkor a parancsok alapértelmezett módon az `lp` nevű nyomtatót fogják használni.

A továbbiakban az *alapértelmezett nyomtató* kifejezés a `PRINTER` környezeti változó által megnevezett nyomtatóra fog utalni, illetve ha ezt nem definiáltuk, akkor az `lp` nevű nyomtatóra.

9.5.1. Nyomtatási feladatok végrehajtása

Az állományok kinyomtatásához írjuk be:

```
% lpr állománynév ...
```

Ezzel kinyomtatjuk az összes felsorolt állományt az alapértelmezett nyomtatón. Ha nem adunk meg állományokat, akkor az `lpr(1)` parancs a szabványos bemenetről várja a nyomtatandó adatokat. Például ezzel a paranccsal néhány igen fontos rendszerállományt tudunk kinyomtatni:

```
% lpr /etc/host.conf /etc/hosts.equiv
```

A nyomtató megválasztásához így adjuk ki a parancsot:

```
% lpr -P nyomtatónév állománynév ...
```

Ez a példa kinyomtatja az aktuális könyvtár részletes listáját a `rattan` nevű nyomtatón:

```
% ls -l | lpr -P rattan
```

Mivel egyetlen állományt sem adtunk meg az `lpr(1)` programnak, az `lpr` parancs a nyomtatandó adatokat a szabványos bemenetről várja, ami jelen esetünkben a `ls -l` parancs kimenete.

Az `lpr(1)` ezeken felül még képes értelmezni rengeteg formázásra, konverzióra, másolatok készítésére stb. utasító kapcsolót is. Erről bővebben a Nyomtatási beállítások című szakaszban lesz szó.

9.5.2. Nyomtatási feladatok felügyelete

Amikor az `lpr(1)` programmal nyomtatunk, az összes nyomtatandónk egy “nyomtatási feladatnak” nevezett csomagba kerül, ami pedig az **LPD** nyomtatási rendszerébe. Minden nyomtatóhoz tartozik egy nyomtatási sor, ahol részünkről és mások által eddig kiadott nyomtatási feladatokat találhatjuk. A nyomtató ezután ezeket érkezési sorrend szerint dolgozza fel.

Az alapértelmezett nyomtatóhoz tartozó sor állapotát az `lpq(1)` programmal tudjuk megnézni. Ha egy adott nyomtatóra vagyunk kíváncsiak, akkor használjuk a `-P` kapcsolót. Például a

```
% lpq -P bamboo
```

parancs a `bamboo` nevű nyomtató sorát fogja megmutatni. Példaképpen lássuk is ilyen esetben az `lpq` parancs eredményét:

```
bamboo is ready and printing
Rank  Owner    Job  Files                                Total Size
active kelly    9    /etc/host.conf, /etc/hosts.equiv    88 bytes
2nd    kelly    10    (standard input)                    1635 bytes
3rd    mary     11    ...                                  78519 bytes
```

Itt három nyomtatási feladatot láthatunk a `bamboo` nyomtatási sorában. Az első nyomtatási feladat, amit a `kelly` nevű felhasználó küldött, a 9-es “feladatszámot” kapta. A nyomtatóhoz tartozó összes feladat kap egy ilyen egyedi számot. Többnyire nyugodtan figyelmen kívül hagyhatjuk, azonban szükségünk lehet rá, ha éppen törölni kívánjuk a hozzá tartozó nyomtatási feladatot. Ezzel majd a Nyomtatási feladatok eltávolítása című szakaszban foglalkozunk.

A kilences számú nyomtatási feladat két állományt tartalmaz: ha a parancssorban több állományt adunk meg az `lpr(1)` programnak, akkor az egy nyomtatási feladatnak számít. Ez egyben a pillanatnyilag aktív nyomtatási feladat (ezt a "Rank" oszlopban szereplő `active` érték jelzi), tehát a nyomtató éppen ezzel foglalatoskodik. A második nyomtatási feladat közvetlenül az `lpr(1)` szabványos bemenetére érkezett. A harmadik a `mary` nevű felhasználótól jött, és ez egy nagyobb méretű nyomtatási feladat. A nyomtatandó állomány elérési útvonala túlságosan hosszú ahhoz, hogy ki lehessen írni, ezért az `lpr(1)` csak három pontot jelez ki helyette.

Az `lpq(1)` kimenetének első sorai is nagyon hasznos információt tartalmaz: megtudhatjuk, mit csinál éppen (legalább is az **LPD** szerint) a nyomtató.

A `-l` kapcsolóval az `lpq(1)` parancstól kérhetünk sokkal részletesebb listázást is. Például így nézhet ki a `lpq -l` parancs eredménye:

```
waiting for bamboo to become ready (offline ?)
kelly: 1st                                [job 009rose]
      /etc/host.conf                      73 bytes
      /etc/hosts.equiv                   15 bytes

kelly: 2nd                                [job 010rose]
      (standard input)                   1635 bytes

mary: 3rd                                 [job 011rose]
      /home/orchid/mary/research/venus/alpha-regio/mapping 78519 bytes
```

9.5.3. Nyomtatási feladatok eltávolítása

Ha meggondoltuk volna magunkat egy nyomtatási feladattal kapcsolatban, az `lprm(1)` paranccsal még törölni tudjuk a sorból. Az `lprm(1)` gyakran még a folyamatban levő nyomtatási feladatot is képes eltávolítani, azonban előfordulhat, hogy a nyomtatási feladat egy része már elvégzésre került.

Az alapértelmezett nyomtató sorából csak úgy tudunk nyomtatási feladatokat törölni, ha először az `lpq(1)` segítségével megkeressük a számukat. Ha ez megvan, írjuk be:

```
% lprm feladatám
```

Adott nyomtatóról a `-P` kapcsoló segítségével tudunk nyomtatási feladatot törölni. A most következő parancs a `bamboo` nevű nyomtatóról törli a 10-es számú nyomtatási feladatot:

```
% lprm -P bamboo 10
```

Az `lprm(1)` parancs esetén még használhatóak az alábbi rövidítések is:

`lprm -`

Eltávolítja a hozzánk tartozó az összes nyomtatási feladatot (az alapértelmezett nyomtatón).

`lprm felhasználó`

Eltávolítja az adott *felhasználó*hoz tartozó összes nyomtatási feladatot (az alapértelmezett nyomtatón).

Kizárólag a rendszergazdák képesek erre, a rendes felhasználók csak a saját nyomtatási feladataikat törölhetik.

lprm

A nyomtatási feladat száma, a felhasználói név vagy a `-P` megadása nélkül az `lprm(1)` törli az alapértelmezett nyomtatón éppen aktív nyomtatási feladatot, amennyiben az a miénk. Csak a rendszergazdák képesek bármilyen aktív nyomtatási feladatot törölni.

Ha kiegészítjük az imént említett rövidítéseket a `-P` paraméter megadásával, akkor az alapértelmezett nyomtató helyett bármelyik másikat is használhatjuk. Például ez a parancs eltávolítja az aktuális felhasználó összes nyomtatási feladatot a `rattan` nevű nyomtatón:

```
% lprm -P rattan -
```

Megjegyzés: Hálózati környezetben az `lprm(1)` csak arról a gépről engedi törölni a nyomtatási feladatokat, amelyről küldték ezeket, még abban az esetben is, amikor ugyanaz a nyomtató más számítógépekről is elérhető. A következő parancssorozat ezt igyekszik szemléltetni:

```
% lpr -P rattan myfile
% rlogin orchid
% lpq -P rattan
Rank   Owner      Job  Files                Total Size
active seeyan   12   ...                49123 bytes
2nd    kelly      13   myfile              12 bytes
% lprm -P rattan 13
rose: Permission denied
% logout
% lprm -P rattan 13
dfA013rose dequeued
cfA013rose dequeued
```

9.5.4. Túl a nyers szövegen: nyomtatási beállítások

Az `lpr(1)` parancs számos olyan beállítást enged, amelyekkel a szövegek formázását, grafikák átalakítását illetve más állományformátumok használatát, másolatok készítését, nyomtatási feladatok irányítását és még sok minden mást el tudunk végezni. Ebben a szakaszban pontosan ezekről a kapcsolókról lesz szó.

9.5.4.1. Formázási és konverziós beállítások

Az `lpr(1)` most következő opciói a nyomtatási feladatokban található állományok formázását vezérlik. Akkor használjuk ezeket a beállításokat, ha a nyomtatási feladat nem tartalmaz nyers szöveget, vagy ha nyers szöveget akarunk formázni a `pr(1)` segédprogrammal.

Például az alábbi parancs kinyomtat egy `halászati-jelentés.dvi` nevű (a \TeX betűszedű rendszerből már jól ismert) DVI állományt a `bamboo` nevű nyomtatón:

```
% lpr -P bamboo -d halászati-jelentés.dvi
```

Ezek a beállítások a nyomtatási feladatban szereplő minden egyes állományra vonatkoznak, ezért nem keverhetjük (például) a DVI és ditroff formátumú állományokat egy nyomtatási feladaton belül. Ehelyett külön nyomtatási

feladatokban kell elküldenünk az eltérő formátumú állományokat, és mindegyik nyomtatási feladathoz külön konverziós beállításokat kell megadnunk.

Megjegyzés: A `-p` és `-T` kapcsolók kivételével az itt felsorolt összes beállításnak a kiválasztott nyomtatóhoz szüksége van a megfelelő konverziós szűrőre. Például a `-d` opció használatához kell egy konverziós szűrő a DVI formátumhoz. A Konverziós szűrők című szakasz erről ad bővebb tájékoztatást.

`-c`

Cifplot állományok nyomtatása.

`-d`

DVI állományok nyomtatása.

`-f`

FORTRAN forrás nyomtatása.

`-g`

Plot formátumú adatok nyomtatása.

`-i` *szám*

A kinyomtatott szöveg behúzásának növelése a *szám* értékével. Ha nem adjuk meg a *számot*, akkor ennek értéke 8 lesz. Ez a beállítás csak bizonyos konverziós szűrőkkel működik.

Megjegyzés: Ne hagyjunk helyet az `-i` és a *szám* között.

`-l`

A szöveg formázás nélküli nyomtatása, vezérlőkarakterekkel együtt.

`-n`

Ditroff (eszközfüggetlen troff) adat nyomtatása.

`-p`

Nyomtatás előtt a szöveg formázása a `pr(1)` programmal. Lásd `pr(1)`.

`-T` *cím*

Az állomány neve helyett a fejlécben a *cím*et jeleníti meg a `pr(1)`. Ennek a beállításnak csak a `-p` opcióval együtt van hatása.

`-t`

Troff adat nyomtatása.

-v

Raszteres adatok nyomtatása.

Vegyünk az iméntiekre egy példát. A következő parancs az `ls(1)` szépen megformázott `man` oldalát nyomtatja ki az alapértelmezett nyomtatón:

```
% zcat /usr/share/man/man1/ls.1.gz | troff -t -man | lpr -t
```

A `zcat(1)` kitömöríti az `ls(1)` `man` oldalának forrását és átadja a `troff(1)` parancsnak, ami ebből létrehoz a GNU `troff` formátumának megfelelő kimenetet és továbbadja az `lpr(1)` parancsnak, ami végül elküldi a nyomtatási feladatot az **LPD** nyomtatási rendszernek. Mivel az `lpr(1)` parancsnak megadtuk az `-t` kapcsolót, a nyomtatási rendszer a GNU `troff` formátumban érkező adatokat magától át fogja alakítani olyan formátumra, amit a nyomtató is képes lesz megérteni.

9.5.4.2. Nyomtatási feladatok kezelése

Az `lpr(1)` most felsorolandó beállításai az **LPD** rendszert arra tudjuk utasítani, hogy a nyomtatási feladatot különleges módon kezelje:

-# *példányszám*

Egyetlen példány helyett hozzon létre *példányszám* számú példányt a nyomtatási feladatban található összes állományból. A rendszergazda a nyomtató kímélése érdekében ezt a lehetőséget letilthatja, amivel inkább a fénymásoló használatára ösztönzi a felhasználókat. Lásd A másolatok számának szabályozása szakasz.

A beállítás illusztrálásaként most az alapértelmezett nyomtatón először nyomtassunk ki három példányt a *parser.c*, majd ezután a *parser.h* állományokból:

```
% lpr -#3 parser.c parser.h
```

-m

A rendszer küldjön levelet a nyomtatási feladat teljesítése után. Ekkor az **LPD** a nyomtatási feladat elvégzése után levelet küld a helyi postafiókunkba. A levélben kifejti, hogy sikeres volt-e a nyomtatás, vagy esetleg valamilyen hiba keletkezett, és ha hiba történt, akkor pontosan mi is volt az.

-s

Ne másolja közvetlenül az állományokat a nyomtatási könyvtárba, hanem készítsen hozzájuk szimbolikus linkeket.

Egy nagyobb nyomtatási feladat elvégzése esetén javasolt használni ezt a kapcsolót. Ezzel a megoldással helyet tudunk spórolni a nyomtatási könyvtárban (amikor a nyomtatási feladatok könnyen megtelítheti a nyomtatási könyvtárat tároló állományrendszert). Emellett időt is takarítunk meg, mivel az **LPD**-nek nem kell a nyomtatási feladat minden egyes bitjét átmásolni a nyomtatási könyvtárba.

Van azonban egy hátránya: mivel az **LPD** ekkor közvetlenül az eredeti állományra fog hivatkozni, ezért a nyomtatás befejezéséig azt nem módosíthatjuk vagy törölhetjük.

Megjegyzés: Ha egy távoli nyomtatónak küldjük a nyomtatási feladatot, akkor az **LPD** a helyi és a távoli számítógép között mégis kénytelen lesz átmásolni a nyomtatási feladatot, így a `-s` kapcsoló egyedül csak a helyi nyomtatási könyvtárban fog helyet spórolni. Ettől eltekintve még ilyenkor is hasznunkra válhat.

-r

Törölje a nyomtatási feladatban szereplő állományokat, miután átmásolta ezeket a nyomtatási könyvtárba, vagy miután a -s kapcsoló használatával kinyomtatta ezeket. Nagy körültekintéssel használjuk!

9.5.4.3. A fejléclapok beállításai

Az lpr(1) most következő beállításai a nyomtatási feladatok fejlécében megjelenő szövegekre vannak hatással. Így ha letiltottuk a fejléclapok használatát, akkor ezek a kapcsolók lényegében semmit sem állítanak. A Fejléclapok című szakaszból tudhatunk meg többet ezek beállításáról.

-C *szöveg*

A fejléclapon megjelenő hálózati név helyett a *szöveg* fog szerepelni. A hálózati név általában annak a gépnek a neve, ahonnan a nyomtatási feladatot küldték.

-J *szöveg*

A fejléclapon megjelenő nyomtatási feladat neve helyett a *szöveg* fog megjelenni. A nyomtatási feladat neve általában a benne szereplő első állomány nevével egyezik meg, ha a szabványos bemenetről nyomtatunk, akkor egyszerűen csak *stdin*.

-h

Ne nyomtasson fejléclapot.

Megjegyzés: Bizonyos helyeken előfordulhat, hogy ennek a kapcsolónak nincs semmilyen hatása a fejléclapok létrehozásának módszeréből fakadóan. A részleteket lásd a Fejléclapok szakaszban.

9.5.5. A nyomtatók vezérlése

A nyomtatóink rendszergazdjaként nekünk kell telepítenünk, üzembe helyezni és kipróbálnunk ezeket. Az lpc(8) parancs használatával még jobban képesek vagyunk kapcsolatba lépni velük. Az lpc(8) paranccsal:

- el tudjuk indítani és le tudjuk állítani a nyomtatókat;
- be- és ki tudjuk kapcsolni a nyomtatási soraikat;
- át tudjuk rendezni az egyes sorokban található nyomtatási feladatokat.

Először is essen pár szó a fogalmakról: ha a nyomtató *leállt*, akkor semmit sem fog kinyomtatni a sorából. A felhasználók továbbra is képesek nyomtatási feladatokat küldeni, amik azonban egészen addig fognak várakozni, amíg a nyomtatót *el nem indítjuk* vagy a sorát ki nem ürítjük.

Ha egy sort *kikapcsolunk*, akkor (a `root` kivételével) egyetlen felhasználó sem képes nyomtatási feladatokat küldeni a nyomtatónak. A *bekapcsolt* sorok képesek csak nyomtatási feladatot fogadni. A nyomtató *elindítható* kikapcsolt sorral is, ilyenkor egészen addig folytatja a nyomtatási feladatok elvégzését, amíg a sor ki nem ürül.

Általánosan elmondható, hogy az `lpc(8)` parancs használatához a `root` felhasználó jogosultságaira van szükségünk. Az `lpc(8)` parancsot minden más esetben csak a nyomtató állapotának ellenőrzésére vagy a megakadt nyomtató újraindítására használhatjuk.

Foglaljuk röviden össze az `lpc(8)` parancsait. A legtöbb parancs kiadásához még szükséges egy *nyomtatónév* paraméter megadása is, amivel megnevezzük az utasítani kívánt nyomtatót. Helyette használható az `all` szó is, amivel az `/etc/printcap` állományban szereplő összes nyomtatót egyszerre utasíthatjuk.

`abort nyomtatónév`

Az aktuális nyomtatási feladat megszakítása és a nyomtató leállítása. Ha a nyomtatási sort még nem kapcsoltuk ki, a felhasználók küldhetnek további nyomtatási feladatokat.

`clean nyomtatónév`

A nyomtató könyvtárából töröljük a régi állományokat. Esetenként adódhat, hogy bizonyos nyomtatási feladatok állományait nem takarította el az **LPD**, különösen abban az esetben, amikor a nyomtatás vagy az adminisztrálás során keletkezett valamilyen hiba. Ez a parancs segít megtalálni a nyomtatási könyvtárból már kikopott állományokat és törli ezeket.

`disable nyomtatónév`

Az újonnan érkező nyomtatási feladatok besorolásának kikapcsolása. Ha a nyomtató még működik, akkor folytatni fogja a sorban még bennmaradt nyomtatási feladatok elvégzését. A rendszergazda (a `root`) még a kikapcsolt sorok esetén is küldhet nyomtatási feladatokat.

Ez a parancs valójában akkor hasznos, ha egy új nyomtató vagy egy új szűrő működését próbálgatjuk: ilyenkor érdemes kikapcsolni a nyomtatási sort és `root` felhasználóként nyomtatási feladatokat küldeni. A többi felhasználó a tesztelés befejezéséig nem tud majd nyomtatási feladatokat küldeni, vagyis egészen addig, amíg a nyomtatási sort vissza nem kapcsoljuk az `enable` paranccsal.

`down nyomtatónév üzenet`

A nyomtató üzenem kívül helyezése. Lényegében megegyezik egy `disable` és utána egy `stop` parancs kiadásával. Az *üzenet* akkor jelenik meg, amikor a valaki megpróbálja lekérdezni a nyomtató állapotát az `lpc status` paranccsal, vagy amikor megnézi a nyomtatási sorát az `lpq(1)` paranccsal.

`enable nyomtatónév`

A nyomtatóhoz tartozó nyomtatási sor bekapcsolása. A felhasználók ezután már képesek lesznek a nyomtatónak feladatokat küldeni, azonban egészen addig nem nyomtatódik ki semmi, amíg a nyomtatót el nem indítjuk.

`help parancsnév`

Megmutatja a *parancsnév* parancshoz tartozó súgót. A *parancsnév* megadása nélkül a rendelkezésre álló parancsok listáját kapjuk meg.

`restart nyomtatónév`

Elindítja a nyomtatót. A felhasználók ezt a parancsot tudják használni abban az esetben, amikor valamilyen megmagyarázhatatlan okból az **LPD** működése megáll, viszont ezzel nem tudják elindítani a `stop` vagy `down`

parancsokkal leállított nyomtatót. A `restart` parancs megegyezik az `abort` és a `start` egymás utáni kiadásával.

`start nyomtatónév`

Elindítja a nyomtatót, és a nyomtató nekilát kinyomtatni a sorában levő nyomtatási feladatokat.

`stop nyomtatónév`

Leállítja a nyomtatót, és a nyomtató az aktuális nyomtatási feladat befejezése után már nem kezd neki újabbnak. Ettől függetlenül a felhasználók még továbbra is képesek feladatokat küldeni a nyomtatási sorába.

`topq nyomtatónév feladat-vagy-felhasználónév`

Átrendezi a *nyomtatónév* nevű nyomtató sorát úgy, hogy a megadott azonosítójú *feladatot* vagy a megadott *felhasználónévhez* tartozó nyomtatási feladatokat a sor elejére teszi. Ennél a parancsnál *nyomtatónévnek* nem adhatjuk meg az `all` értéket.

`up nyomtatónév`

Üzembe helyezi a nyomtatót, tulajdonképpen a `down` parancs ellentéte. Megegyezik egy egymás után kiadott `start` és `enable` paranccsal.

Az `lpc(8)` a fenti parancsokat a parancssorból fogadja el. Ha itt nem adunk meg neki semmilyen parancsot, akkor az `lpc(8)` interaktív módba vált, ahol ugyanezeket a parancsokat adhatjuk ki, egészen az `exit`, `quit` parancsok vagy az állományvége jelzés begépeléséig.

9.6. Más nyomtatási rendszerek

Ha derekasan végigolvastuk eddig ezt a fejezetet, akkor mostanra már valószínűleg mindent tudunk a FreeBSD-ben található **LPD** nyomtatási rendszerről. Ezzel együtt tisztában vagyunk a hiányosságaival is, aminek kapcsán természetes módon felmerülhet bennünk a kérdés: “Milyen más (FreeBSD-vel is működni képes) nyomtatási rendszerek léteznek még?”

LPRng

Az **LPRng**, aminek jelentése “LPR Next Generation” (Az LPR következő generációja), a PLP teljesen újraírt változata. Patrick Powell és Justin Mason (a PLP eredeti karbantartója) együttes munkájának gyümölcse az **LPRng**. Az **LPRng** honlapja: <http://www.lprng.org/>.

CUPS

A **CUPS**, vagy más néven a “Common UNIX Printing System” (Közös UNIX-os nyomtatási rendszer), egy hordozható nyomtatási réteget nyújt a UNIX-alapú operációs rendszerek számára. Az Easy Software Products fejlesztése és szinte az összes UNIX gyártó és felhasználó szemében elfogadott szabványos nyomtatási rendszer.

A **CUPS** a nyomtatási feladatok és sorok kezelését az internetes nyomtatási protokollon (Internet Printing Protocol, IPP) használatával oldja meg. Csökkentett képességekkel ugyan, de a sornyomtató démon (Line Printer Daemon, LPD), szerverüzenet-blokk (Server Message Block, SMB), és AppSocket (más néven

JetDirect) protokollokat is ismeri. A CUPS a komolyabb UNIX-os nyomtatási feladatokhoz ezeken felül még a hálózati nyomtatók közti választást és PostScript nyomtatók leírásán (PostScript Printer Description, PPD) alapuló nyomtatási beállításokat is támogatja.

A **CUPS** honlapja: <http://www.cups.org/>.

HPLIP

A **HPLIP**, másnéven HP Linux Imaging and Printing, egy HP által kidolgozott programcsalád, amely támogatja a HP eszközök nyomtatási, lapolvasási és faxolási lehetőségeit. A benne található programok bizonyos nyomtatási feladatokhoz backendként a **CUPS** nyomtatási rendszert használják.

A **HPLIP** honlapja a <http://hplipopensource.com/hplip-web/index.html> címen érhető el.

9.7. Hibakeresés

Miután az `lpctest(1)` programmal elvégeztünk néhány egyszerű próbát, a várt helyett a következők egyikét kaphatjuk eredményül:

Egy kis idő után minden remekül működött, vagy nem dobta ki az egész lapot.

A nyomtató nyomtatott egy keveset, aztán egy ideig csendben maradt és nem csinált semmit. Ilyenkor a nyomtatnivalók megjelenéséhez minden bizonnyal meg kell nyomnunk a nyomtatón levő "PRINT REMAINING" vagy "FORM FEED" feliratú gombokat.

Ebben az esetben a nyomtató valószínűleg még arra vár, hogy még a nyomtatás megkezdése előtt érkezik valamilyen további adat. Ettől a gondtól úgy szabadulhatunk meg, ha beállítunk egy szövegszűrőt, amely minden (szükséges) esetben küld egy "FORM FEED" (lapdobás) jelzést is a nyomtatónak. Ez kell általában ahhoz, hogy a szövegnek a nyomtató belső pufferében megmaradt része azonnal kinyomtatódjon. Akkor is a javunkra válhat ez, ha minden egyes nyomtatási feladatot külön lapon akarunk kezdeni, mivel így a következő nyomtatási feladat sosem közvetlenül ott kezdődik, ahol az előző feladat befejezte a nyomtatást.

A `/usr/local/libexec/if-simple` szűrő helyett a következő szkript használhatóval tudunk minden nyomtatási feladat elvégzése után elküldeni egy lapdobást:

```
#!/bin/sh
#
# if-simple - Egyszerű lpd szövegszűrő
# Helye: /usr/local/libexec/if-simple
#
# Egyszerűen átmásolja a szabvány bemenetet a szabvány kimenetre, és
# figyelmen kívül hagyja az összes többi paramétert. Minden nyomtatási
# nyomtatási feladat elvégzése után küld egy lapdobást (\f).

/bin/cat && printf "\f" && exit 0
exit 2
```

"Lépcsősen" jelentek meg a sorok.

Ekkor a következőt látjuk a lapon:

```
! "#$%&' ( ) * + , - . / 0 1 2 3 4
      "#$%&' ( ) * + , - . / 0 1 2 3 4 5
```

#\$%&'()*+,-./0123456

Az ún. *lépcsőhatás* áldozatává váltunk, amelyet a sortörést jelző karakter eltérő értelmezései okoznak. A UNIX stílusú operációs rendszerek erre mindössze egyetlen karaktert használnak: ez a 10-es kódú ASCII karakter (sordobás, Line Feed, LF). Az MS-DOS, OS/2® és mások pedig két karakterrel oldják meg ezt a feladatot: a 10-es és 13-as kódú (kocsivissza, Carriage Return, CR) ASCII karakterekkel. A sortöréseknél sok nyomtató az MS-DOS szokásait követi.

Amikor a FreeBSD-vel nyomtatunk, akkor csak egyetlen karaktert használunk sortörésre. Ennek láttán a nyomtató lépteti a sort, azonban a fej vízszintes pozícióját nem változtatja meg a következő sor nyomtatásának megkezdésekor. Erre lenne a kocsivissza karakter, vagyis ennek hatására fogja a nyomtató a papír bal oldalára visszaállítani a következő nyomtatandó karakter pozícióját.

A FreeBSD így szeretné utasítani a nyomtatót:

A nyomtató kocsivisszát kap

A nyomtató visszalépteti a pozíciót

A nyomtató sordobást kap

A nyomtató új sort kezd

Néhány módszer ennek kiváltására:

- A nyomtatón található kapcsolók vagy vezérlőpanel segítségével próbáljuk meg átállítani a vezérlőkarakterek nyomtató szerinti értelmezését. Keressük meg a nyomtató kézikönyvében, hogyan tudjuk ezt megcsinálni.

Megjegyzés: Ha a FreeBSD mellett más operációs rendszerekkel is használni akarjuk a nyomtatót, akkor azok indítása előtt mindig *át kell állítani* a nyomtatót a megfelelő értelmezés alkalmazására. Ilyenkor valószínűleg a lassabb szereplő megoldásokat részesítjük majd inkább előnyben.

- Állítsuk be úgy a FreeBSD soros vonali meghajtóját, hogy magától alakítsa át az LF karaktereket CR+LF párokká. Természetesen ez a megoldás *csak* a soros portra csatlakozó nyomtatók esetében működhet. Ehhez az `/etc/printcap` állományban a nyomtató leírásánál az `ms#` tulajdonságnál adjuk meg az `onlcr` módot.
- Küldjünk olyan *kódot* a nyomtatónak, amelynek hatására ideiglenesen máshogy fogja kezelni az LF karaktereket. Nézzük meg a nyomtatóhoz mellékelt útmutatóban, hogy milyen kódokat tudunk ilyen célra használni. Ha találtunk ilyen kódot, akkor írjuk át úgy a hozzá tartozó szövegszűrőt, hogy a nyomtatási feladatok előtt mindig elküldjük azt.

Most bemutatjuk egy olyan szövegszűrő kódját, amely a Hewlett-Packard PCL kódjait ismerő nyomtatókhoz készült. Ebben a szűrőben először kiadjuk, hogy az LF karaktereket LF és CR karakterek kombinációjának tekintse a nyomtató, majd elküldjük magát a nyomtatási feladatot, és a nyomtatási feladat eredményének utolsó lapja után elküldünk egy lapdobást. Szinte az összes Hewlett Packard nyomtatóval működnie kell.

```
#!/bin/sh
#
# hpif - Egyszerű lpd bemeneti szűrő a HP-PCL alapú nyomtatókhoz
# Helye: /usr/local/libexec/hpif
#
# Egyszerűen átmásolja a szabvány kimenetet a szabvány bemenetre, és
# figyelmen kívül hagyja a paramétereket. Elküldi a nyomtatónak, hogy
# az LF karaktereket CR+LF-ként kezelje, majd a feladat befejeztével
# lapot dobát.
```

```
printf "\033&k2G" && cat && printf "\033&l0H" && exit 0
exit 2
```

Példaként megadjuk még az orchid nevű számítógép /etc/printcap állományát is. Ebben egyetlen nyomtató csatlakozik a párhuzamos portra, amelynek a típusa LaserJet 3Si és a neve teak. Az előbb bemutatott szövegszűrőt használja:

```
#
# /etc/printcap (orchid)
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:
```

Egymásra írja a sorokat.

A nyomtató nem lépteti a sorokat, ezért az összes sor egymáson jelenik meg.

Ez pontosan a ritka “ellentéte” a fentebb leírt lépcsőhatásnak. A FreeBSD által sortörésre használt LF karakterek valamiért CR karakterekként viselkednek, ezért a nyomtató nem sort vált, hanem a lap bal szélére állítja a fejet.

A nyomtatón található kapcsolókkal vagy vezérlőpanellel így állítsuk be a sordobás és kocsivissza karakterek értelmezését:

Amit a nyomtató kap	Arra a nyomtató nyomtat
CR	CR
LF	CR + LF

A nyomtató elhagy karaktereket.

Miközben nyomtatunk, a nyomtató bizonyos karaktereket nem hajlandó megjeleníteni. A probléma ennél nagyobb, ha a nyomtató működése közben egyre több és több karaktert hagy ki.

Itt az a gond, hogy a nyomtató nem képes tartani az iramot a számítógép által a soros vonalon átküldött adatok sebességével (ez a probléma nem jelentkezhet a párhuzamos nyomtatók esetén). Két módon kerekedhetünk felül ezen:

- Ha a nyomtató ismeri a XON/XOFF típusú forgalomirányítást, akkor az ms# tulajdonságnál adjuk meg a FreeBSD számára az ixon beállítást.
- Ha a nyomtató ismeri a “Request to Send / Clear to Send” alapú hardveres kézfogást (más néven RTS/CTS forgalomirányítást), akkor az ms# tulajdonságnál a crtscts beállítást adjuk meg. Gondoskodjunk róla, hogy a számítógépet és a nyomtatót összekötő kábel meg tudja majd birkózni ezzel a típusú forgalomirányítással.

Mindenféle szemetet nyomtat.

A nyomtató nem a nyomtatni kívánt szöveget hozza létre, hanem összevissza nyomtat.

Ez a soros nyomtatók helytelen kommunikációs beállításának egy másik jellemző tünete. Ellenőrizzük a br tulajdonságnál megadott adatátviteli sebességet és az ms# tulajdonságnál megadott paritási beállításokat. Egyeztessük a nyomtató saját és az /etc/printcap állományban tárolt beállításait.

Semmi sem történik.

Ha semmi sem történt, akkor a gond magával a FreeBSD-vel lehet, nem pedig a hardverrel. Az `/etc/printcap` állományba a vizsgálni kívánt nyomtató leírásához (az `lf` tulajdonsággal) illesszünk be naplózást. Például így fog kinézni a rattan nevű nyomtató bejegyzése az `lf` tulajdonság megadásával kibővíve:

```
rattan|line|diablo|lp|Diablo 630 Line Printer:\
      :sh:sd=/var/spool/lpd/rattan:\
      :lp=/dev/lpt0:\
      :if=/usr/local/libexec/if-simple:\
      :lf=/var/log/rattan.log
```

Miután ezt megcsináltuk, próbálkozzunk újra. Nézzük meg a naplóállományban (ami a példánkban a `/var/log/rattan.log` nevén érhető el), hogy látunk-e valamilyen hibaüzenetet. Az itt tapasztalt hibaüzenetek nyomán elindulva igyekezzünk megszüntetni a probléma forrását.

Ha nem adjuk meg az `lf` tulajdonságot, akkor az **LPD** erre a célra alapértelmezés szerint a `/dev/console` állományt használja.

Fejezet 10. Bináris Linux kompatibilitás

Átdolgozta és egyes részeit aktualizálta: Jim Mock. Eredetileg írta: Brian N. Handy és Rich Murphey.

10.1. Áttekintés

A FreeBSD számos más UNIX-szerű operációs rendszerhez nyújt bináris kompatibilitást, köztük a Linuxhoz is. Elcsodálkozhatnánk rajta, hogy vajon miért kell tudnia a FreeBSD-nek Linux binárisokat futtatnia. A válasz erre nagyon egyszerű. Rengeteg cég és fejlesztő kizárólag csak Linuxra fejleszt, hiszen ez mostanság egy nagyon “izgalmas téma” az informatika világában. Emiatt azonban a FreeBSD közösségnek külön győzködni kell ezeket a cégeket és fejlesztőket, hogy készítsék el a termékeik natív FreeBSD-s változatát. Ezzel az a gond, a legtöbb ilyen cég egyszerűen nem veszi észre, hogy ha létezne a terméküknek FreeBSD-re írt változata, akkor még többen használnák. Így továbbra is csak Linuxra fejlesztenek. Mit tudnak tenni ilyenkor a FreeBSD használói? Nos, ekkor jön jól a FreeBSD bináris szintű kompatibilitása.

Dióhéjban úgy tudnánk összefoglalni, hogy ennek köszönhetően a FreeBSD felhasználók képesek a linuxos alkalmazások közel 90%-át mindenféle további módosítás nélkül futtatni. Így tehát használható a **StarOffice**, **Netscape** Linux változata, az **Adobe Acrobat**, **RealPlayer**, **VMware™**, **Oracle**, **WordPerfect®**, **Doom**, **Quake**, és még sok minden más. Sőt, egyes tapasztalatok szerint bizonyos helyzetekben a FreeBSD által futtatott Linux binárisok sokkal jobban teljesítenek, mint Linux alatt.

Azonban vannak olyan Linuxra jellemző, az operációs rendszer szintjén meghúzódó eszközök, amelyek FreeBSD alatt nem használhatóak. FreeBSD-n nem fognak működni azok a Linux binárisok, amelyek túlzottan kihasználják az olyan i386-os rendszerhívásokat, mint például a virtuális 8086 mód.

A fejezet elolvasása során megismerjük:

- hogyan engedélyezzük rendszerünkön a Linux kompatibilitást;
- hogyan telepítsünk linuxos osztott könyvtárakat;
- hogyan telepítsünk linuxos alkalmazásokat a FreeBSD rendszerünkre;
- a FreeBSD Linux kompatibilitásának implementációs részleteit.

A fejezet elolvasásához ajánlott:

- külső szoftverek telepítésének ismerete (4 fejezet).

10.2. Telepítés

A bináris Linux kompatibilitás alapértelmezés szerint nem engedélyezett. Legkönnyebben úgy tudjuk elérhetővé tenni, ha betöltjük a `linux` nevű KLD modult (“Kernel Loadable”). Ehhez `root` felhasználóként a következőket kell begépelni:

```
# kldload linux
```

Ha minden egyes rendszerindítás során engedélyezni szeretnénk a bináris kompatibilitást, akkor tegyük bele az `/etc/rc.conf` állományba ezt a sort:

```
linux_enable="YES"
```

A modul betöltődését a `kldstat(8)` paranccsal tudjuk ellenőrizni:

```
% kldstat
Id Refs Address      Size      Name
  1    2 0xc0100000 16bdb8    kernel
  7    1 0xc24db000 d000      linux.ko
```

Ha valamiért nem akarjuk vagy nem éppen nem tudjuk betölteni a modult, akkor a bináris Linux kompatibilitást az `options COMPAT_LINUX` beállítással be is tudjuk építeni a rendszermagba. Ennek pontos menetét a 8 fejezetben találjuk meg.

10.2.1. Linuxos futtatókönyvtárak telepítése

A linuxos könyvtárakat két módon is felrakhatjuk: egyrészt a `linux_base` port telepítésével, másrészt manuálisan.

10.2.1.1. A könyvtárak telepítése a `linux_base` porttal

A futtatókönyvtárakat a lehető legegyszerűbben a `emulators/linux_base` porton keresztül tudjuk telepíteni. Teljesen úgy történik, mint a Portgyűjtemény (`/usr/ports/`) akármelyik másik portjának telepítése. Csupán ennyit kell beírunk:

```
# cd /usr/ports/emulators/linux_base-f10
# make install distclean
```

Megjegyzés: A FreeBSD 8.0 kiadását megelőző változataiban az `emulators/linux_base-f10` port helyett az `emulators/linux_base-fc4` portot használjuk.

A telepítés végeztével kaptunk is egy működő bináris Linux kompatibilitást, habár egyes programok még panaszkodhatnak a rendszerkönyvtárak alverzióit illetően. Általánosságban véve ez azonban nem okoz nagyobb gondot.

Megjegyzés: A `emulators/linux_base` portnak több változata is használható, melyek az egyes Linux disztribúcióknak feleltethetők meg. Ilyenkor mindig érdemes közülük azt választani, amelyik a leginkább megfelel a telepíteni kívánt linuxos alkalmazás igényeinek.

10.2.1.2. A könyvtárak telepítése manuálisan

Ha korábban még nem telepítettük volna a Portgyűjteményt, akkor egyénileg kell felraknunk az egyes könyvtárakat. Közülük azokra lesz szükségünk, amelyeket maga az alkalmazás is használni akar, valamint a futásidejű linkerre. Emellett még a FreeBSD rendszerünkön levő Linux binárisok számára a `/compat/linux` könyvtárban létre kell hoznunk a gyökér ún. “árnyékkönyvtárát” is. A FreeBSD alatt elindított Linux programok először ebben a könyvtárban fogják keresni a hozzájuk tartozó osztott könyvtárakat. Így tehát, amikor egy linuxos program betölti például a `/lib/libc.so` függvénykönyvtárat, akkor a FreeBSD először a `/compat/linux/lib/libc.so` állományt próbálja meg megnyitni, majd ha az nem létezik, akkor a `/lib/libc.so` állományt. Az osztott

könyvtárak ezért a `/compat/linux/lib` árnyékkönyvtárba telepítendőek, és nem oda, ahova a linuxos `ld.so` mutat.

Általánosságban szólva eleinte elég csak azokat az osztott könyvtárakat megkeresni és felrakni, amelyekre a telepítendő linuxos alkalmazásunknak ténylegesen szüksége van. Egy idő után úgyis összegyűlnek azok a fontosabb függvénykönyvtárak, amelyek segítségével már minden további ráfordítás nélkül futtatni tudjuk a frissen importált programokat.

10.2.1.3. Hogyan telepítsünk újabb osztott könyvtárakat?

Mit tegyünk, ha az `emulators/linux_base` port telepítése után az alkalmazás még mindig hiányol néhány osztott könyvtárat? Honnan tudhatjuk meg, hogy milyen osztott könyvtárak kellenek majd egy Linux bináris használatához, és honnan szerezzük be ezeket? Erre alapvetően két lehetőségünk van (az utasításokat `root` felhasználóként kell majd végrehajtanunk).

Ha hozzáférünk egy Linux rendszerhez, akkor szedjük össze az alkalmazásunk futtatásához szükséges osztott könyvtárakat, és másoljuk ezeket a FreeBSD partíciójára. Például:

Tegyük fel, hogy FTP-n keresztül leszedtük a **Doom** Linux változatát, és felraktuk egy általunk elérhető Linux rendszerre. Az `ldd linuxdoom` parancs segítségével ki tudjuk deríteni, milyen osztott könyvtárak kellenek majd nekünk:

```
% ldd linuxdoom
libXt.so.3 (DLL Jump 3.1) => /usr/X11/lib/libXt.so.3.1.0
libX11.so.3 (DLL Jump 3.1) => /usr/X11/lib/libX11.so.3.1.0
libc.so.4 (DLL Jump 4.5p126) => /lib/libc.so.4.6.29
```

Az utolsó oszlopban levő állományokat másoljuk át, tegyük ezeket a `/compat/linux` könyvtárba, és hozzunk létre az első oszlopban szereplő szimbolikus linkeket. Így tehát a következő állományok kellenének:

```
/compat/linux/usr/X11/lib/libXt.so.3.1.0
/compat/linux/usr/X11/lib/libXt.so.3 -> libXt.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3 -> libX11.so.3.1.0
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

Megjegyzés: Ha már rendelkezünk az `ldd` kimenetének első oszlopában szereplő főverziószámú osztott könyvtárral, akkor nem kell átmásolni az utolsó oszlopban levő állományokat, hiszen így is működnie kellene mindennek. Ha viszont egy újabb változattal találkozunk, akkor érdemes mégis inkább átmásolni. Miután a szimbolikus linkeket átirányítottuk az új változatra, a régit akár törölhetjük is. Ha például ezek a könyvtárak elérhetőek a rendszerünkön:

```
/compat/linux/lib/libc.so.4.6.27
/compat/linux/lib/libc.so.4 -> libc.so.4.6.27
```

Észrevesszük, hogy az `ldd` kimenetében az új bináris egy újabb változatot igényel:

```
libc.so.4 (DLL Jump 4.5p126) -> libc.so.4.6.29
```

Ha csak az utolsó jegyében marad le valamivel a verziószám, akkor nem kell különösebben aggódnunk a `/lib/libc.so.4.6.29` miatt sem, hiszen a programnak egy picivel korábbi verzióval is remekül kellene tudnia működni. Természetesen, ha akarjuk, ettől függetlenül lecserélhetjük a `libc.so` állományt, ami ezt eredményezi:

```
/compat/linux/lib/libc.so.4.6.29
```

```
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

Megjegyzés: A szimbolikus linkek karbantartása *csak* a Linux binárisok esetén szükséges. A FreeBSD saját futásidejű linkere magától megkeresi a megfelelő főverziószámú könyvtárakat, ezért emiatt általában nem kell aggódni.

10.2.2. Linux ELF binárisok telepítése

Az ELF binárisok futtatása előtt néha még szükség van a “megbélyegzés” (branding) használatára is. Ha egy bélyegezetlen ELF binárist akarunk elindítani, akkor a következő hibaüzenetet kapjuk:

```
% ./egy-linux-elf-bináris
ELF binary type not known
Abort
```

A FreeBSD rendszermagjának a `brandelf(1)` paranccsal tudunk segíteni a FreeBSD és a Linux binárisainak megkülönböztetésében.

```
% brandelf -t Linux egy-linux-elf-bináris
```

A GNU által fejlesztett eszközök manapság már automatikusan elhelyezik az ELF binárisok azonosításához szükséges bélyegeket, ezért ez a lépés a jövőben egyre inkább feleslegessé válik.

10.2.3. Tetszőleges RPM formátumú csomag telepítése

A FreeBSD a telepített (akár linuxos) alkalmazások nyomkövetésére saját csomagadatbázissal rendelkezik, amelynek következtében a Linux által felkínált RPM adatbázisokat nem támogatja.

Ennek ellenére akármelyik RPM alapú Linux alkalmazás telepíthető rendszerünkre a következő módon:

```
# cd /compat/linux
# rpm2cpio -q < /a/linuxos/allomány.hegy.rpm | cpio -id
```

Ezt követően a `brandelf(1)` segítségével állítsuk be az ELF binárisokat (könyvtárakat viszont ne!) megfelelő típusúra. Ekkor ugyan nem leszünk képesek rendesen eltávolítani az így telepített szoftvert, de ez a módszer teszteléshez megfelelő.

10.2.4. A névfeloldó beállítása

Ha a névfeloldás (DNS) valamiért nem működne, vagy egy ehhez hasonló üzenetet kapunk:

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

Akkor a `/compat/linux/etc/host.conf` állományba be kell illesztenünk a következő sorokat:

```
order hosts, bind
multi on
```

Az itt megszabott sorrend szerint először az `/etc/hosts` állományt nézi át, és majd csak ezután próbálja meg feloldani a nevet. Ha a `/compat/linux/etc/host.conf` állomány nem létezik, akkor a linuxos alkalmazás a FreeBSD `/etc/host.conf` állományát találja meg, és panaszkodni fog a FreeBSD eltérő formátumára. Távolítsuk el a `bind` szócskát, ha nem állítottunk be névszerveret az `/etc/resolv.conf` állományhoz.

10.3. A Mathematica® telepítése

A Mathematica 5.X verziójához igazította: Boris Hollas.

Ebben a szakaszban megismerhetjük, hogyan telepítsük a **Mathematica 5.X** Linux változatát FreeBSD rendszerekre.

A **Mathematica** vagy a **Mathematica for Students** linuxos változatai közvetlenül megrendelhetők a fejlesztőtől: <http://www.wolfram.com/>.

10.3.1. A Mathematica telepítőjének elindítása

Először is jeleznünk kell a FreeBSD-nek, hogy a **Mathematica** binárisai a linuxos ABI-t (Application Binary Interface) fogják használni. Itt legkönnyebben úgy járhatunk el, ha egyszerűen beállítjuk, hogy a rendszer a bélyegezetlen ELF binárisokat automatikusan Linux binárisoknak tekintse:

```
# sysctl kern.fallback_elf_brand=3
```

Ennek köszönhetően a FreeBSD most már az összes bélyegezetlen ELF bináris esetén a linuxos ABI-t fogja használni, és így a telepítőt akár már közvetlenül a CD-ről is indíthatjuk.

Most másoljuk át a `MathInstaller` nevű állományt a merevlemezünkre:

```
# mount /cdrom
# cp /cdrom/Unix/Installers/Linux/MathInstaller helyi_könyvtár
```

Az állományban cseréljük ki az első sorban található `/bin/sh` hivatkozást a `/compat/linux/bin/sh` hivatkozásra. Ezzel biztosíthatjuk, hogy a telepítőt a linuxos `sh(1)` fogja elindítani. Ezután a kedvenc szövegszerkesztőnkkel vagy a következő szakaszban található szkript segítségével helyettesítsük benne a `Linux`) szöveg összes előfordulását a `FreeBSD`) szöveggel. Mivel a **Mathematica** telepítője az `uname -s` parancsra kapott válaszból állapítja meg az operációs rendszer típusát, ezért ezzel a módosítással a FreeBSD-t is a Linuxhoz hasonló módon fogja kezelni. A `MathInstaller` elindítása után most már telepíthető a **Mathematica**.

10.3.2. A Mathematica állományainak módosítása

A **Mathematica** telepítése során létrejött szkripteket a használatuk előtt át kell írunk. Amennyiben a **Mathematica**hoz tartozó programokat a `/usr/local/bin` könyvtárba telepítettük, akkor itt találjuk a `math`, `mathematica`, `Mathematica` és `MathKernel` állományokra mutató szimbolikus linkeket. Ezek mindegyikében cseréljük ki a `Linux`) karakterláncot a `FreeBSD`) szövegre a kedvenc szövegszerkesztőnkkel vagy az alábbi szkripttel:

```
#!/bin/sh
cd /usr/local/bin
for i in math mathematica Mathematica MathKernel
do sed 's/Linux)/FreeBSD)/g' $i > $i.tmp
sed 's/\\bin\\sh\\/compat\\/linux\\/bin\\/sh/g' $i.tmp > $i
rm $i.tmp
chmod a+x $i
done
```

10.3.3. A Mathematica jelszavának megszerzése

A **Mathematica** első indítása során kérni fog egy jelszót. Ha még nem kértünk volna jelszót a fejlesztőtől, akkor a “számítógépünk azonosítójának” (machine ID) megállapításához indítsuk el a telepítés könyvtárában található `mathinfo` nevű programot. Ez az azonosító lényegében az elsődleges Ethernet kártyánk MAC-címe lesz, ezért a **Mathematica** nem futtatható több számítógépen.

Amikor e-mailen, telefonon vagy faxon keresztül regisztráljuk a terméket a Wolframnál, akkor meg kell adnunk nekik ezt az azonosítót “machine ID” néven, amire ők elküldik a hozzá tartozó jelszót.

10.3.4. A Mathematica frontendjének futtatása hálózaton keresztül

A **Mathematica** a szabványos betűkészletekkel meg nem jeleníthető szimbólumokhoz (integráljelek, szummák, görög betűk, matematikai jelölések stb.) használ néhány olyan speciális betűtípust, amelyek nem minden esetben állnak rendelkezésre. Az X által használt protokoll miatt ezeket a betűtípusokat *helyben* kell telepíteni. Ennek értelmében a **Mathematica** CD-jén található betűtípusokat telepítenünk kell a számítógépünkre is. A CD-n ezeket általában a `/cdrom/Unix/Files/SystemFiles/Fonts` könyvtárban találjuk meg, vagy a merevlemezen a `/usr/local/mathematica/SystemFiles/Fonts` könyvtárban. Ezen belül pedig a `Type1` és `X` alkönyvtárakra van szükségünk. Az alábbiakban leírtak szerint több módon is használhatjuk ezeket.

Az egyik ilyen módszer, ha átmásoljuk az imént említett könyvtárakat a többi mellé, vagyis a `/usr/X11R6/lib/X11/fonts` könyvtárba. Ekkor szükségünk lesz még a `fonts.dir` állomány átírására is, ahova fel kell vennünk a betűtípusok neveit, majd ennek megfelelően az első sorban módosítanunk a könyvtárban található betűtípusok számát. De ugyanígy lefuttathatjuk ebben a könyvtárban a `mkfontdir(1)` parancsot is.

Az a másik megoldás, ha a könyvtárakat így másoljuk át a `/usr/X11R6/lib/X11/fonts` helyre:

```
# cd /usr/X11R6/lib/X11/fonts
# mkdir X
# mkdir MathType1
# cd /cdrom/Unix/Files/SystemFiles/Fonts
# cp X/* /usr/X11R6/lib/X11/fonts/X
# cp Type1/* /usr/X11R6/lib/X11/fonts/MathType1
# cd /usr/X11R6/lib/X11/fonts/X
# mkfontdir
# cd ../MathType1
# mkfontdir
```

Most adjuk hozzá az új könyvtárakat a betűtípusok könyvtáraihoz:

```
# xset fp+ /usr/X11R6/lib/X11/fonts/X
# xset fp+ /usr/X11R6/lib/X11/fonts/MathType1
```

```
# xset fp rehash
```

Ha az **Xorg** szerveret használjuk, akkor az `xorg.conf` állományban megadhatjuk ezen könyvtárak automatikus betöltését is.

Megjegyzés: Az **XFree86** típusú szerverek esetén az `XF86Config` konfigurációs állományt kell módosítanunk.

Ha még *nincs* `/usr/X11R6/lib/X11/fonts/Type1` nevű könyvtárunk, akkor a példában szereplő `MathType1` könyvtárat nyugodtan átnevezhetjük `Type1` névre.

10.4. A Maple™ telepítése

Írta: Aaron Kaplan. Köszönet: Robert Getschmann.

A **Maple™** egy **Mathematica**hoz hasonló kereskedelmi alkalmazás. A használatához először meg kell vásárolni a <http://www.maplesoft.com/> címről, majd a licenc megszerzéséhez ugyanott regisztrálni. FreeBSD-re a szoftvert a következő egyszerű lépéseken keresztül tudjuk telepíteni.

1. Indítsuk el a termékhez mellékelt `INSTALL` nevű szkriptet. Válasszuk a telepítőprogram által felkínált opciók közül a “RedHat” címkéjűt. A telepítés célkönyvtára legyen a `/usr/local/maple`.
2. Ha eddig még nem tettük volna meg, rendeljük meg a **Maple** licencét a Maple Waterloo Software-től (<http://register.maplesoft.com/>) és másoljuk az `/usr/local/maple/license/license.dat` állományba.
3. Az **Maple**-höz mellékelt `INSTALL_LIC` szkript elindításával telepítsük a **FLEXlm** licenckezelőt. A szervernek adjuk meg a számítógépünk hálózati nevét.
4. Javítsuk át a `/usr/local/maple/bin/maple.system.type` állományt a következő módon:

```
----- itt kezdődik a módosítás -----
*** maple.system.type.orig      Sun Jul  8 16:35:33 2001
--- maple.system.type      Sun Jul  8 16:35:51 2001
*****
*** 72,77 ***
--- 72,78 ----
        # the IBM RS/6000 AIX case
        MAPLE_BIN="bin.IBM_RISC_UNIX"
        ;;
+   "FreeBSD" |\
    "Linux")
        # the Linux/x86 case
        # We have two Linux implementations, one for Red Hat and
----- módosítás vége -----
```

Vigyázzunk, hogy a `"FreeBSD" |\` kezdetű sor végén nem szabad semmilyen további whitespace karakternek lennie.

Ez a javítás arra utasítja a **Maple**-t, hogy a “FreeBSD”-t Linux rendszerként ismerje fel. A `bin/maple` szkript hívja a `bin/maple.system.type` szkriptet, amely pedig a `uname -a` hívással próbálja kideríteni az operációs rendszer nevét. Ettől függően választja ki, hogy milyen típusú binárisokat fog futtatni.

5. Indítsuk el a licenckezelő szerveret.

A most következő szkripttel könnyedén el tudjuk indítani az `lmgrd` programot. A szkriptet `/usr/local/etc/rc.d/lmgrd.sh` néven hozzuk létre:

```
----- nyissz -----

#!/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin
PATH=${PATH}:/usr/local/maple/bin:/usr/local/maple/FLEXlm/UNIX/LINUX
export PATH

LICENSE_FILE=/usr/local/maple/license/license.dat
LOG=/var/log/lmgrd.log

case "$1" in
start)
    lmgrd -c ${LICENSE_FILE} 2>> ${LOG} 1>&2
    echo -n " lmgrd"
    ;;
stop)
    lmgrd -c ${LICENSE_FILE} -x lmdown 2>> ${LOG} 1>&2
    ;;
*)
    echo "Usage: `basename $0` {start|stop}" 1>&2
    exit 64
    ;;
esac

exit 0

----- nyissz -----
```

6. Próbáljuk meg elindítani a **Maple**-t:

```
% cd /usr/local/maple/bin
% ./xmaple
```

Szerencsés esetben innentől kezdve már minden működik. És ne felejtünk el írni a Maplesoftnak, hogy szeretnénk egy natív FreeBSD verziót a termékükből!

10.4.1. Általános buktatók

- A **FLEXlm** licenckezelővel esetenként nehéz lehet elboldogulni. Erről a témáról bővebben a <http://www.globetrotter.com/> címen találunk leírásokat.
- Az `lmgrd` nagyon válogatós a licencállományokat illetően és bármilyen apróságra kiakad. Egy szabályos licencállomány valahogy így néz ki:

```
# =====
# License File for UNIX Installations ("Pointer File")
# =====
SERVER chillig ANY
#USE_SERVER
VENDOR maplelmg
```

```
FEATURE Maple maplelmg 2000.0831 permanent 1 XXXXXXXXXXXX \
  PLATFORMS=i86_r ISSUER="Waterloo Maple Inc." \
  ISSUED=11-may-2000 NOTICE=" Technische Universitat Wien" \
  SN=XXXXXXXXXX
```

Megjegyzés: A sorozatszámot természetesen eltávolítottuk. Itt a `chillig` a számítógép neve.

Az itt megadott licencállomány remekül használható egészen addig a pontig, amíg békén hagyjuk a “FEATURE” kezdetű sort (melyet a licenckulcs véd).

10.5. A MATLAB® telepítése

Írta: Dan Pelleg.

Ez a leírás azt mutatja be, hogyan telepítsük FreeBSD rendszerekre a **MATLAB® version 6.5** Linux változatát. A **Java Virtual Machine™** (lásd 10.5.3 Szakasz) használatától eltekintve meglepően jól működik.

A **MATLAB** Linux változata közvetlenül megrendelhető a The MathWorks-től, a <http://www.mathworks.com> címen. Ne felejtsük el beszerezni a licencállományt és az elkészítéséhez szükséges útmutatót. Ha már úgyis arra járunk, jelezzük a fejlesztőknek, hogy igényt tartanánk a termékük natív FreeBSD-s változatára is!

10.5.1. A MATLAB telepítése

A **MATLAB** telepítéséhez a következőket kell tennünk:

1. Helyezzük be a telepítő CD-t és csatlakoztassuk. A telepítőszkript javaslatának megfelelően váltsunk át a `root` felhasználóra. A szóbanforgó szkript elindításához gépeljük be a következőt:

```
# /compat/linux/bin/sh /cdrom/install
```

Tipp: A telepítő grafikus. Ha a megjelenítő használatáról szóló hibaüzeneteket kapunk, akkor adjuk ki a `setenv HOME ~FELHASZNÁLÓ` parancsot, ahol a `FELHASZNÁLÓ` annak a felhasználónak a neve legyen, amivel az imént meghívtuk a `su(1)` programot.

2. Amikor a **MATLAB** könyvtárát kell megadnunk, ezt írjuk be: `/compat/linux/usr/local/matlab`.

Tipp: A telepítés további részeinek megkönnyítése érdekében írjuk be ezt a parancssorba: `set MATLAB=/compat/linux/usr/local/matlab`

3. Miután megkaptuk a **MATLAB** licencét, az útmutatás szerint szerkesszük át.

Tipp: A licencállományt a kedvenc szövegszerkesztőnkkel akár már korábban elő is készíthetjük, és majd amikor a telepítőnek szüksége lesz rá, másoljuk be `$MATLAB/license.dat` helyre.

4. Futtassuk le a telepítést.

Ezzel befejeződött a **MATLAB** hagyományos telepítése. Innentől már csak a FreeBSD rendszer “hozzátapasztásán” fogunk dolgozni.

10.5.2. A licenckezelő elindítása

1. Hozzunk létre szimbolikus linkeket a licenckezelő szkriptjeire:

```
# ln -s $MATLAB/etc/lmboot /usr/local/etc/lmboot_TMW
# ln -s $MATLAB/etc/lmdown /usr/local/etc/lmdown_TMW
```

2. Hozzunk létre egy indítószkriptet `/usr/local/etc/rc.d/flexlm.sh` néven. A lentebb látható minta a **MATLAB**hoz mellékelt `$MATLAB/etc/rc.lm.glnx86` állomány egy módosított változata. Benne az állományok helyét és a licenckezelő indításának körülményeit változtattuk meg (hogyan Linux emuláció alatt fusson).

```
#!/bin/sh
case "$1" in
  start)
    if [ -f /usr/local/etc/lmboot_TMW ]; then
      /compat/linux/bin/sh /usr/local/etc/lmboot_TMW -u felhasználó && echo 'MATLAB_lmg
    fi
    ;;
  stop)
    if [ -f /usr/local/etc/lmdown_TMW ]; then
      /compat/linux/bin/sh /usr/local/etc/lmdown_TMW > /dev/null 2>&1
    fi
    ;;
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac

exit 0
```

Fontos: Tegyük ezt az állományt végrehajthatóvá:

```
# chmod +x /usr/local/etc/rc.d/flexlm.sh
```

A fenti szkriptben cseréljük ki a `felhasználó` nevét a rendszerünkben levő egyik felhasználó nevére (ami persze nem a `root`).

3. A licenckezelőt az alábbi paranccsal indítsuk el:

```
# /usr/local/etc/rc.d/flexlm.sh start
```

10.5.3. A Java futtató környezet élesztése

A **Java** futtató környezet (Java Runtime Environment, JRE) linkjét irányítsuk át egy FreeBSD alatt működő változatára:

```
# cd $MATLAB/sys/java/jre/glnx86/
# unlink jre; ln -s ../jre1.1.8 ./jre
```

10.5.4. A MATLAB indítószkriptjének elkészítése

1. Hozzunk létre egy ilyen indítószkriptet a `/usr/local/bin/matlab` könyvtárban:

```
#!/bin/sh
/compat/linux/bin/sh /compat/linux/usr/local/matlab/bin/matlab "$@"
```

2. Futtassuk le a `chmod +x /usr/local/bin/matlab` parancsot.

Tipp: A szkript lefutása során az `emulators/linux_base` verziójától függően hibákat is kaphatunk. Ha el akarjuk kerülni ezeket, akkor szerkesszük át a `/compat/linux/usr/local/matlab/bin/matlab` állomány következő sorát:

```
if [ `expr "$lscmd" : '.*->.*'` -ne 0 ]; then
```

(a 13.0.1 számú verzióban ez 410. sor) erre:

```
if test -L $newbase; then
```

10.5.5. A MATLAB leállító szkriptjének elkészítése

A MATLAB szabálytalan kilépéseit az alábbi utasítások nyomán tudjuk megszüntetni.

1. Hozzunk létre egy `$MATLAB/toolbox/local/finish.m` nevű állományt, majd írjuk bele ezt a sort:

```
! $MATLAB/bin/finish.sh
```

Megjegyzés: A `$MATLAB` szöveget pontosan így írjuk be.

Tipp: Ugyanebben a könyvtárban találjuk a beállításaink kilépés előtti mentéséért felelős `finishsav.m` és `finishdlg.m` állományokat. Ha ezek valamelyikét módosítjuk, akkor az előbbi parancsot közvetlenül a `save` után szűrjük be.

2. Hozzunk létre egy `$MATLAB/bin/finish.sh` állományt, amelyben szerepeljen a következő:

```
#!/usr/compat/linux/bin/sh
(sleep 5; killall -1 matlab_helper) &
```

```
exit 0
```

3. Tegyük végrehajthatóvá:

```
# chmod +x $MATLAB/bin/finish.sh
```

10.5.6. A MATLAB használata

Most már a `matlab` parancs begépelésével bármikor elindíthatjuk.

10.6. Az Oracle® telepítése

Írta: Marcel Moolenaar.

10.6.1. Előszó

Ez a leírás azt mutatja be, hogyan telepítsük FreeBSD-re az **Oracle 8.0.5** és **Oracle 8.0.5.1 Enterprise Edition** Linux változatait.

10.6.2. A Linux környezet telepítése

Telepítsük az `emulators/linux_base` és `devel/linux_devtools` portokat a Portgyűjteményből. Amennyiben ennek során nehézségekbe ütköznénk, próbálkozzunk a korábbi változataikkal.

Fel kell raknunk a Red Hat Tcl csomagját is, ha az alkalmazáshoz tartozó intelligens ügynököt is futtatni szeretnénk. Ez a `tcl-8.0.3-20.i386.rpm`. A hivatalos **RPM** port segítségével az alábbi általános parancson keresztül tudunk csomagokat telepíteni:

```
# rpm -i --ignoreos --root /compat/linux --dbpath /var/lib/rpm csomag
```

A `csomag` telepítésének semmilyen hibát nem kellene okoznia.

10.6.3. Az Oracle környezetének létrehozása

Az **Oracle** telepítéséhez először ki kell alakítanunk a megfelelő környezetet. Ez a leírás *kifejezetten* arról szól, hogy FreeBSD-n hogyan futtassuk a linuxos **Oracle**-t, nem pedig az **Oracle** telepítési útmutatójában bemutatottakat taglalja.

10.6.3.1. A rendszermag hangolása

Ahogy az **Oracle** telepítési útmutatójában is olvashatjuk, be kell állítanunk az osztott memória maximális méretét. FreeBSD alatt erre a célra ne használjuk az `SHMMAX` értéket, mivel az `SHMMAX` az `SHMAXPGS` és `PGSIZE` értékekből számolódik ki. Ezért nekünk itt a `SHMAXPGS` értéket kell meghatároznunk. Minden egyéb beállítás történhet az útmutatóban megadottak szerint. Például:

```
options SHMAXPGS=10000
options SHMMNI=100
```

```
options SHMSEG=10
options SEMMNS=200
options SEMMNI=70
options SEMMSL=61
```

Hangoljuk be ezeket az értékeket az **Oracle** tervezett használatához.

Emellett a konfigurációs állományban ne feledkezzünk meg az alábbi beállítások megadásáról sem:

```
options SYSVSHM #SysV osztott memória
options SYSVSEM #SysV szemaforok
options SYSVMSG #SysV folyamatok közti kommunikáció
```

10.6.3.2. Az Oracle hozzáférése

Egy rendes hozzáféréshez hasonlóan hozzunk létre egy külön oracle hozzáférést is rendszerünkön. Az oracle hozzáférés csak annyiban különleges, hogy linuxos parancsértelmezőt kell társítanunk hozzá. Ehhez vegyük fel `/compat/linux/bin/bash` sort az `/etc/shells` állományba, majd állítsuk át az oracle nevű felhasználó parancsértelmezőjét a `/compat/linux/bin/bash` programra.

10.6.3.3. Környezet

A megszokott **Oracle** környezeti változók, mint például az `ORACLE_HOME` és `ORACLE_SID` mellett még definiálnunk kell a következőket is:

Változó	Érték
<code>LD_LIBRARY_PATH</code>	<code>\$ORACLE_HOME/lib</code>
<code>CLASSPATH</code>	<code>\$ORACLE_HOME/jdbc/lib/classes111.zip</code>
<code>PATH</code>	<code>/compat/linux/bin /compat/linux/sbin</code> <code>/compat/linux/usr/bin /compat/linux/usr/sbin /bin /sbin</code> <code>/usr/bin /usr/sbin /usr/local/bin \$ORACLE_HOME/bin</code>

Javasoljuk, hogy az összes környezeti változót a `.profile` állományban adjuk meg. Ennek megfelelően a példa beállításai így fognak kinézni benne:

```
ORACLE_BASE=/oracle; export ORACLE_BASE
ORACLE_HOME=/oracle; export ORACLE_HOME
LD_LIBRARY_PATH=$ORACLE_HOME/lib
export LD_LIBRARY_PATH
ORACLE_SID=ORCL; export ORACLE_SID
ORACLE_TERM=386x; export ORACLE_TERM
CLASSPATH=$ORACLE_HOME/jdbc/lib/classes111.zip
export CLASSPATH
PATH=/compat/linux/bin:/compat/linux/sbin:/compat/linux/usr/bin
PATH=$PATH:/compat/linux/usr/sbin:/bin:/sbin:/usr/bin:/usr/sbin
PATH=$PATH:/usr/local/bin:$ORACLE_HOME/bin
export PATH
```

10.6.4. Az Oracle telepítése

A Linux emulátorban meghúzódó apró egyenletlenségek miatt a telepítés előtt létre kell hoznunk egy `.oracle` nevű alkönyvtárat a `/var/tmp` könyvtárban. Helyezzük ezt az `oracle` felhasználó tulajdonába. Ezt követően minden további gond nélkül képesek leszünk az **Oracle** telepítésére. Ha netalán mégis problémákba ütköznénk, először mindig az **Oracle** telepítési és konfigurációs állományait ellenőrizzük! Az **Oracle** telepítése után rakjuk fel a következő szakaszokban bemutatandó javításokat.

Gyakran problémát okoz, ha a TCP protokollt még nem telepítettük. Ennek következményeképpen ugyanis nem tudnak elindulni a TCP alapú szolgáltatások. Az alábbi műveletek ebben igyekeznek segíteni:

```
# cd $ORACLE_HOME/network/lib
# make -f ins_network.mk ntcontab.o
# cd $ORACLE_HOME/lib
# ar r libnetwork.a ntcontab.o
# cd $ORACLE_HOME/network/lib
# make -f ins_network.mk install
```

Ne felejtsük el ismét elindítani a `root.sh` szkriptet!

10.6.4.1. A root.sh javítása

Az **Oracle** telepítése során `root` (privilegizált) felhasználóként elvégzendő műveleteket a `root.sh` elnevezésű szkriptben találjuk. Ez a szkript az `orainst` könyvtárba kerül. A `chown` parancs helyes lefutásához alkalmazzuk az alább mellékelt javítást, vagy az egész szkriptet egy linuxos parancsértelmezőből indítsuk el.

```
*** orainst/root.sh.orig Tue Oct 6 21:57:33 1998
--- orainst/root.sh Mon Dec 28 15:58:53 1998
*****
*** 31,37 ***
# This is the default value for CHOWN
# It will redefined later in this script for those ports
# which have it conditionally defined in ss_install.h
! CHOWN=/bin/chown
#
# Define variables to be used in this script
--- 31,37 ---
# This is the default value for CHOWN
# It will redefined later in this script for those ports
# which have it conditionally defined in ss_install.h
! CHOWN=/usr/sbin/chown
#
# Define variables to be used in this script
```

Ha nem CD-ről telepítjük az **Oracle-t**, akkor akár a `root.sh` forrását is kijavíthatjuk. A neve `rthd.sh`, és a forrásfa `orainst` könyvtárában található.

10.6.4.2. A gencintsh javítása

A `gencintsh` szkript a kliensek által használt osztott könyvtár létrehozására alkalmazható. Általában demók fordításához van rá szükség. Az alábbi javítás alkalmazásával a `PATH` változó értéke törölhető:

```

*** bin/genclntsh.orig Wed Sep 30 07:37:19 1998
--- bin/genclntsh Tue Dec 22 15:36:49 1998
*****
*** 32,38 ****
#
# Explicit path to ensure that we're using the correct commands
#PATH=/usr/bin:/usr/ccs/bin export PATH
! PATH=/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin export PATH
#
# each product MUST provide a $PRODUCT/admin/shrept.lst
--- 32,38 ----
#
# Explicit path to ensure that we're using the correct commands
#PATH=/usr/bin:/usr/ccs/bin export PATH
! #PATH=/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin export PATH
#
# each product MUST provide a $PRODUCT/admin/shrept.lst

```

10.6.5. Az Oracle futtatása

Ha rendesen követtük az iménti utasításokat, akkor most már úgy tudjuk futtatni az **Oracle**-t, mintha csak Linuxon futna.

10.7. Az SAP® R/3® telepítése

Írta: Holger Kipp. Az eredeti verziót SGML-re ültette: Valentino Vaschetto.

Az **SAP** típusú rendszerek telepítéséhez FreeBSD-re hivatalosan nem kaphatunk műszaki segélynyújtást — csak a minősített platformokat támogatják.

10.7.1. Előszó

Ez a leírás az **SAP R/3** rendszer és **Oracle** adatbázis Linux változatainak telepítését mutatja be FreeBSD-n, beleértve a FreeBSD és az **Oracle** telepítését. Kétféle konfigurációt írunk le:

- **SAP R/3 4.6B (IDES)** és **Oracle 8.0.5**, FreeBSD 4.3-STABLE
- **SAP R/3 4.6C** és **Oracle 8.1.7**, FreeBSD 4.5-STABLE

Habár ez a dokumentum igyekszik az összes fontos lépést a lehető legrészletesebb módon tárgyalni, semmiképpen sem célja az **Oracle** és az **SAP R/3** alkalmazásokhoz mellékelt telepítési útmutatók kiváltása.

A kifejezetten az **SAP** vagy az **Oracle** Linux változataira vonatkozó kérdések, valamint az **Oracle** és az **SAP OSS** konkrét használatával kapcsolatos leírások tekintetében a saját dokumentációjukat olvassuk el.

10.7.2. A szoftver

Az **SAP** telepítéséhez az alábbi CD-ket használtuk fel:

10.7.2.1. SAP R/3 4.6B, Oracle 8.0.5

Név	Szám	Leírás
KERNEL	51009113	SAP Kernel Oracle / telepítő / AIX, Linux, Solaris
RDBMS	51007558	Oracle / RDBMS 8.0.5.X / Linux
EXPORT1	51010208	IDES / DB-Export / 1. lemez
EXPORT2	51010209	IDES / DB-Export / 2. lemez
EXPORT3	51010210	IDES / DB-Export / 3. lemez
EXPORT4	51010211	IDES / DB-Export / 4. lemez
EXPORT5	51010212	IDES / DB-Export / 5. lemez
EXPORT6	51010213	IDES / DB-Export / 6. (utolsó) lemez

Emellett még használtuk az **Oracle 8 Server** (az előzetes 8.0.5 változat a Linux 2.0.33 verziójához) CD-jét is, amely igazából nem feltétlenül szükséges, valamint a FreeBSD (a 4.3 RELEASE kiadása után nem sokkal levő) 4.3-STABLE változatát.

10.7.2.2. SAP R/3 4.6C SR2, Oracle 8.1.7

Név	Szám	Leírás
KERNEL	51014004	SAP Kernel Oracle / SAP Kernel 4.6D változat / DEC, Linux
RDBMS	51012930	Oracle 8.1.7/ RDBMS / Linux
EXPORT1	51013953	4.6C kiadás SR2 / Export / 1. lemez
EXPORT1	51013953	4.6C kiadás SR2 / Export / 2. lemez
EXPORT1	51013953	4.6C kiadás SR2 / Export / 3. lemez
EXPORT1	51013953	4.6C kiadás SR2 / Export / 4. (utolsó) lemez
LANG1	51013954	4.6C kiadás SR2 / Nyelvi támogatás / német, angol, francia / 1. lemez

A telepítendő nyelvtől függően egyéb nyelvi támogatást tartalmazó CD használata is szükségessé válhat. Itt most csak a német és angol nyelveket használjuk, ezért elegendő az első CD. Csendben hozzátesszük, hogy mind a négy EXPORT CD száma megegyezik. Ugyanígy a három nyelvi CD-nek is megegyeznek a számai (ez eltér a 4.6B IDES kiadás CD számozásától). Az írás pillanatában a FreeBSD 4.5-STABLE (2002.03.20-i) változatát használjuk.

10.7.3. SAP füzetek

Az **SAP R/3** telepítésével kapcsolatban az alábbi füzetek bizonyultak hasznosnak:

10.7.3.1. SAP R/3 4.6B, Oracle 8.0.5

Szám	Cím
0171356	SAP Software on Linux: Essential Comments
0201147	INST: 4.6C R/3 Inst. on UNIX - Oracle
0373203	Update / Migration Oracle 8.0.5 --> 8.0.6/8.1.6 LINUX
0072984	Release of Digital UNIX 4.0B for Oracle
0130581	R3SETUP step DIPGNTAB terminates
0144978	Your system has not been installed correctly
0162266	Questions and tips for R3SETUP on Windows NT / W2K

10.7.3.2. SAP R/3 4.6C, Oracle 8.1.7

Szám	Cím
0015023	Initializing table TCPDB (RSXP0004) (EBCDIC)
0045619	R/3 with several languages or typefaces
0171356	SAP Software on Linux: Essential Comments
0195603	RedHat 6.1 Enterprise version: Known problems
0212876	The new archiving tool SAPCAR
0300900	Linux: Released DELL Hardware
0377187	RedHat 6.2: important remarks
0387074	INST: R/3 4.6C SR2 Installation on UNIX
0387077	INST: R/3 4.6C SR2 Inst. on UNIX - Oracle
0387078	SAP Software on UNIX: OS Dependencies 4.6C SR2

10.7.4. Hardverkövetelmények

Az alábbi hardvereszközök szükségesek az **SAP R/3** rendszer telepítéséhez. Az éles használathoz ennél természetesen valamivel több kell majd:

Változat	4.6B	4.6C
Processzor	Két Pentium III 800MHz	Két Pentium III 800MHz
Memória	1GB ECC	2GB ECC
Szabad hely a merevlemezen	50 - 60GB (IDES)	50 - 60GB (IDES)

Éles használatra nagyobb gyorsítótárral rendelkező Xeon processzorokat, nagysebességű háttértárakat (SCSI,

hardveres RAID vezérlővel), USV és ECC memória modulok ajánlottak. A nagy tárigényt egyébként az előre beállított IDEs rendszer indokolja, ami egy 27 GB méretű adatbázist hoz létre a telepítés során. Ez a terület általában elegendő egy frissen induló rendszer és hozzá tartozó alkalmazásadatok tárolására.

10.7.4.1. SAP R/3 4.6B, Oracle 8.0.5

A következő hardverkonfigurációt használtuk: két 800 MHz-es Pentium III processzor és a hozzájuk tartozó alaplap, egy Adaptec® 29160 Ultra160 SCSI-vezérlő (a 40/80 GB méretű DLT szalagos meghajtó és CD-meghajtó használatához) és egy Mylex® AcceleRAID™ RAID-vezérlő (2 csatorna, 6.00-1-00 verziójú firmware és 32 MB memória), amihez két 17 GB-os (tükrözött) merevlemez és négy 36 GB-os merevlemez (RAID 5) csatlakozik.

10.7.4.2. SAP R/3 4.6C, Oracle 8.1.7

Itt a hardver egy Dell™ PowerEdge™ 2500 volt: kétprocesszoros alaplap, két darab 1000 MHz-es Pentium III processzorral (fejenként 256 KB gyorsítótárral), 2 GB PC133-as ECC SDRAM memóriával, PERC/3 DC PCI RAID-vezérlővel (128 MB memória), valamint egy EIDE DVD-meghajtóval. A RAID-vezérlőre két, egyenként 18 GB méretű merevlemezt (tükrözve) és négy 36 GB méretű merevlemezt csatlakoztattunk (RAID 5-ben).

10.7.5. A FreeBSD telepítése

Először is telepítenünk kell a FreeBSD-t. Ez több módon is lehetséges, ezekről a 2.13 Szakaszban olvashatunk bővebben.

10.7.5.1. A lemezek felosztása

Az egyszerűség kedvéért az **SAP R/3 4.6B** és **SAP R/3 4.6C SR2** telepítése során is ugyanazt a felosztást használtuk. Egyedül az eszközök nevei változtak, mivel a telepítés eltérő hardvereken történt (/dev/da) és /dev/amr, tehát ha az AMI MegaRAID® esetén a /dev/da0s1a helyett a /dev/amr0s1a eszközt láthatjuk):

Állományrendszer	Méret	Csatlakozási pont
/dev/da0s1a	1 GB	/
/dev/da0s1b	6 GB	lapozóállomány
/dev/da0s1e	2 GB	/var
/dev/da0s1f	8 GB	/usr
/dev/da1s1e	45 GB	/compat/linux/oracle
/dev/da1s1f	2 GB	/compat/linux/sapmnt
/dev/da1s1g	2 GB	/compat/linux/usr/sap

Előre állítsuk be és inicializáljuk a két logikai meghajtót a Mylex és a PERC/3 RAID-vezérlőkön. A hozzá tartozó szoftver a BIOS indításának fázisában hívható be.

A lemezek felosztása némileg eltér az SAP által javasoltaktól, mivel az SAP szerint az **Oracle** könyvtárait (néhány másikkal együtt) külön-külön érdemes csatlakoztatni — mi most az egyszerűsítés kedvéért csak létrehoztuk ezeket.

10.7.5.2. A `make world` és egy új rendszermag

Töltsük le a legfrissebb -STABLE forrásokat. Fordítsuk újra az összes forrást (`make world`) és a beállításainak elvégzése után a saját rendszermagunkat is. Itt ne felejtjük el megadni az **SAP R/3** és az **Oracle** működéséhez szükséges paramétereket.

10.7.6. A Linux környezet telepítése

10.7.6.1. Az linuxos alaprendszer telepítése

Elsőként a `linux_base` portot kell felraknunk (`root` felhasználóként):

```
# cd /usr/ports/emulators/linux_base-fc4
# make install distclean
```

10.7.6.2. A linuxos fejlesztői környezet telepítése

Ha az **Oracle**-t FreeBSD-re a 10.6 Szakaszban leírtak szerint akarjuk telepíteni, akkor szükségünk lesz a linuxos fejlesztőeszközökre is:

```
# cd /usr/ports/devel/linux_devtools
# make install distclean
```

A linuxos fejlesztőkörnyezetet csak az **SAP R/3 46B IDES** telepítésénél raktuk fel. Nincs rá szükségünk, ha a FreeBSD rendszeren nem akarjuk újralinkelni az **Oracle** adatbázist. Pontosan ez a helyzet, amikor egy Linux rendszerhez gyártott **Oracle** készletet használunk.

10.7.6.3. A szükséges RPM csomagok telepítése

Az `R3SETUP` elindításához PAM támogatásra is szükségünk lesz. Amikor először próbáltuk meg telepíteni a FreeBSD 4.3-STABLE változatára az **SAP**-t, felraktuk a PAM-et és az összes hozzá tartozó csomagot, majd végül úgy bírtuk működésre, hogy kényszerítettük a PAM telepítését is. Az **SAP R/3 4.6C SR2** esetén szintén sikerült önmagában felrakni a PAM RPM csomagját is, tehát úgy néz ki, hogy a függőségeit már nem kell telepíteni:

```
# rpm -i --ignoreos --nodeps --root /compat/linux --dbpath /var/lib/rpm \
pam-0.68-7.i386.rpm
```

Az **Oracle 8.0.5** verziójához mellékelt intelligens ügynök futtatásához fel kell rakni a RedHat `tc1-8.0.5-30.i386.rpm` nevű Tcl csomagját is (máskülönben a az **Oracle** telepítése közben szükséges újralinkelés nem fog működni). Vannak ugyan egyébként is gondok az **Oracle** újralinkelésével, azonban ez linuxos probléma, nem pedig FreeBSD-s.

10.7.6.4. Néhány további tipp

Hasznos lehet, ha felvesszük a `linprocfs` bejegyzést az `/etc/fstab` állományba. Ennek pontos részleteit a `linprocfs(5)` man oldalon találjuk meg. Másik fontos paraméter a `kern.fallback_elf_brand=3`, amelyet az `/etc/sysctl.conf` állományba kell beszúrunk.

10.7.7. Az SAP R/3 környezetének létrehozása

10.7.7.1. A szükséges állományrendszerek és csatlakozási pontok létrehozása

Egy egyszerűbb telepítéshez elég csupán a következő állományrendszereket elkészíteni:

csatlakozási pont	méret GB-ban
/compat/linux/oracle	45 GB
/compat/linux/sapmnt	2 GB
/compat/linux/usr/sap	2 GB

Készítenünk kell még néhány linket is, különben az **SAP** telepítője panaszkodni fogni az ellenőrzésük során:

```
# ln -s /compat/linux/oracle /oracle
# ln -s /compat/linux/sapmnt /sapmnt
# ln -s /compat/linux/usr/sap /usr/sap
```

Az egyik ilyen telepítés közben megjelenő hibaüzenet (a *PRD* rendszer és az **SAP R/3 4.6C SR2** telepítése esetén):

```
INFO 2002-03-19 16:45:36 R3LINKS_IND_IND SyLinkCreate:200
    Checking existence of symbolic link /usr/sap/PRD/SYS/exe/dbg to
    /sapmnt/PRD/exe. Creating if it does not exist...

WARNING 2002-03-19 16:45:36 R3LINKS_IND_IND SyLinkCreate:400
    Link /usr/sap/PRD/SYS/exe/dbg exists but it points to file
    /compat/linux/sapmnt/PRD/exe instead of /sapmnt/PRD/exe. The
    program cannot go on as long as this link exists at this
    location. Move the link to another location.

ERROR 2002-03-19 16:45:36 R3LINKS_IND_IND Ins_SetupLinks:0
    can not setup link '/usr/sap/PRD/SYS/exe/dbg' with content
    '/sapmnt/PRD/exe'
```

10.7.7.2. A felhasználók és könyvtárak létrehozása

Az **SAP R/3** rendszernek két felhasználóra és három csoportra van szüksége. Az igényelt felhasználók nevei az **SAP** rendszer azonosítójától (System ID, SID) függenek, amely három betűből áll. Egyes ilyen rendszerazonosítók az **SAP** számára vannak fenntartva. (Például a *SAP* és a *NIX*. Ezek teljes listáját az **SAP** dokumentációjában találjuk meg.) Erre az *IDES* telepítéséhez az *IDS*, a 4.6C SR2 telepítésénél a *PRD* neveket adtuk, mivel ezeket a rendszereket éles használatra szánták. Ennélfogva a következő csoportokat hoztuk létre hozzájuk (a csoportok azonosítói ugyan eltérhetnek az általunk használtaktól):

csoport azonosítója	csoport neve	leírás
100	dba	Adatbázis adminisztrátor
101	sapsys	SAP rendszer
102	oper	Adatbázis operátor

Az **Oracle** alapértelmezett telepítésénél csak a *dba* csoport jön létre. A *dba* csoportot *oper* csoportként is használhatjuk (bővebb információkért lásd az **Oracle** és az **SAP** dokumentációját).

Ezenkívül az alábbi felhasználókra van még szükségünk:

felhasználói azonosító	felhasználói név	általános név	csoport	egyéb csoportok	leírás
1000	idsadm/prdadm	<i>sidadm</i>	sapsys	oper	SAP adminisztrátor
1002	oraid/oraprd	<i>orasid</i>	dba	oper	Oracle adminisztrátor

Az `adduser(8)` parancs használata során a következőkre lesz szükségünk egy “SAP Administrator” létrehozásához (figyeljük a parancsértelmezőt (shell) és a felhasználói könyvtárat (home directory)):

```
Name: sidadm
Password: *****
Fullname: SAP Administrator SID
Uid: 1000
Gid: 101 (sapsys)
Class:
Groups: sapsys dba
HOME: /home/sidadm
Shell: bash (/compat/linux/bin/bash)
```

Ugyanígy az “Oracle Administrator” esetében:

```
Name: orasid
Password: *****
Fullname: Oracle Administrator SID
Uid: 1002
Gid: 100 (dba)
Class:
Groups: dba
HOME: /oracle/sid
Shell: bash (/compat/linux/bin/bash)
```

A `dba` és `oper` csoportok használata során ne felejtsük el megadni az `oper` csoportot sem.

10.7.7.3. Könyvtárak létrehozása

A könyvtárakat általában külön állományrendszerekként hozzák létre, de ez teljesen az igényeinken múlik. Mi most egyszerű könyvtárakként alakítottuk ki ezeket, ezért tulajdonképpen ugyanazon a RAID 5 tömbön találhatóak meg:

Ehhez először beállítjuk az egyes könyvtárak tulajdonosait és engedélyeit (`root` felhasználóként):

```
# chmod 775 /oracle
# chmod 777 /sapmnt
# chown root:dba /oracle
# chown sidadm:sapsys /compat/linux/usr/sap
# chmod 775 /compat/linux/usr/sap
```

Másodsorban *orasid* felhasználóként hozzuk létre az `/oracle/SID` alkönyvtárait:

```
# su - orasid
```

```
# cd /oracle/SID
# mkdir mirrlogA mirrlogB origlogA origlogB
# mkdir sapdata1 sapdata2 sapdata3 sapdata4 sapdata5 sapdata6
# mkdir saparch sapreorg
# exit
```

Az **Oracle 8.1.7** telepítésénél még további könyvtárakra is szükségünk lesz:

```
# su - orasid
# cd /oracle
# mkdir 805_32
# mkdir client stage
# mkdir client/80x_32
# mkdir stage/817_32
# cd /oracle/SID
# mkdir 817_32
```

Megjegyzés: A `client/80x_32` könyvtárnak pontosan ilyen névvel kell rendelkeznie. Ne cseréljük ki a benne szereplő `x`-et semmire se!

A harmadik lépésben létrehozzuk a `sidadm` felhasználóhoz tartozó könyvtárakat:

```
# su - sidadm
# cd /usr/sap
# mkdir SID
# mkdir trans
# exit
```

10.7.7.4. Az `/etc/services`

A **SAP R/3** működéséhez fel kell vennünk néhány olyan bejegyzést is az `/etc/services` állományba, amelyek a FreeBSD telepítése során nem jönnek létre. Így tehát írjuk be az alábbi sorokat (legalább a használni kívánt példány számához illő sorokat adjuk meg — ez jelen esetünkben most a 00. Természetesen az sem okoz gondot, ha a `dp`, `gw`, `sp` és `ms` esetén beírjuk az összes példánynak megfelelő portot 00-tól 99-ig). Amennyiben a **SAProuter** vagy az **SAP OSS** használatára lenne szükségünk, akkor adjuk meg a **SAProuter** által lefoglalt 99-es példánynak megfelelő 3299-es portot a rendszerünkön:

<code>sapdp00</code>	<code>3200/tcp</code>	# SAP menetirányító	3200 + a példány száma
<code>sapgw00</code>	<code>3300/tcp</code>	# SAP átjáró	3300 + a példány száma
<code>sapsp00</code>	<code>3400/tcp</code>	#	3400 + a példány száma
<code>sapms00</code>	<code>3500/tcp</code>	#	3500 + a példány száma
<code>sapmsSID</code>	<code>3600/tcp</code>	# SAP üzenetkezelő szerver	3600 + a példány száma
<code>sapgw00s</code>	<code>4800/tcp</code>	# biztonságos SAP átjáró	4800 + a példány száma

10.7.7.5. A szükséges nyelvi beállítások

Az **SAP**-nek legalább két olyan nyelvre van szüksége, amely nem része az alap RedHat telepítéseknek. Az SAP a saját FTP szervereiről elérhetővé tette az ehhez szükséges RPM csomagokat (amelyek viszont csak OSS típusú hozzáférés birtokában tölthetők le). A 0171356 számú jegyzet tartalmazza a beszerzendő RPM-ek listáját.

Megcsinálhatjuk úgy is, hogy egyszerűen csak linkeket hozunk létre (például a *de_DE* és *en_US* könyvtárakra), habár ezt egy éles rendszer esetében semmiképpen sem ajánljuk (az IDES rendszerrel tapasztalataink szerint eddig még remekül működött). Az alábbi nyelvi beállítások fognak tehát nekünk kelleni:

```
de_DE.ISO-8859-1
en_US.ISO-8859-1
```

Így hozzuk létre hozzájuk a linkeket:

```
# cd /compat/linux/usr/share/locale
# ln -s de_DE de_DE.ISO-8859-1
# ln -s en_US en_US.ISO-8859-1
```

A telepítés során az iméntiek hiánya gondokat okozhat. Ha folyamatosan figyelmen kívül hagyjuk az ezekből fakadó hibákat (vagyis a CENTRDB.R3S állományban a gondot okozó lépések STATUS értékét OK-ra állítjuk), akkor komolyabb erőfeszítések megtétele nélkül majd képtelenek leszünk bejelentkezni a frissen telepített **SAP** rendszerünkbe.

10.7.7.6. A rendszermag finomhangolása

Az **SAP R/3** rendszerek temérdek mennyiségű erőforrást igényelnek. Ennek kielégítésére az alábbi paramétereket adjuk hozzá a rendszermag beállításait tartalmazó állományhoz:

```
# Adjunk a memórizabálókknak (SAP és Oracle):
options MAXDSIZ="(1024*1024*1024)"
options DFLDSIZ="(1024*1024*1024)"
# Kell néhány System V beállítás is:
options SYSVSHM # SYSV típusú osztott memória be
options SHMMAXPGS=262144 # a megosztható memória maximális mérete lapokban
#options SHMMAXPGS=393216 # a 46C telepítésekor ezt használjuk
options SHMMNI=256 # az osztott memóriákhoz tartozó azonosítók maximális száma
options SHMSEG=100 # a futó programonként megosztható szegmensek maximuma
options SYSVMSG # SYSV típusú üzenetsorok
options MSGSEG=32767 # a rendszerben keringő üzenetszegmensek maximális száma
options MSGSSZ=32 # az üzenetszegmensek mérete. 2 hatványa LEGYEN
options MSGMNB=65535 # maximális karakter üzenetsoronként
options MSGTQL=2046 # a rendszerben levő üzenetek maximuma
options SYSVSEM # SYSV típusú szemaforok
options SEMMNU=256 # a szemaforok UNDO struktúráinak száma
options SEMMNS=1024 # a rendszerben levő szemaforok száma
options SEMMNI=520 # a szemaforok azonosítóinak mennyisége
options SEMUME=100 # az UNDO kulcsok száma
```

Az itt megadott minimum értékek az SAP által kiadott dokumentációkból származnak. Mivel a Linux változathoz erről nincs külön leírás, ezért a (32 bites) HP-UX változat dokumentációi között érdemes ennek utánanézni. Mivel a 4.6C SR2 telepítéséhez használt rendszeren valamivel több fizikai memória állt rendelkezésünkre, ezért az osztott

szegmensek méretét nagyobbra tudtuk megválasztani mind az **SAP**, mind az **Oracle** esetében, ami magyarázza a megosztható lapok nagyobb számát.

Megjegyzés: A FreeBSD i386 változatának telepítése során hagyjuk meg a `MAXDSIZ` és `DFLDSIZ` értékek alapértelmezett 1 GB-os maximumát. Ellenkező esetben ezekhez hasonló furcsa hibaüzeneteket láthatunk:
ORA-27102: out of memory vagy Linux Error: 12: Cannot allocate memory.

10.7.8. Az SAP R/3 telepítése

10.7.8.1. Az SAP CD-k előkészítése

Sok CD-t kell a telepítés során mozgatni, tehát csatlakoztatni és leválasztani. Ha viszont elegendő meghajtóval rendelkezünk, akkor akár csatlakoztathatjuk egyszerre is az összeset. Vagy felmásolhatjuk a CD-k tartalmát a nekik megfelelő könyvtárakba:

```
/oracle/SID/sapreorg/cd-neve
```

ahol a `cd-neve` a következők valamelyike: `KERNEL`, `RDBMS`, `EXPORT1`, `EXPORT2`, `EXPORT3`, `EXPORT4`, `EXPORT5` és `EXPORT6` (4.6B/IDES), valamint `KERNEL`, `RDBMS`, `DISK1`, `DISK2`, `DISK3`, `DISK4` és `LANG` (4.6C SR2). A csatlakoztatott CD-ken található állományok neveinek nagybetűseknek kell lenniük. Ha nem így lenne, akkor a csatlakoztatásnál adjuk meg a `-g` opciót. Így tehát a következő parancsokat kell kiadnunk:

```
# mount_cd9660 -g /dev/cd0a /mnt
# cp -R /mnt/* /oracle/SID/sapreorg/cd-neve
# umount /mnt
```

10.7.8.2. A telepítőszkript futtatása

Elsőként egy `install` nevű könyvtárat kell előkészítenünk:

```
# cd /oracle/SID/sapreorg
# mkdir install
# cd install
```

Ezután futtassuk le a telepítőszkriptet, ami pedig bemásolja az `install` könyvtárba szinte az összes fontos állományt:

```
# /oracle/SID/sapreorg/KERNEL/UNIX/INSTTOOL.SH
```

Az IDES (4.6B) változathoz egy teljes SAP R/3 bemutató rendszer is tartozik, ezért a megszokott három CD helyett hat `EXPORT` típusú CD-ből áll. Itt a `CENTRDB.R3S` telepítősablon csak a szabvány központi példányt hozza létre (**R/3** és az adatbázis), az IDES központi példányát már nem. Ezért az `EXPORT1` könyvtárból ki kell másolnunk a `CENTRDB.R3S` állományt, különben az `R3SETUP` csak három `EXPORT` CD-t fog kérni.

Az újabb **SAP 4.6 SR2** kiadáshoz négy `EXPORT` CD tartozik. A telepítés folyamatát a `CENTRAL.R3S` állományban levő paraméterek vezérlik. A korábbi kiadásokkal ellentétben nincsenek külön sablonok az adatbázissal és a nélküle

telepítendő központi példányok számára. Az **SAP** az adatbázisok telepítésére külön sablont használ. Újrakezdéskor a telepítést ettől függetlenül elegendő az eredeti állománnyal újraindítani.

A telepítés közben és után az **SAP**-nek a `hostname` paranccsal csak a gép saját nevét, nem pedig a teljes hálózati nevét kell megadnunk. Ilyenkor ezt vagy egyenként begépeljük, vagy létrehozunk rá egy álnevet az `orasid` és `sidadm` (valamint a megfelelő lépésekben a `root`) felhasználóknak: `alias hostname='hostname -s'`. Ezenkívül még az **SAP** telepítésekor létrehozott mindkét felhasználó `.profile` és `.login` állományait is beállíthatjuk ennek megfelelően.

10.7.8.3. Az R3SETUP 4.6B verziójának indítása

Ne felejtjük el jól beállítani az `LD_LIBRARY_PATH` környezeti változót:

```
# export LD_LIBRARY_PATH=/oracle/IDS/lib:/sapmnt/IDS/exe:/oracle/805_32/lib
```

A telepítés könyvtárában `root` felhasználóként indítsuk el az `R3SETUP` programot:

```
# cd /oracle/IDS/sapreorg/install
# ./R3SETUP -f CENTRDB.R3S
```

A szkript ezek után feltesz néhány kérdést (az alapértelmezett válaszok zárójelben, közvetlenül a megadottak után):

Kérdés	Alapértelmezés	Válasz
Enter SAP System ID	[C11]	IDSEnter
Enter SAP Instance Number	[00]	Enter
Enter SAPMOUNT Directory	[/sapmnt]	Enter
Enter name of SAP central host	[troubadix.domain.de]	Enter
Enter name of SAP db host	[troubadix]	Enter
Select character set	[1] (WE8DEC)	Enter
Enter Oracle server version (1) Oracle 8.0.5, (2) Oracle 8.0.6, (3) Oracle 8.1.5, (4) Oracle 8.1.6		1Enter
Extract Oracle Client archive	[1] (Yes, extract)	Enter
Enter path to KERNEL CD	[/sapcd]	/oracle/IDS/sapreorg/KERNEL
Enter path to RDBMS CD	[/sapcd]	/oracle/IDS/sapreorg/RDBMS
Enter path to EXPORT1 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT1
Directory to copy EXPORT1 CD	[/oracle/IDS/sapreorg/CD4_DIR]	Enter
Enter path to EXPORT2 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT2
Directory to copy EXPORT2 CD	[/oracle/IDS/sapreorg/CD5_DIR]	Enter
Enter path to EXPORT3 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT3
Directory to copy EXPORT3 CD	[/oracle/IDS/sapreorg/CD6_DIR]	Enter
Enter path to EXPORT4 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT4
Directory to copy EXPORT4 CD	[/oracle/IDS/sapreorg/CD7_DIR]	Enter
Enter path to EXPORT5 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT5
Directory to copy EXPORT5 CD	[/oracle/IDS/sapreorg/CD8_DIR]	Enter
Enter path to EXPORT6 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT6

Kérdés	Alapértelmezés	Válasz
Directory to copy EXPORT6 CD	[/oracle/IDS/sapreorg/CD9_DIR]	Enter
Enter amount of RAM for SAP + DB		850 Enter (megabyte)
Service Entry Message Server	[3600]	Enter
Enter Group-ID of sapsys	[101]	Enter
Enter Group-ID of oper	[102]	Enter
Enter Group-ID of dba	[100]	Enter
Enter User-ID of <i>sidadm</i>	[1000]	Enter
Enter User-ID of <i>orasid</i>	[1002]	Enter
Number of parallel procs	[2]	Enter

Ha a CD-ket nem különböző helyekre másoltuk, akkor az **SAP** telepítője nem fogja megtalálni ezeket (a rajtuk levő LABEL.ASC segít neki az azonosításban) és kérni fogja a CD csatlakoztatását, illetve a csatlakozási pontjának megadását.

A CENTRDB.R3S sem minden esetben mentes a hibáktól. A tapasztalataink szerint az EXPORT4 címkéjű CD-t kérte újra, miközben a helyes kulcsokat jelezte ki (6_LOCATION, majd 7_LOCATION stb.), így egyszerűen csak lépünk tovább az értékek meghagyásával.

Függetlenül az imént említett problémáktól, egészen az Oracle adatbáziskezelő telepítéséig mindennek működnie kellene.

10.7.8.4. Az R3SETUP 4.6C SR2 elindítása

Állítsuk be jól az LD_LIBRARY_PATH környezeti változó értékét. Ez némileg eltér a 4.6B és az **Oracle 8.0.5** párosának beállításától:

```
# export LD_LIBRARY_PATH=/sapmnt/PRD/exe:/oracle/PRD/817_32/lib
```

A telepítés könyvtárából `root` felhasználóként indítsuk el az R3SETUP programot:

```
# cd /oracle/PRD/sapreorg/install
# ./R3SETUP -f CENTRAL.R3S
```

A szkript ezek után feltesz néhány kérdést (az alapértelmezett válaszok zárójelben, közvetlenül a megadottak után):

Kérdés	Alapértelmezés	Válasz
Enter SAP System ID	[C11]	PRD Enter
Enter SAP Instance Number	[00]	Enter
Enter SAPMOUNT Directory	[/sapmnt]	Enter
Enter name of SAP central host	[majestix]	Enter
Enter Database System ID	[PRD]	PRD Enter
Enter name of SAP db host	[majestix]	Enter
Select character set	[1] (WE8DEC)	Enter
Enter Oracle server version (2)		2 Enter
Oracle 8.1.7		

Kérdés	Alapértelmezés	Válasz
Extract Oracle Client archive	[1] (Yes, extract)	Enter
Enter path to KERNEL CD	[/sapcd]	/oracle/PRD/sapreorg/KERNEL
Enter amount of RAM for SAP + DB	2044	1800 Enter (megabyte)
Service Entry Message Server	[3600]	Enter
Enter Group-ID of sapsys	[100]	Enter
Enter Group-ID of oper	[101]	Enter
Enter Group-ID of dba	[102]	Enter
Enter User-ID of oraprd	[1002]	Enter
Enter User-ID of prdadm	[1000]	Enter
LDAP support		3 Enter (nincs támogatás)
Installation step completed	[1] (continue)	Enter
Choose installation service	[1] (DB inst,file)	Enter

Az OSUSERDBSID_IND_ORA és OSUSERIDADM_IND_ORA lépésekben az *orasid* és *sidadm* felhasználók létrehozása hibákra futhat.

Függetlenül az említett problémáktól, az Oracle adatbáziskezelő telepítéséig mindennek remekül kell működnie.

10.7.9. Az Oracle 8.0.5 telepítése

Az **Oracle** Linux változatának telepítése során felmerülő problémák tekintetében keressük fel az SAP füzeteket és az Oracle Readme állományait. A legtöbb, ha nem is az összes gondot az egymással nem kompatibilis függvénykönyvtárak okozzák.

Az **Oracle** telepítésének részleteit a Az Oracle telepítése című szakaszban találjuk.

10.7.9.1. Az Oracle 8.0.5 telepítése az *oraInst* segítségével

Az **Oracle** 8.0.5 verziójának használata esetén néhány további függvénykönyvtár újralinkelésére is szükség lesz, mivel az **Oracle 8.0.5** még a régi glibc könyvtárral lett fordítva (RedHat 6.0), viszont a RedHat 6.1 már a glibc újabb verzióját használja. A linkelés működéséhez az alábbi csomagokat kell még telepítenünk:

- `compat-libs-5.2-2.i386.rpm`
- `compat-glibc-5.2-2.0.7.2.i386.rpm`
- `compat-egcs-5.2-1.0.3a.1.i386.rpm`
- `compat-egcs-c++-5.2-1.0.3a.1.i386.rpm`
- `compat-binutils-5.2-2.9.1.0.23.1.i386.rpm`

A részleteket lásd az SAP füzeteiben vagy az Oracle Readme állományaiban. Amennyiben ez nem oldható meg, akkor az eredeti binárisok, esetleg az eredeti RedHat rendszerből származó újralinkelt binárisok is használhatóak (habár a telepítés pillanatában személyesen ezt nem tudtuk ellenőrizni).

Az intelligens ügynök lefordításához fel kell raknunk a RedHat saját Tcl csomagját. Ha ehhez nem tudjuk beszerezni a `tcl-8.0.3-20.i386.rpm` csomagot, akkor a RedHat 6.1 változatához készült `tcl-8.0.5-30.i386.rpm` is megteszi.

Az újralinkeléstől eltekintve a telepítés többi része szinte adja magát:

```
# su - oraids
# export TERM=xterm
# export ORACLE_TERM=xterm
# export ORACLE_HOME=/oracle/IDS
# cd $ORACLE_HOME/orainst_sap
# ./orainst
```

Az *Oracle On-Line Text Viewer* kikapcsolásán (mivel az jelenleg Linux alatt sem érhető el) kívül mindegyik képernyőt hagyjuk jóvá az **Enter** billentyű lenyomásával. Az **Oracle** ezután a rendelkezésre álló `gcc`, egcs vagy `i386-redhat-linux-gcc` helyett a `i386-glibc20-linux-gcc` használatával újra akarja linkelni magát.

Idő hiányában az **Oracle 8.0.5 PreProduction** kiadásából emeltünk ki binárisokat, de az adatbáziskezelő rendszer felélesztésére tett első kísérleteink kudarcba fulladtak, és ezután a megfelelő RPM-ek összeszedése valódi rémálomnak bizonyult.

10.7.9.2. Az Oracle 8.0.5 Pre-production Release for Linux (Kernel 2.0.33) telepítése

A telepítés nagyon könnyű. Csatlakoztassuk a CD-t, majd indítsuk el a telepítőt. Ezután meg kell adnunk az Oracle felhasználói könyvtárát és a telepítő odamásolja az összes binárist. Habár a telepítés megkezdése előtt a korábbi kísérleteink nyomát nem tüntettük el.

Ezt követően az **Oracle** adatbázisrendszer minden további gond nélkül elindítható.

10.7.10. Az Oracle 8.1.7 Linux változatának telepítése

Szedjük le az `oracle8172.tgz` állományt a Linux rendszeren létrehozott könyvtárából, és bontsuk ki a `/oracle/SID/817_32/` könyvtárba.

10.7.11. Az SAP R/3 telepítésének folytatása

Először is ellenőrizzük az `isamd` (`sidadm`) és `oraids` (`orasid`) felhasználók környezeti beállításait. A `.profile`, `.login` és `.cshrc` állományaikban a korábbi beállítások szerint kell szerepelnie a `hostname` parancsnak. Ha még mindig a teljes hálózati név lenne meg bennük, akkor a `hostname` parancsot át kell írni mind a három állományban a `hostname -s` parancsra.

10.7.11.1. Az adatbázis feltöltése

Ezután az `R3SETUP` folytatható vagy újraindítható (attól függően, hogy a kilépést választottuk-e vagy sem). Az `R3SETUP` ekkor létrehozza az adatbázisban a táblákat és az `R3load` meghívásával feltölti ezeket adatokkal (a 46B IDES változat esetében az `EXPORT1` - `EXPORT6`, a 46C esetében pedig a `DISK1` - `DISK4` lemezekről).

Amikor a feltöltés befejeződött (ami akár órákig is eltarthat), szükség lesz még néhány jelszó megadására is. A próbatelepítéseknél nyugodtan használhatjuk a jól ismert alapértelmezett jelszavakat (azonban mindenképpen változtassuk meg ezeket, ha egy kicsit is számít a biztonság!):

Kérdés	Válasz
Enter Password for sapr3	sap Enter
Confirum Password for sapr3	sap Enter
Enter Password for sys	change_on_install Enter
Confirm Password for sys	change_on_install Enter
Enter Password for system	manager Enter
Confirm Password for system	manager Enter

A 4.6B telepítése során még gondjaink akadtak a `diagntab` használatával.

10.7.11.2. Az Oracle Listener elindítása

Így kell elindítani az `orasid` felhasználóval az **Oracle** Listenert:

```
% umask 0; lsnrctl start
```

Ha máshogy próbálkozunk, akkor az ORA-12546 kódú hibát fogjuk kapni, mert a hálózati portok socketei nem rendelkeznek a szükséges engedélyekkel. Lásd a 072984-es SAP füzet.

10.7.11.3. Az MNLS táblák frissítése

Ha nem Latin 1 kódolású nyelveket akarunk importálni az **SAP** rendszerbe, akkor frissítenünk kell a többnyelvű nyelvi támogatáshoz (Multi National Language Support, MNLS) tartozó táblázatokat. Ezek bemutatását a 15023 és 45619 számú SAP OSS füzetekben olvashatjuk. Minden más esetben az **SAP** telepítésekor nyugodtan kihagyhatjuk.

Megjegyzés: Ha még nincs is konkrétan szükségünk az MNLS-re, akkor is ellenőriznünk és inicializálnunk kell a TCPDB táblát. A 0015023 és 0045619 számú SAP füzetekben tudhatunk meg erről többet.

10.7.12. Telepítés utáni teendők

10.7.12.1. Az SAP R/3 licenckulcsának megszerzése

Az **SAP R/3** licenckulcsát külön kell kérni. Fontos, mert a telepítéshez használatos ideiglenes licenc csak négy hétig érvényes. Először szerezzük meg a hardverkulcsot. Jelentkezzünk be az `idsadm` felhasználóval és adjuk ki a `saplicense` parancsot:

```
# /sapmnt/IDS/exe/saplicense -get
```

A `saplicense` paraméter nélkül meghívására válaszul opciókat listáz ki. A licenckulcsot megérkezése után így tudjuk élesíteni:

```
# /sapmnt/IDS/exe/saplicense -install
```

Ezután a következő értékeket kell megadni:

```
SAP SYSTEM ID      = SID, 3 karakter
CUSTOMER KEY       = hardverkulcs, 11 karakter
INSTALLATION NO    = telepítés száma, 10 számjegy
EXPIRATION DATE    = ééééhhnn, tehát "99991231"
LICENSE KEY        = licenckulcs, 24 karakter
```

10.7.12.2. A felhasználók létrehozása

Hozzunk létre egy felhasználót a 000 kliensen belül (a csak rajta belül elvégezhető feladatokhoz, aki különbözik a sap* és ddic felhasználóktól). Felhasználónévként általában a wartung nevet választottuk (ami angolul a service névnek, avagy szolgáltatásnak felel meg). A sap_new és sap_all nevű profilok is kellenek. A biztonságosság kedvéért a kliens összes alapértelmezett felhasználójának (beleértve a sap* és ddic felhasználókat is) változtassuk meg a jelszavát.

10.7.12.3. A szállítási rendszer, a profilok, működési módok stb. beállítása

A ddic és sap* felhasználóktól eltérő nevű felhasználóval a 000 kliensen belül legalább a következőket végezzük el:

Feladat	Tranzakció
A szállítási rendszer (Transport System) beállítása, például a <i>Stand-Alone Transport Domain Entity</i> értékre	STMS
A rendszer profiljának létrehozása és szerkesztése	RZ10
A működési módok és példányok karbantartása	RZ04

Az iménti és az összes többi telepítés utáni lépések leírása teljes egészében megtalálható az **SAP** telepítési útmutatóiban.

10.7.12.4. Az `initSID.sap` (`initIDS.sap`) szerkesztése

Az `/oracle/IDS/dbs/initIDS.sap` állomány tartalmazza a **SAP** tartalék profilját. Itt többek közt a használni kívánt szalag méretét, a tömörítés típusát és hasonló paramétereket kell definiálni. A `sapdba / brbackup` futtatásához a következő értékeket változtattuk meg:

```
compress = hardware
archive_function = copy_delete_save
cpio_flags = "-ov --format=newc --block-size=128 --quiet"
cpio_in_flags = "-iuv --block-size=128 --quiet"
tape_size = 38000M
tape_address = /dev/NSA0
tape_address_rew = /dev/sa0
```

Magyarázat:

`compress` (tömörítés): HP DLT1 típusú szalagot használtunk, ami tud hardveres tömörítést.

`archive_function` (archiválási házirend): Ez adja meg, hogy alapértelmezés szerint mi történjen az Oracle archivált naplóival: az új naplóállományok először a szalagra mentődnek, majd a már lementett naplók ismét mentésre kerülnek és végül törölődnek. Ezzel sok fejfájástól menekülünk meg, mivel ilyenkor az archiváló szalagok esetleges sérülése esetén is valószínűleg képesek leszünk visszaállítani az adatbázist.

`cpio_flags` (a `cpio` beállítása): A `-B` használata alapértelmezés, amivel a blokkok mérete 5120 byte-ra állítódik. A DLT típusú szalagokhoz a HP legalább 32 KB-os blokkméretet javasolt, ezért a `--block-size=128` beállítással ezt 64 KB-ra növeltük. Szükségünk volt a `--format=newc` beállításra is, mivel 65535-nél több inode számunk van. Az utolsó beállítás a `--quiet`, amivel megakadályozzuk, hogy a `cpio` lementett blokkokat összefoglaló kijelzésére begerjedjen a `brbackup`.

`cpio_in_flags` (a `cpio` bemeneti beállításai): A szalagok visszatöltésénél használt beállítások. A formátumot automatikusan felismeri.

`tape_size` (szalagméret): Ezzel adjuk meg általában a szalag nyers kapacitását. Biztonsági okokból (hardveres tömörítést használunk) ez az érték a ténylegesnél valamivel kisebb.

`tape_address` (szalagos eszköz): a `cpio` által használható nem visszatekerhető eszköz.

`tape_address_rew` (visszatekerhető szalagos eszköz): A `cpio` által használható visszatekerhető eszköz.

10.7.12.5. Telepítés utáni beállítások

Az **SAP** alábbi paramétereit kell beállítani a telepítés után (IDES 46B, 1 GB memóriával):

Név	Érték
ztta/roll_extension	250000000
abap/heap_area_dia	300000000
abap/heap_area_nondia	400000000
em/initial_size_MB	256
em/blocksize_kB	1024
ipc/shm_psize_40	70000000

0013026 SAP füzet:

Név	Érték
ztta/dynpro_area	2500000

0157246 SAP füzet:

Név	Érték
rdisp/ROLL_MAXFS	16000
rdisp/PG_MAXFS	30000

Megjegyzés: A fenti paraméterek használatával egy 1 gigabyte fizikai memóriával rendelkező rendszer esetén nagyjából így alakul a memóriahasználat:

Mem: 547M Active, 305M Inact, 109M Wired, 40M Cache, 112M Buf, 3492K Free

(547 MB aktív, 305 MB inaktív, 109 MB rögzített, 40 MB gyorsítótár, 112 MB puffer, 3492 KB szabad)

10.7.13. A telepítés során adódó problémák

10.7.13.1. Az R3SETUP újraindítása egy probléma kijavítása után

Az R3SETUP hiba esetén leáll. Miután átnéztük a hibára utaló naplókat és elhárítottuk a hiba okát, újra el kell indítanunk az R3SETUP programot, majd a REPEAT opció kiválasztásával próbáljuk megismételni az R3SETUP által kifogásolt legutóbbi műveletet.

Az R3SETUP újraindításához egyszerűen adjuk meg a megfelelő R3S állományt:

```
# ./R3SETUP -f CENTRDB.R3S
```

a 4.6B verzió esetén, vagy a

```
# ./R3SETUP -f CENTRAL.R3S
```

a 4.6C verzió esetén, függetlenül attól, hogy a hiba a CENTRAL.R3S vagy DATABASE.R3S állományoknál keletkezett.

Megjegyzés: Egyes lépéseknél az R3SETUP úgy véli, hogy az **SAP** programjai működnek (mivel a hozzájuk tartozó lépéseket már megtettük), így a hibák miatt az adatbázist esetleg korábban nem tudta elindítani. Ezért a hibák kijavításának befejeztével az R3SETUP ismételt indítása előtt nekünk kell beindítani mind az adatbázist, mind pedig az **SAP** rendszert.

Ne felejtjük el újra elindítani az **Oracle** Listener segédprogramját sem (az *orasid* felhasználóval adjuk ki a `umask 0; lsnrctl start` parancsot), ha az időközben leállt volna (például a rendszer kényszerű újraindítása miatt).

10.7.13.2. OSUSERSIDADM_IND_ORA az R3SETUP közben

Ha az R3SETUP panaszkodik ebben a lépésben, akkor írjuk át az általa ekkor használt sablont (a 4.6B esetén ez a CENTRDB.R3S, illetve a 4.6C esetén ez a CENTRAL.R3S vagy a DATABASE.R3S). Keressük a [OSUSERSIDADM_IND_ORA] szöveget, vagy csak a STATUS=ERROR bejegyzést, majd írjuk be a következő értékeket:

```
HOME=/home/sidadm (üres volt)
STATUS=OK (ERROR státusza volt)
```

Ezután indítsuk újra az R3SETUP programot.

10.7.13.3. OSUSERDBSID_IND_ORA az R3SETUP közben

Az R3SETUP ebben a lépésben is hajlamos panaszkodni. Az itt felbukkanó hiba hasonló az OSUSERSIDADM_IND_ORA lépésben jelentkezőhöz. Szerkesszük át az R3SETUP által ilyenkor használt sablont (4.6B verzió esetén ez a CENTRDB.R3S, illetve 4.6C verziónál a CENTRAL.R3S vagy DATABASE.R3S). Keressük

meg a [OSUSERDBSID_IND_ORA] részt, vagy csak a STATUS=ERROR bejegyzést, majd írjuk át az ebben a szakaszban szereplő értéket így:

```
STATUS=OK
```

Indítsuk újra az R3SETUP programot.

10.7.13.4. oraview.vrf FILE NOT FOUND hiba az Oracle telepítése közben

A telepítés megkezdése előtt nem tiltottuk le az *Oracle On-Line Text Viewer* felrakását. Habár Linux esetén ez nem használható, alapértelmezés szerint mégis ki van választva. Az Oracle telepítő menüjében tiltsuk le ezt és nélküle kezdjük újra a telepítést.

10.7.13.5. TEXTENV_INVALID hiba az R3SETUP, RFC vagy SAPgui Start programokban

Ha ilyen hibával kerülünk szembe, akkor hiányoznak a megfelelő nyelvi állományok. A 0171356 SAP füzet tartalmazza a telepítendő RPM csomagok felsorolását (például a RedHat 6.1 esetén a saplocales-1.0-3 és saposcheck-1.0-1). Amennyiben figyelmen kívül hagyjuk az ilyen hibákat, és az R3SETUP minden kiakadásánál átírjuk (a CENTRDB.R3S állományban) az STATUS értékét az ERROR értékről az OK értékre és újraindítjuk, az **SAP** nem állítódik be jól és nem tudunk a **SAPgui** alkalmazással rácsatlakozni a frissen telepített rendszerre még akkor sem, ha el tudtuk indítani. Amikor a régebbi linuxos **SAPgui** alkalmazással csatlakozunk, a következő üzeneteket kapjuk:

```
Sat May 5 14:23:14 2001
*** ERROR => no valid userarea given [trgmsg0. 0401]
Sat May 5 14:23:22 2001
*** ERROR => ERROR NR 24 occurred [trgmsg1. 0410]
*** ERROR => Error when generating text environment. [trgmsg1. 0435]
*** ERROR => function failed [trgmsg1. 0447]
*** ERROR => no socket operation allowed [trxio.c 3363]
Speicherzugriffsfehler
```

Ez a viselkedés annak köszönhető, hogy az **SAP R/3** nem képes jól összerendelni a nyelvi beállításokat, sőt, magát sem képes jól beállítani (hiányoznak némely bejegyzések az adatbázis egyes tábláiban). Az **SAP**-hez úgy tudunk ilyenkor csatlakozni, ha a DEFAULT.PFL állományba felvesszük a következő bejegyzéseket (lásd 0043288 füzet):

```
abap/set_etct_env_at_new_mode = 0
install/collate/active = 0
rscp/TCP0B = TCP0B
```

Majd indítsuk újra az egész **SAP** rendszert. Ezután már tudunk csatlakozni hozzá, még ha az országra jellemző nyelvi beállítások nem is működnek tökéletesen. Miután korrigáltuk az ország beállításait (és felraktuk a megfelelő nyelvi állományokat), távolítsuk el az iménti bejegyzéseket a DEFAULT.PFL állományból és indítsuk újra az **SAP** rendszert.

10.7.13.6. Az ORA-00001 hiba

Ez a hiba FreeBSD alatt az **Oracle 8.1.7** használata során következhet be. Akkor történik, amikor az **Oracle** adatbázis nem volt képes rendesen inicializálni magát és összeomlott, aminek révén szemaforokat és memóriát hagyott megosztva a rendszerben. Így az adatbázis következő indításakor kapunk egy kövér ORA-00001 hibát.

Az `ipcs -a` paranccsal keressük meg ezeket, majd az `ipcrm` segítségével pedig számoljuk fel.

10.7.13.7. Az ORA-00445 (a PMON háttérprogram nem indult el) hiba

Ez a hiba az **Oracle 8.1.7** használatakor következhet be. Akkor kapjuk ezt a hibát, amikor `prdadm` felhasználóként a elindítjuk `startsap` szkriptet (például `startsap_majestix_00`).

Erre gyógyír lehet, ha ehelyette az adatbázis elindításához az `oraprd` felhasználóval adjuk ki az `svrmgrl` parancsot:

```
% svrmgrl
SVRMGR> connect internal;
SVRMGR> startup;
SVRMGR> exit
```

10.7.13.8. Az ORA-12546 (A Listener indítása megfelelő engedélyekkel) hiba

Az **Oracle** Listener alkalmazását `oraids` felhasználóként az alábbi paranccsal indítsuk el:

```
# umask 0; lsnrctl start
```

Máskülönben ORA-12546 hibát kapunk, mivel a hálózati portokhoz tartozó socketek nem rendelkeznek a megfelelő engedélyekkel. Lásd 0072984 SAP füzet.

10.7.13.9. Az ORA-27102 (Nincs elég memória) hiba

Akkor fordul elő ilyen hiba, amikor a `MAXDSIZ` és `DFLDSIZ` értékeit 1 GB-nál (1024 x 1024 x 1024-nél) nagyobbra állítottuk. Mellé még kapunk egy `Linux Error 12: Cannot allocate memory` hibát is.

10.7.13.10. [DIPGNTAB_IND_IND] az R3SETUP közben

Erről alapvetően a 0130581 számú SAP füzet ad tájékoztatást (az `R3SETUP DIPGNTAB` lépése hibára fut). Az **IDES** telepítése során az **SAP** rendszer valamiért az “IDS” név helyett egy üres karakterláncot használ. Ez a könyvtárak elérésében kisebb gondokat okoz, mivel az elérési útvonaluk a `SID`-ből generálódik (ami ebben az esetben az `IDS`). Tehát a

```
/usr/sap/IDS/SYS/...
/usr/sap/IDS/DVMGS00
```

helyett a következőt próbálja meg elérni:

```
/usr/sap//SYS/...
/usr/sap/D00
```

A telepítés folytatásához létrehoztunk egy linket és egy másik könyvtárat:

```
# pwd
/compat/linux/usr/sap
# ls -l
total 4
drwxr-xr-x 3 idsadm sapsys 512 May 5 11:20 D00
```

```
drwxr-x--x 5  idsadm sapsys 512 May 5 11:35 IDS
lrwxr-xr-x 1  root   sapsys 7 May 5 11:35 SYS -> IDS/SYS
drwxrwxr-x 2  idsadm sapsys 512 May 5 13:00 tmp
drwxrwxr-x 11 idsadm sapsys 512 May 4 14:20 trans
```

Észrevettük, hogy a SAP füzetekben (0029227 és 0008401) ugyanezt a viselkedést írják le. Az **SAP 4.6C** telepítésénél azonban ilyen hibával nem találkoztunk.

10.7.13.11. [RFCRSWBOINI_IND_IND] az R3SETUP közben

Az **SAP 4.6C** telepítése folyamán ez a hiba csupán egy korábban bekövetkezett másik hiba utóhatása volt. Itt át kell néznünk az összes érintett naplót és ki kell javítanunk a tényleges problémát.

Amennyiben a naplók átvizsgálása után csak ezt találjuk egyedüli hibának (lásd SAP füzetek), állítsuk át (a CENTRDB.R3S állományban) a STATUS értékét az OK értékre, majd indítsuk újra az R3SETUP programot. A telepítés befejezése után hajtsuk végre az SE38 tranzakcióból az RSWBOINS riportot. A további RFCRSWBOINI és RFCRADDBDIF lépésekkel kapcsolatban lásd a 0162266 SAP füzetet.

10.7.13.12. [RFCRADDBDIF_IND_IND] az R3SETUP közben

Itt az előbbihez hasonló feltételek élnek: mindenképpen ellenőrizzük a naplókban, hogy a hibát nem egy korábban keletkezett hiba okozta.

Ha tényleg csak az 0162266 SAP füzetben leírtak érvényesek, akkor (a CENTRDB.R3S állományban) állítsuk a gondot okozó lépés STATUS értékét az ERROR értékről az OK értékre, és indítsuk újra az R3SETUP programot. A telepítés után pedig hajtsuk végre az SE38 tranzakcióból az RADDBDIF riportot.

10.7.13.13. A sigaction sig31: File size limit exceeded hiba

Ez a *disp* és *work* **SAP** programok indítása során történhet meg. Az **SAP** rendszert indító *startsap* szkriptről leválva indulnak el a többi **SAP** program elindításáért felelős alfolyamatok. Ennek eredményeképpen a szkript maga nem fogja észrevenni a hibát.

Az **SAP** programok elindulását az `ps ax | grep SID` paranccsal tudjuk ellenőrizni. Az eredményül kapott listában az összes aktív **Oracle** és **SAP** programnak szerepelnie kell. Ha ebből az tűnik ki, hogy bizonyos programok hiányoznak, vagy nem képesek kapcsolódni az **SAP** rendszerhez, akkor az `/usr/sap/SID/DVEBMGSnr/work/` könyvtárban nézzük át a hozzájuk tartozó naplóállományokat. Elsősorban a `dev_ms` és a `dev_disp` állományok fontosak számunkra.

A 31-es jelzés akkor keletkezik, ha az **Oracle** és az **SAP** által használt osztott memória mértéke meghaladja a rendszermag beállításai közt megadott értéket. Ezt tehát ennek növelésével lehet orvosolni:

```
# az éles 46C rendszereknek több kell:
options SHMMAXPGS=393216
# a 46B beéri kevesebb is:
#options SHMMAXPGS=262144
```

10.7.13.14. A `saposc1` nem indul

A `saposc1` (4.6D verzió) programmal akad néhány probléma. Az **SAP** rendszer az `saposc1` segítségével próbál adatokat gyűjteni a rendszer teljesítményéről. Mivel ez a program nem feltétlenül szükséges az **SAP** rendszer működéséhez, ez a probléma nem tekinthető komolynak. A korábbi (4.6B) verziókban ugyan működik, de semmilyen adatot nem képes begyűjteni (mivel a legtöbb hívás, például a processzorhasználat függvénye, egyszerűen csak nullát ad vissza).

10.8. Témák haladóknak

Ha kíváncsiak vagyunk a Linux emuláció működésére, olvassuk el ezt a szakaszt. Az itt leírtak leginkább Terry Lambert (tlambert@primenet.com) FreeBSD chat levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat>) címére írt levele nyomán kerülnek bemutatásra (Az üzenet azonosítója: `<199906020108.SAA07001@usr09.primenet.com>`).

10.8.1. Hogyan működik?

A FreeBSD rendelkezik egy ún. “végrehajtási osztály betöltővel” (execution class loader). Ez lényegében a `execve(2)` rendszerhívás alatt meghívódó absztrakciós réteg.

A FreeBSD-nek a `#!` karaktersorozat hatására parancsértelmezők vagy a hozzájuk tartozó szkriptek betöltésére utasító biztonsági betöltő helyett van egy listája az alkalmas betöltőkről.

A UNIX rendszerek a hagyományok szerint egyetlen betöltővel rendelkeznek, ami először megvizsgálja a betölteni kívánt állomány bűvös számát (ami általában az első 4 vagy 8 byte) és ez alapján eldönti, hogy az adott formátum támogatott-e. Amennyiben ez így van, meghívja a betöltőt.

Ha a bináris típusa nem ismert a rendszer számára, akkor az `execve(2)` hívás hibával tér vissza, és a parancsértelmező próbálja meg a saját parancsaiként értelmezni.

Eddig ez volt az alapértelmezés, “akármilyen parancsértelmezőnk is volt”.

Később az `sh(1)` kódjába bekerült egy aprócska okosítás, amivel megnézte az állomány első két karakterét, és ha az `:\n` volt, akkor a futtatáshoz maga helyett a `csh(1)` parancsértelmezőt hívta meg (ezt állítólag először a SCO csinálta).

A FreeBSD viszont végignézi a betöltők teljes listáját, amiben a sor végén szerepel egy általános `#!` formátumú betöltő. Ez az állomány futtatásához használatos értelmezők kódját keresi, és ha egyet sem sikerül azonosítania, akkor a `/bin/sh` programot indítja el.

A Linux ABI támogatását a FreeBSD úgy oldja meg, hogy először észleli az ELF bináris bűvös számát (ekkor még nem tesz különbséget a FreeBSD, Solaris, Linux vagy más ELF típusú binárisokat használó operációs rendszerek közt).

Ezután az ELF formátum betöltője az ELF állomány megjegyzéseket tároló szakaszában *bélyegek* (brand) után kutat, ami SVR4 és Solaris ELF binárisok esetén nem létezik.

A Linux binárisokat működésükhöz a `brandelf(1)` segítségével Linux típusúnak kell *megbélyegezni*:

```
# brandelf -t Linux állomány
```

Miután ezt megcsináltuk, az ELF betöltő észre fogja venni az állomány Linux típusát.

Mikor az ELF betöltő észleli, hogy az állomány Linux típusú, kicseréli egy mutató értékét a `proc` struktúrában. Minden rendszerhívás ezen a mutatón keresztül érhető el (a hagyományos UNIX rendszerekben ez a rendszerhívásokat tartalmazó `sysent[]` struktúratömb). Emellett a frissen elindított program szoftveres megszakításait tartalmazó tömbjéhez beállítja a speciális jelzések kezelését, valamint a Linux modul által végzett néhány további (kisebb) javítást.

A Linux rendszerhívásokat tartalmazó tömb többek közt tartalmazza a `sysent[]` bejegyzések egy listáját, amelyek címei a rendszermag Linux moduljára mutatnak.

Amikor a Linux bináris hív egy rendszerhívást, a hozzá tartozó szoftveres megszakítás kódja a `proc` struktúrából a neki megfelelő rendszerhívás kódját hivatkozza, így FreeBSD rendszerhívás belépési pontja helyett a Linuxét kapja meg.

Ráadásul Linux módban a különböző állományok hivatkozásai is *átírányítódnak*. Ez lényegében olyan, mint amit az állományrendszerek csatlakoztatásánál a `union` beállítás csinál (ami *nem* egyezik meg az `unionfs` állományrendszerrel!). Ilyenkor az állományokat először a `/compat/linux/eredeti-hely` könyvtárban keresi, és *majd* ha ott nem találja, csak akkor kezdi el keresni az `/eredeti-hely` ponton. Ezzel oldhatjuk meg, hogy más binárisok futtatását igénylő binárisok is képesek legyenek rendesen működni (például így az egész linuxos eszköztár tud futni a Linux ABI-n keresztül). Egyúttal arra is utal, hogy ha a Linux binárisok számára nem áll rendelkezésre a megfelelő bináris, akkor FreeBSD binárisokat is el tudnak indítani. Ha a `uname(1)` programot pedig bemásoljuk a `/compat/linux` könyvtáron belülre, akkor a Linux binárisok képtelenek lesznek megmondani, hogy nem Linux alatt futnak.

Így lényegében egy Linux magot találunk a FreeBSD rendszermagjában. A benne megtalálható különböző szolgáltatásokat megvalósító függvények: az állományműveletek, a virtuális memória kezelése, a jelzések küldése és System V típusú folyamatok közti kommunikáció stb. megegyeznek a FreeBSD és a Linux hívásai esetén egyaránt. Egyetlen eltérés, hogy a FreeBSD binárisok a FreeBSD *segédfüggvényein* (glue function), a Linux binárisok pedig a Linux *segédfüggvényein* keresztül férnek hozzájuk (a legelső operációs rendszerek tulajdonképpen csak a saját *segédfüggvényeiket* tartalmazták: a hívást kezdeményező program `proc` struktúrájában a függvények dinamikusan beállított címe helyett egy globális `sysent[]` struktúratömbben tárolták a meghívható függvényeket).

Melyik közülük a FreeBSD natív ABI-ja? Ez teljesen lényegtelen. Alapvetően az egyetlen különbség csupán annyi (pillanatnyilag, de ez a jövőben még változhat, valószínűleg hamarosan), hogy a FreeBSD *segédfüggvényei* statikusan megtalálhatóak a rendszermagban, míg a Linux *segédfüggvényei* egyaránt elérhetőek modulból vagy statikus linkeléssel.

Na igen, de akkor ez most emuláció? Nem. Ez egy ABI, nem emuláció. Itt szó sincs emulátorról (ahogy szimulátorról sincs).

De akkor mégis miért hívják ezt sokszor “Linux emulációnak”? Hát hogy nehezebb legyen eladni a FreeBSD-t! Komolyra fordítva a szót: ennek a kezdeti változata akkoriban született meg, amikor erre még nem volt rendes szó. Nem mondhattuk, hogy a FreeBSD befordítás vagy egy modul betöltése nélkül képes lett volna Linux binárisokat futtatni, ezért valamilyen módon meg kellett neveznünk az ilyenkor betöltött kódot — ebből lett “a Linux emulátor”.

III. Rendszeradminisztráció

A FreeBSD kézikönyv fennmaradó fejezeteiben a FreeBSD rendszerek adminisztrációjának különböző aspektusait mutatjuk be. Mindegyik fejezet elején megtudhatjuk mit is fogunk megismerni a fejezet elolvasása során, illetve arról is információkat kapunk, hogy mivel kell már tisztában lennünk a tárgyalt anyag feldolgozásához.

Ezeket a fejezeteket annak érdekében alakítottuk ki, hogy az adott témákban ismereteket adjunk. Nincs köztük semmilyen sorrendi kötöttség, sőt, ezeket egyáltalán nem is szükséges elolvasni a FreeBSD alapvető használatához.

Fejezet 11. Beállítás és finomhangolás

Írta: Chern Lee. Az alapjául szolgáló bemutatást írta: Mike Smith. Valamint az alapját képző tuning(7) oldalt írta: Matt Dillon.

11.1. Áttekintés

A FreeBSD egyik fontos szempontja a rendszer megfelelő beállítása, aminek segítségével elkerülhetjük a későbbi frissítések során keletkező kellemetlenségeket. Ez a fejezet a FreeBSD beállítási folyamatából kíván minél többet bemutatni, köztük a FreeBSD rendszerek finomhangolására szánt paramétereket.

A fejezet elolvasása során megismerjük:

- hogyan dolgozzunk hatékonyan az állományrendszerekkel és a lapozóállományokkal;
- az `rc.conf` beállításának alapjait és a `/usr/local/etc/rc.d` könyvtárban található indítási rendszert;
- hogyan állítsunk be és próbáljunk ki egy hálózati kártyát;
- hogyan állítsunk be virtuális címeket a hálózati eszközeinken;
- hogyan használjuk az `/etc` könyvtárban megtalálható különféle konfigurációs állományokat;
- hogyan hangoljuk a FreeBSD működését a `sysctl` változóinak segítségével;
- hogyan hangoljuk a lemezek teljesítményét és módosítsuk a rendszermag korlátozásait.

A fejezet elolvasásához ajánlott:

- a UNIX és a FreeBSD alapjainak megértése (3 fejezet);
- a rendszermag beállításához és fordításához kötődő alapok ismerete (8 fejezet).

11.2. Kezdeti beállítások

11.2.1. A partíciók kiosztása

11.2.1.1. Alappartíciók

Amikor a `bsdlabel(8)` vagy a `sysinstall(8)` segítségével állományrendszereket telepítünk, nem szabad figyelmen kívül hagynunk a tényt, hogy a merevlemez egységeiben a külső sávokból gyorsabban lehet hozzáférni az adatokhoz, mint a belsőkből. Emiatt a kisebb és gyakrabban elérni kívánt állományrendszereket a meghajtó lemezének külsejéhez közel kell létrehozni, míg például a `/usr` partícióhoz hasonló nagyobb partíciókat annak belső része felé. A partíciókat a következő sorrendben érdemes kialakítani: gyökér (rendszerindító), lapozóállomány, `/var` és `/usr`.

A `/var` méretének tükröznie kell a számítógép szándékolt használatát. A `/var` partíción foglalnak helyet a felhasználók postaládái, a naplóállományok és a nyomtatási sorok. A postaládák és a naplóállományok egészen váratlan mértékben is képesek megnövekedni attól függően, hogy mennyi felhasználónk van a rendszerben és hogy mekkora naplókat tartunk meg. Itt a legtöbb felhasználónak soha nem lesz szüksége egy gigabyte-nál több helyre.

Megjegyzés: Bizonyos esetekben a `/var/tmp` könyvtárban azért ennél több tárterület szükséges. Amikor a `pkg_add(1)` segítségével egy friss szoftvert telepítünk a rendszerünkre, akkor a program a `/var/tmp` könyvtárba tömöríti ki a hozzá tartozó csomag tartalmát. Ezért a nagyobb szoftvercsomagok, mint például a **Firefox** vagy az **OpenOffice** esetén gondok merülhetnek fel, ha nem rendelkezünk elegendő szabad területtel a `/var/tmp` könyvtárban.

A `/usr` partíció tartalmaz számos, a rendszer működéséhez elengedhetetlenül fontos állományt, többek közt a portok gyűjteményét (ajánlott, lásd `ports(7)`) és a forráskódot (választható). A portok és az alaprendszer forrásai telepítés során választhatóak, de telepítésük esetén akkor ezen a partíción legalább két gigabyte-nyi hely ajánlott.

Vegyük figyelembe a tárbeli igényeket, amikor megválasztjuk a partíciók méretét. Igen kellemetlen lehet, amikor úgy futunk ki az egyik partíción a szabad helyből, hogy a másikat alig használjuk.

Megjegyzés: Egyes felhasználók szerint előfordulhat, hogy a `sysinstall(8)` `Auto-defaults` opciója a `/var` és `/` partíciók méretét túl kicsire választja. Particionáljunk okosan és nagylelkűen!

11.2.1.2. A lapozóállomány partíciója

Általános szabály, hogy a lapozóállományt tároló partíció mérete legyen a rendszer fizikai memóriájának (RAM) kétszerese. Például, ha a számítógépünk 128 megabyte memóriával rendelkezik, akkor a lapozóállomány méretének 256 megabyte-nak kell lennie. Az ennél kevesebb memóriát maguknak tudó rendszerek több lapozóállománnyal jobban teljesítenek. 256 megabyte-nál kevesebb lapozóállományt semmiképpen sem ajánlunk, és inkább a fizikai memóriát érdemes bővítenünk. A rendszermag virtuális memóriát kezelő lapozási algoritmusait úgy állították be, hogy abban az esetben teljesítsenek a legjobban, ha a lapozóállomány mérete legalább kétszerese a központi memória mennyiségének. A túl kicsi lapozóállomány beállítása rontja a virtuális memória lapkeresésési rutinjának hatékonyságát és a memória bővítése esetén még további gondokat is okozhat.

A több SCSI-lemezzel (vagy a különböző vezérlőkre csatlakoztatott több IDE-lemezzel) bíró nagyobb rendszerek esetében érdemes minden egyes (de legfeljebb négy) meghajtóra beállítani lapozóállományt. A lapozóállományoknak közel azonos méretűnek kell lenniük. A rendszermag tetszőleges méretűeket képes kezelni, azonban a belsejében alkalmazott adatszerkezetek a legnagyobb lapozóállomány méretének négyszereséig képesek növekedni. Ha a lapozóállományokat nagyjából ugyanazon a méreten tartjuk, akkor a rendszermag képes lesz a lapozáshoz felhasznált területet optimálisan elosztani a lemezek között. A nagyobb lapozóállományok használata még akkor is jól jön, ha nem is használjuk annyira. Segítségével sokkal könnyebben talpra tudunk állni egy elszabadult program tombolásából, és nem kell rögtön újraindítanunk a rendszert.

11.2.1.3. Miért particionáljunk?

Egyes felhasználók úgy gondolják, hogy egyetlen nagyobb méretű partíció mindenre megfelel, ám ez a gondolat több okból is helytelennek tekinthető. Először is, minden egyes partíciónak eltér a működési jellemzője, és különválasztásukkal lehetővé válik az állományrendszerek megfelelő behangolása. Például a rendszerindításhoz használt és a `/usr` partíciókat többségében csak olvasásra használják, és nem sokat írnak rájuk. Eközben a `/var` és `/var/tmp` könyvtárakban zajlik az írák és olvasások túlnyomó része.

A rendszer megfelelő felosztásával a kisebb, intenzívebben írt partíciókon megjelenő töredezettség nem szivárogoz át a többségében csak olvasásra használt partíciókra. Ha a sokat írt partíciókat közel tartjuk a lemez széléhez, akkor

azokon a partíciókon növekszik az I/O teljesítménye, ahol az a leggyakrabban megjelenik. Mivel mostanság az I/O teljesítményére inkább a nagyobb partíciók esetén van szükség, azzal nem érünk el ebben különösebb mértékű növekedést, ha a `/var` partíciót a lemez szélére toljuk. Befejezésképpen hozzátesszük, hogy ennek vannak biztonsági megfontolásai is. Egy kisebb és takarosabb rendszerindító partíció, ami többnyire írásvédett, nagyobb eséllyel él túl egy csúfos rendszerösszeomlást.

11.3. A mag beállítása

A rendszer beállításaira vonatkozó információk központi lelőhelye az `/etc/rc.conf` állomány. Ez az állomány tartalmazza a beállításokra vonatkozó adatok széles körét, amelyet elsősorban a rendszer indulása során a rendszer beállítására használnak. Erre a neve is utal: ez az `rc*` állományok konfigurációs állománya.

A rendszergazda az `rc.conf` állományban tudja felülbírálni az `/etc/defaults/rc.conf` állományban szereplő alapértelmezett beállításokat. Az alapértelmezéseket tartalmazó állományt nem szabad közvetlenül átmásolni az `/etc` könyvtárba, hiszen alapértelmezett értékeket tartalmaz, nem pedig mintákat. Minden rendszerfüggő beállítást magában az `rc.conf` állományban kell elvégezni.

Számos stratégia létezik a tömegesen adminisztrált számítógépeknél a közös és rendszerfüggő beállítások különválasztására, ezáltal a karbantartási költségek csökkentésére. A közös beállításokat ajánlott egy másik helyre, például az `/etc/rc.conf.site` állományba rakni, majd hivatkozni erre a kizárólag csak rendszerfüggő információkat tartalmazó `/etc/rc.conf` állományból.

Mivel az `rc.conf` állományt az `sh(1)` dolgozza fel, ezt elég könnyen el tudjuk érni. Például:

- `rc.conf`:

```
. /etc/rc.conf.site
hostname="node15.example.com"
network_interfaces="fxp0 lo0"
ifconfig_fxp0="inet 10.1.1.1"
```

- `rc.conf.site`:

```
defaultrouter="10.1.1.254"
saver="daemon"
blanktime="100"
```

Az `rc.conf.site` állományt ezt követően az `rsync` parancs használatával már szétszórható a rendszerben, miközben az `rc.conf` állomány mindenkinél egyedi marad.

Ha a rendszert a `sysinstall(8)` vagy a `make world` használatával frissítjük, akkor az `rc.conf` tartalma nem íródik felül, így a rendszer beállításairól szóló adatok nem vesznek el.

11.4. Az alkalmazások beállítása

A telepített alkalmazások általában saját konfigurációs állományokkal, azok pedig saját formátummal stb. rendelkeznek. Fontos, hogy ezeket az állományokat az alaprendszerrel elkülönítve tároljuk, ezáltal a csomagkezelő eszközök könnyen rájuk tudjanak találni és dolgozni velük.

Ezeket az állományokat általában a `/usr/local/etc` könyvtárban találjuk meg. Amennyiben egy alkalmazáshoz több konfigurációs állomány is tartozik, akkor ahhoz ezen belül egy külön alkönyvtár jön létre.

Normális esetben, amikor egy portot vagy csomagot telepítünk, minta konfigurációs állományokat is kapunk. Ezek nevében többnyire a `.default` utótag szerepel. Ha még nincs konfigurációs állomány az adott alkalmazáshoz, akkor a `.default` jelzésű állományokból ez létrehozható.

Példaképpen most tekintsük a `/usr/local/etc/apache` könyvtár tartalmát:

```
-rw-r--r-- 1 root wheel 2184 May 20 1998 access.conf
-rw-r--r-- 1 root wheel 2184 May 20 1998 access.conf.default
-rw-r--r-- 1 root wheel 9555 May 20 1998 httpd.conf
-rw-r--r-- 1 root wheel 9555 May 20 1998 httpd.conf.default
-rw-r--r-- 1 root wheel 12205 May 20 1998 magic
-rw-r--r-- 1 root wheel 12205 May 20 1998 magic.default
-rw-r--r-- 1 root wheel 2700 May 20 1998 mime.types
-rw-r--r-- 1 root wheel 2700 May 20 1998 mime.types.default
-rw-r--r-- 1 root wheel 7980 May 20 1998 srm.conf
-rw-r--r-- 1 root wheel 7933 May 20 1998 srm.conf.default
```

Az állományok mérete jól mutatja, hogy csak az `srm.conf` változott meg. Az **Apache** későbbi frissítései ezt az állományt nem fogják felülírni.

11.5. Szolgáltatások indítása

Írta: Tom Rhodes.

A felhasználók közül sokan választják a FreeBSD Portgyűjteményében található külső szoftverek telepítését. A telepített szoftvert ilyenkor gyakran úgy kell beállítani, hogy a rendszer indulásával együtt induljon. Az olyan szolgáltatások, mint például a `mail/postfix` vagy a `www/apache13` csupán két olyan szoftvercsomag, amelyet a rendszerrel együtt kell elindítani. Ebben a szakaszban a külső szoftverek indítására használatos eljárásokkal foglalkozunk.

A FreeBSD-ben megjelenő legtöbb szolgáltatás, mint például a `cron(8)`, a rendszerindító szkripteken keresztül kel életre. Habár ezek a szkriptek a FreeBSD egyes verziói vagy az egyes gyártók esetén különbözhetnek, azonban az mindegyikükben közös, hogy az elindításukra vonatkozó beállítások egyszerű indítószkriptekkel adhatóak meg.

11.5.1. Az alkalmazások részletesebb beállítása

Most miután a FreeBSD rendelkezik egy `rc.d` könyvtárral, az alkalmazások indításának beállítása is könnyebbé és ügyesebbé vált. Az `rc.d` működéséről szóló szakaszban megismert kulcsszavak segítségével az alkalmazások mostantól kezdve a többi szolgáltatás, például a DNS után indulnak el, és az `rc.conf` állományon keresztül a szkriptekbe huzalozottak helyett most már tetszőleges paramétereket is átadhatunk stb. Egy egyszerű szkript ehhez hasonlóan néz ki:

```
#!/bin/sh
#
# PROVIDE: utility
# REQUIRE: DAEMON
# KEYWORD: shutdown
```

```
. /etc/rc.subr

name=utility
rcvar=utility_enable

command="/usr/local/sbin/utility"

load_rc_config $name

#
# NE VÁLTOZTASSUK MEG AZ ITT LÉVŐ ALAPÉRTTELMEZÉSEKET,
# INKÁBB AZ /etc/rc.conf ÁLLOMÁNYBAN ÁLLÍTSUK BE EZEKET
#
utility_enable=${utility_enable-"NO"}
pidfile=${utility_pidfile-"/var/run/utility.pid"}

run_rc_command "$1"
```

Ez a szkript gondoskodik arról, hogy a **utility** nevű alkalmazás a DAEMON szolgáltatás után induljon el. Emellett még felkínál egy módszert a PID avagy futó programok azonosítójának beállítására és nyomonkövetésére is.

Ezt követően az `/etc/rc.conf` állományból az alkalmazás elindítható az alábbi sor hozzáadásával:

```
utility_enable="YES"
```

Ez a módszer megkönnyíti a parancssorban átadott paraméterek módosítását, az `/etc/rc.subr` állományban szereplő alapértelmezett függvények használatát, az `rcorder(8)` segédprogrammal szembeni kompatibilitást és az `rc.conf` állomány könnyebb beállítását.

11.5.2. Szolgáltatások indítása szolgáltatásokkal

Más szolgáltatások, mint például a POP3 vagy IMAP szerverek démonai stb. az `inetd(8)` segítségével indíthatóak el. Ez a Portgyűjteményből telepített szolgáltatások esetén magával vonja az adott segédprogram felvételét vagy a hozzá tartozó sor engedélyezését az `/etc/inetd.conf` állományban. Az **inetd** működésével és annak beállításával mélyrehatóbban az `inetd` szakasza foglalkozik.

A legtöbb esetben a `cron(8)` démon használata kézenfekvő a rendszerszintű szolgáltatások elindításában. Ez a megközelítés számos előnyt tartogat, mivel a `cron` ezeket a programokat a felhasználó `crontab` állománya alapján futtatja. Ezzel a mezey felhasználók számára is lehetővé válik, hogy elindítsanak és karbantartsanak alkalmazásokat.

A `cron` segédprogramnak van egy olyan speciális lehetősége, hogy az idő helyett a `@reboot` értéket adhatjuk meg. Ennek hatására a feladat a `cron(8)` indításával együtt fut le, tehát megszokott esetben a rendszer indítása során.

11.6. A cron segédprogram beállítása

Írta: Tom Rhodes.

A `cron(8)` a FreeBSD egyik leghasznosabb segédprogramja. A `cron` segédprogram a háttérben fut és folyamatosan figyeli az `/etc/crontab` állományt. Emellett a `cron` új `crontab` állományok után kutatva folyamatosan ellenőrzi a

`/var/cron/tabs` könyvtárat. Ezek a `crontab` állományok olyan feladatokról tárolnak adatokat, amelyeket a `cron` programnak egy adott pillanatban el kell végeznie.

A `cron` a konfigurációs állományok két külön fajtáját, a rendszer- és felhasználói `crontab`okat használja. A két típus között levő egyetlen különbség a hatodik mezőben található. A rendszerszintű `crontab`ok esetében a hatodik mező annak a felhasználónak a nevét tartalmazza, amivel a program fut. Ezzel a rendszer szintjén működő `crontab`oknak megadott az a képesség, hogy tetszőleges felhasználó nevében futtassanak programokat. A felhasználói `crontab`jaiban a hatodik mező a futtatandó parancsot tartalmazza, és ilyenkor az összes parancs a `crontab`ot létrehozó felhasználó nevében hajtódik végre. Ez utóbbi egy fontos biztonsági jellemző.

Megjegyzés: A felhasználói `crontab`ok lehetővé teszik az egyes felhasználók számára, hogy a `root` felhasználó jogosultságai nélkül képesek legyenek feladatokat ütemezni, ugyanis a felhasználóhoz tartozó `crontab`ban szereplő parancsok mindegyike a tulajdonosának engedélyeivel fut.

Az átlagos felhasználókhoz hasonlóan a `root` felhasználónak is lehet `crontab`ja, ami nem ugyanaz, mint az `/etc/crontab` (a rendszer saját `crontab` állománya). De mivel a rendszernek külön `crontab`ja van, ezért a `root` felhasználónak nem kell külön `crontab`ot létrehozni.

Vessünk egy pillanatazt az `/etc/crontab` (a rendszer `crontab`jának) tartalmára:

```
# /etc/crontab - a root crontabja FreeBSD alatt
#
# $FreeBSD: src/etc/crontab,v 1.32 2002/11/22 16:13:39 tom Exp $
# ❶
#
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin ❷
HOME=/var/log
#
#
#minute      hour    day      month    wday     who      command ❸
#
#
*/5          *       *         *         *         root     /usr/libexec/atrun ❹
```

- ❶ A FreeBSD legtöbb konfigurációs állományához hasonlóan itt is a `#` jelöli a megjegyzéseket. Az ilyen megjegyzések remekül használhatóak annak feljegyzésére, hogy mit és miért akarunk futtatni. A megjegyzések azonban nem szerepelhetnek a parancssal egy sorban, mivel máskülönben a parancs részeként kerülnek értelmezésre. Tehát mindig új sorba kell raknunk ezeket. Az üres sorokat a program nem veszi figyelembe.
- ❷ Először is meg kell adnunk egy környezetet. Az egyenlőség (=) karakter használatos a környezeti beállítások meghatározására, ahogy mindezt az itteni példában is tapasztalhatjuk a `SHELL`, `PATH` és `HOME` értékek esetében. Ha nem adunk meg mást, akkor a `cron` az alapértelmezés szerinti `sh` parancsértelmezőt használja. Ha nem adjuk meg a `PATH` változó értékét, akkor minden állományra abszolút elérési úttal kell hivatkoznunk, mivel ennek nincs alapértelmezett értéke. Ha nem definiáljuk a `HOME` változó értékét, akkor a `cron` a parancshoz tartozó felhasználó könyvtárából fog dolgozni.
- ❸ Ez a sor írja le a megadható hét mezőt. Az itt szereplő értékek a `minute` (perc), `hour` (óra), `mday` (a hónap napja), `month` (hónap), `wday` (a hét napja), `who` (ki) és `command` (mit). A mezők szinte maguktól értetődnek. A `minute` egy órán belül adja meg azokat a percek, amikor az adott parancsot le kell futtatni. A `hour` hasonló a `minute` beállításhoz, csak az itt szereplő értékét órákban kell értelmezni. Az `mday` a hónap napjaiban számol. A

month hasonló a minute és hour opciókhoz, de ez hónapot jelöl. A wday a hét egy napját jelzi. Ezeknek a mezőknek numerikus, valamint a huszonnégy órás időformátumnak megfelelő értékeket kell tartalmazniuk. A who mező, a többiekétől eltérő módon, csak az /etc/crontab állományban jelenik meg. Ez a mező adja meg, hogy a parancsot milyen felhasználóval kell futtatni. Ez az opció nem jelenik meg a felhasználók saját crontab állományainak telepítésekor. A sor végén láthatjuk még a command oszlopot is. Ez az utolsó mező, és ide kerül a végrehajtandó parancs.

- ④ Ez az utolsó sor a fentebb tárgyalt értékeket határozza meg. Észrevehetjük, hogy a sor egy */5 alakú felírással kezdődik, amelyet további * karakterek követnek. A * karakterek jelentése “első-utolsó”, ami arra utal, hogy *mindig*. Ennek megfelelően úgy értelmezhetjük ezt a sort, hogy a root felhasználóval le kell futtatni az atrun parancsot minden ötödik percben, függetlenül attól, hogy milyen nap vagy hónap van. Az atrun parancsról részletesebben az atrun(8) man oldalán kapunk felvilágosítást.

Az itt szereplő parancsoknak tetszőleges mennyiségű paraméter adható át, azonban a több soron keresztül átíró parancsok tördelését a sor végén a “\” karakterrel kell jelezni.

Ez mindegyik crontab állomány alapbeállítása, habár ettől általában egy dologban eltérnek. A hatodik mező, ahol a felhasználót adtuk meg, csak a rendszer /etc/crontab állományában jelenik meg. Ez a mező a felhasználók crontab állományaiból kimarad.

11.6.1. Egy crontab telepítése

Fontos: Nem kötelező az itt ismertetésre kerülő módon szerkeszteni vagy telepíteni a rendszer crontabját. Egyszerűen nyissuk meg a kedvenc szövegszerkesztőnkkel, és a cron segédprogram majd észreveszi, hogy az állomány megváltozott, majd ennek megfelelően neki is lát a módosított változat használatának. Erről a GYIK-ban (angolul) (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/admin.html#ROOT-NOT-FOUND-CRON-ERRORS) többet is megtudhatunk.

Egy frissen készített felhasználói crontab telepítéséhez először a kedvenc szövegszerkesztőnk segítségével létre kell hoznunk a megfelelő formátumú állományt, majd használnunk a crontab segédprogramot. Ennek általános alakja:

```
% crontab crontab_állomány
```

Ebben a példában a crontab_állomány a korábban létrehozott crontab neve lesz.

Lehetőségünk van lekérdezni a telepített crontab állományokat: egyszerűen adjuk át a -l kapcsolót a crontab parancsnak, és nézzük meg, mit ad vissza.

A crontab -e használata olyan felhasználók számára ajánlott, akik sablon alkalmazása nélkül szeretnének teljesen maguktól megírni egy crontab állományt. Ennek hatására a kiválasztott szövegszerkesztő egy üres állományt kap. Miután ezt az állományt elmentettük, a crontab programmal magától telepítésre kerül.

Ha a későbbiekben törölni akarjuk a felhasználónkhoz tartozó crontab állományt, akkor erre a célra használjuk a crontab -r kapcsolóját.

11.7. Az rc használata FreeBSD alatt

Írta: Tom Rhodes.

A rendszer indítására a FreeBSD 2002-ben átvette a NetBSD `rc.d` rendszerét. Ezt a felhasználók könnyen felismerhetik az `/etc/rc.d` könyvtárban található állományokról. A legtöbbjük olyan alapvető szolgáltatás, amelyet a `start`, `stop` és `restart` paraméterekkel lehet vezérelni. Például az `sshd(8)` az alábbi paranccsal indítható újra:

```
# /etc/rc.d/sshd restart
```

Ez az eljárás hasonló a többi szolgáltatás esetén is. Természetesen ezek a szolgáltatások általában maguktól indulnak el a rendszer indítása során az `rc.conf(5)` állományban megadottak szerint. Például ha a rendszerünk indulásakor szeretnénk aktiválni a hálózati címfordítással foglalatосkodó démont, akkor csak adjuk hozzá az `/etc/rc.conf` állományhoz a következő sort:

```
natd_enable="YES"
```

Amennyiben a `natd_enable="NO"` sor már szerepel benne, akkor egyszerűen írjuk át a `NO` értéket `YES`-re. Ezután az `rc` szkriptek a rendszer következő indításakor a lentieknek megfelelően automatikusan elindítják a hozzá tartozó szolgáltatásokat is.

Mivel az `rc.d` rendszert elsősorban arra használják, hogy szolgáltatásokat indítsanak el vagy állítsanak le az operációs rendszerrel együtt, a szabványos `start`, `stop` és `restart` paraméterek csak abban az esetben látják el a feladatukat, ha a nekik megfelelő változókat beállítottuk az `/etc/rc.conf` állományban. Tehát például az `sshd restart` csak abban az esetben fog bármit is csinálni, ha az `/etc/rc.conf` állományban az `sshd_enable` változót a `YES` értékre állítottuk. Ha az `/etc/rc.conf` beállításaitól függetlenül kívánunk egy szolgáltatásnak `start`, `stop` vagy `restart` parancsot adni, akkor elé kell tennünk egy “one” szót. Például ha az `sshd` szolgáltatás újraindításához az `/etc/rc.conf` tartalmát figyelmen kívül akarjuk hagyni, akkor ezt a parancsot kell kiadnunk:

```
# /etc/rc.d/sshd onerestart
```

Könnyen ellenőrizni tudjuk, hogy az adott szolgáltatás az `/etc/rc.conf` részéről engedélyezett-e, ha a neki megfelelő `rc.d` szkriptnek megadjuk az `rcvar` paramétert. Ennek segítségével például a rendszergazda így képes ellenőrizni, hogy az `sshd` szolgáltatást engedélyezi-e az `/etc/rc.conf`:

```
# /etc/rc.d/sshd rcvar
# sshd
$sshd_enable=YES
```

Megjegyzés: A második sor (`# sshd`) az `sshd` parancs kimenete, nem pedig a `root` parancssora.

A `status` paraméterrel kideríthetjük, hogy egy szolgáltatás aktív-e. Ezzel például így tudjuk ellenőrizni az `sshd` szolgáltatás működését:

```
# /etc/rc.d/sshd status
sshd is running as pid 433.
```

Az üzenet:

Az `sshd` a 433-as azonosítóval fut.

Bizonyos esetekben a `reload` paraméter használatával lehetőségünk van a szolgáltatások újraindítására is. Ilyenkor a rendszer megpróbál egy olyan jelzést küldeni a szolgáltatásnak, amivel a konfigurációs állományainak újraolvasását kéri. A legtöbbször lényegében ez a `SIGHUP` jelzés kiküldését rejti magában. Ez a lehetőség azonban nem mindegyik szolgáltatás esetén érhető el.

Az `rc.d` rendszer nem csupán hálózati szolgáltatások esetén használatos, hanem nagyrésztben hozzájárul a rendszer indításához is. Erre vegyük példának a `bgfsck` állományt. Amikor ez a szkript lefut, a következő üzenetet jeleníti meg:

```
Starting background file system checks in 60 seconds.
```

Az üzenet fordítása:

A háttérben 60 másodperc múlva megkezdődik az állományrendszerek ellenőrzése.

Ennek megfelelően tehát ezt az állományt az állományrendszerek háttérben folyó ellenőrzésére használják, ami pedig a rendszer indítása során fut le.

Számos rendszerszolgáltatás igényel a működéséhez további szolgáltatásokat. Például a NIS és más egyéb távoli eljárás híva alapján alapú szolgáltatások egészen addig nem képesek elindulni, amíg az `rpcbind` (portmapper) szolgáltatást el nem indítjuk. Az ilyen jellegű gondok feloldására az indító szkriptek elején levő megjegyzésekben található egy kevés metainformáció a szkript működéséhez szükséges elemekre (függőségeire) vonatkozóan. A rendszer indítása közben az `rcorder(8)` nevű program képes a megjegyzések közt ezeket az információkat feldolgozni és ebből megállapítani, hogy a függőségi viszonyok betartásával milyen sorrendben kell elindítani a rendszer által felkínált szolgáltatásokat.

Ehhez a következő kulcsszavakat kell megadni az egyes indító szkriptek elején (az `rc.subr(8)` így tudja “engedélyezni” az indító szkriptet):

- `PROVIDE`: segítségével megmondjuk, hogy ez az állomány milyen szolgáltatásokat nyújt.

A következő kulcsszavak az egyes indítóállományok elején szerepelhetnek. Nem kell feltétlenül használnunk ezeket, de velük az `rcorder(8)` munkáját segíthetjük:

- `REQUIRE`: felsoroljuk azokat a szolgáltatásokat, amelyek a futásához kellenek. Az állomány tehát az itt megadott szolgáltatások *után* fog lefutni.
- `BEFORE`: felsoroljuk azokat a szolgáltatásokat, amelyek *előtt* futtatni kell ezt az állományt.

Az indító szkriptekben a kulcsszavak ügyes megválasztásával a rendszergazda nagyon finoman képes az indításkor végrehajtódó szkriptek sorrendjét szabályozni és a többi UNIX alapú operációs rendszerből ismert “futtatási szintek” használata nélkül vezérelni a rendszerben megjelenő szolgáltatásokat.

Az `rc.d` rendszerről bővebben az `rc(8)` és `rc.subr(8)` man oldalakon olvashatunk. Ha szeretnénk saját `rc.d` szkripteket írni vagy javítani a már meglévőkön, akkor ez a cikk (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/rc-scripting) (angolul) segítségünkre lehet.

11.8. A hálózati kártyák beállítása

Írta: Marc Fonvieuille.

Manapság már el sem tudunk képzelni számítógépet hálózati csatlakozás nélkül. A hálózati csatoló kártyák

hozzáadása és beállítása egy FreeBSD rendszergazda mindennapos feladata.

11.8.1. A megfelelő meghajtóprogram felderítése

Mielőtt bárminek is nekikezdenénk, érdemes tisztában lennünk azzal, hogy a rendelkezésünkre álló kártya milyen típusú, milyen chipet használ és hogy PCI vagy ISA buszon csatlakozik-e. A FreeBSD a PCI és ISA csatolós kártyák széles spektrumát ismeri. Az egyes kiadásokhoz mellékelt "Hardware Compatibility List" (Hardverkompatibilitási lista) dokumentumokban tudjuk ellenőrizni, hogy a kártyákat ismeri a rendszer.

Miután meggyőződünk róla, hogy a kártyánkat ismeri a rendszer, meg kell keresnünk a hozzá tartozó meghajtót. A `/usr/src/sys/conf/NOTES` és a `/usr/src/sys/arch/conf/NOTES` állományok tartalmazzák a hálózati kártyák meghajtóinak rövid leírását, benne a támogatott chipsetek és kártyák típusaival. Ha ez alapján nem tudjuk teljes biztonsággal eldönteni, hogy melyik a számunkra megfelelő meghajtó, nézzük meg a saját man oldalát. Ezen a man oldalon megtaláljuk az általa ismert összes eszközt és a velük kapcsolatban előforduló jellemző problémákat.

Ha egy elterjedt típust sikerült beszerezni, akkor nem kell különösebben sokáig keresnünk a neki megfelelő meghajtót. Az ismertebb hálózati kártyák meghajtói ugyanis alpból benne vannak a `GENERIC` rendszermagban, ezért a rendszer indítása során ehhez hasonlóan meg is jelennek a kártyák:

```
dc0: <82c169 PNIC 10/100BaseTX> port 0xa000-0xa0ff mem 0xd3800000-0xd38
000ff irq 15 at device 11.0 on pci0
miibus0: <MII bus> on dc0
bmtphy0: <BCM5201 10/100baseTX PHY> PHY 1 on miibus0
bmtphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc0: Ethernet address: 00:a0:cc:da:da:da
dc0: [ITHREAD]
dc1: <82c169 PNIC 10/100BaseTX> port 0x9800-0x98ff mem 0xd3000000-0xd30
000ff irq 11 at device 12.0 on pci0
miibus1: <MII bus> on dc1
bmtphy1: <BCM5201 10/100baseTX PHY> PHY 1 on miibus1
bmtphy1: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc1: Ethernet address: 00:a0:cc:da:da:db
dc1: [ITHREAD]
```

Ebben a példában láthatunk is két olyan kártyát, amelyek a `dc(4)` meghajtót használják.

Ha a hálózati kártyánk meghajtója nem szerepel a `GENERIC` konfigurációban, akkor a működéséhez be kell tölteni a megfelelő meghajtót. Ezt alapvetően kétféleképpen érhetjük el:

- Ennek legegyszerűbb módja, ha a `kldload(8)` használatával alkalmanként vagy a `/boot/loader.conf` állományban a megfelelő sor hozzáadásával a rendszer indításával együtt betöltjük a hálózati kártya meghajtójához tartozó modult. Nem mindegyik hálózati kártya meghajtója érhető el modul formájában. Erre konkrét például szolgálnak az ISA kártyákhoz tartozó modulok.
- Másik lehetőségünk, ha statikusan beépítjük a kártyánk támogatását a rendszermagba. A `/usr/src/sys/conf/NOTES` és az `/usr/src/sys/arch/conf/NOTES` állományok, valamint a meghajtóhoz tartozó man oldal elolvasásából megtudhatjuk a rendszermag beállításait tartalmazó állományban megadandó paramétereket. A rendszermag újrafordítását lásd 8 fejezet. Ha a rendszermag (`GENERIC`) az indulás során észlelte a kártyánkat, nem kell újat készítenünk.

11.8.1.1. A Windows NDIS meghajtóinak használata

Sajnos még mindig sok olyan gyártó akad, akik a nyílt forrású közösség számára nem adják ki a meghajtók működésének alapjait, mivel az ilyen adatokat szakmai titoknak tekintik. Ebből következik, hogy a FreeBSD és más operációs rendszerek fejlesztői számára két választás marad: vagy a gyári meghajtók visszafejtésének hosszú és fájdalmas útján haladva fejlesztik ki a saját meghajtójukat, vagy pedig a Microsoft Windows platformra kiadott meghajtók binárisait hasznosítják. A legtöbb fejlesztő, köztük a FreeBSD fejlesztői is, ez utóbbi megközelítést választották.

Bill Paul (wpaul) jóvoltából a FreeBSD 5.3-RELEASE változatában megjelent a “Network Driver Interface Specification” (NDIS, avagy hálózati meghajtók szabványos felülete) “natív” támogatása. A FreeBSD NDISulator (másnéven Project Evil, a Gonosz terve) nevű komponense fog egy Windows-os meghajtót és elhiti vele, hogy a Windows operációs rendszerrel kommunikál. Mivel az ndis(4) meghajtó Windows binárisokat használ fel, ezért csak i386 és amd64 rendszerek esetén érhető el.

Megjegyzés: Az ndis(4) meghajtó leginkább a PCI, CardBus és PCMCIA csatolójú eszközök támogatására lett kitalálva, az USB eszközöket még nem ismeri.

Az NDISulator használatához három tényezőre van szükségünk:

1. A rendszermag forrása
2. a Windows XP meghajtó binárisa (.SYS a kiterjesztése)
3. a Windows XP meghajtó konfigurációs állománya (.INF a kiterjesztése)

Keressük meg az említett állományokat az adott kártyához. Ezeket általában a mellékelt CD-n vagy a gyártó honlapján találjuk meg. A most következő példákban a W32DRIVER.SYS és a W32DRIVER.INF neveket fogjuk használni.

Megjegyzés: A Windows i386 architektúrájú verziójához készült meghajtóprogramokat nem tudjuk a FreeBSD/amd64 verziójával használni. A működéshez amd64-re készült Windows-os meghajtókra van szükség.

A következő lépés a meghajtó binárisainak betölthető modulba fordítása. Ennek eléréséhez használjuk az ndisgen(8) parancsot a root felhasználóval:

```
# ndisgen /windowsos/meghajtó/W32DRIVER.INF /windowsos/meghajtó/W32DRIVER.SYS
```

Az ndisgen(8) egy interaktív segédprogram, amely működése közben még rákérdez néhány szükséges információra. Az aktuális könyvtárban létrehoz egy rendszermagmodult, amelyet az alábbi módon tudunk betölteni:

```
# kldload ./W32DRIVER_SYS.ko
```

Az előállított modul mellé be kell töltenünk még az ndis.ko és az if_ndis.ko modulokat is. Ez általában minden olyan modul esetén megtörténik magától, amely függ az ndis(4) használatától. Kézzel a következő parancsokkal tudjuk ezeket betölteni:

```
# kldload ndis
# kldload if_ndis
```

Itt az első parancs betölti az NDIS miniport meghajtó burkolására szánt kódot, valamint a másodikkal a tényleges hálózati csatolófelületet.

Most pedig a `dmesg(8)` kimenetében ellenőrizzük, hogy történt-e valamilyen hiba a betöltés során. Ha minden jól ment, akkor az alábbiakhoz hasonló kimenetet produkált:

```
ndis0: <Wireless-G PCI Adapter> mem 0xf4100000-0xf4101fff irq 3 at device 8.0 on pci1
ndis0: NDIS API version: 5.0
ndis0: Ethernet address: 0a:b1:2c:d3:4e:f5
ndis0: 11b rates: 1Mbps 2Mbps 5.5Mbps 11Mbps
ndis0: 11g rates: 6Mbps 9Mbps 12Mbps 18Mbps 36Mbps 48Mbps 54Mbps
```

Innentől kezdve az `ndis0` nevű eszközt úgy tudjuk használni, mint bármelyik más hálózati felületet (például `dc0`).

A többi modulhoz hasonló módon be tudjuk állítani, hogy a rendszer indulásával együtt betöltődjének az NDIS modulok. Ehhez először másoljuk az imént létrehozott modult, az `W32DRIVER_SYS.ko` állományt a `/boot/modules` könyvtárba. Ezután adjuk hozzá a következő sort a `/boot/loader.conf` állomány tartalmához:

```
W32DRIVER_SYS_load="YES"
```

11.8.2. A hálózati kártya beállítása

Ahogy betöltődött a megfelelő meghajtó a hálózati kártyánkhoz, be is kell állítanunk a kártyát. A hálózati kártyák sok más dologgal együtt beállíthatók a telepítés során a **sysinstall** segítségével.

A rendszerünkben beállított hálózati csatolófelületek megjelenítéséhez gépeljük be a következő parancsot:

```
% ifconfig
dc0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=80008<VLAN_MTU,LINKSTATE>
    ether 00:a0:cc:da:da:da
    inet 192.168.1.3 netmask 0xffffffff broadcast 192.168.1.255
    media: Ethernet autoselect (100baseTX <full-duplex>)
    status: active
dc1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=80008<VLAN_MTU,LINKSTATE>
    ether 00:a0:cc:da:da:db
    inet 10.0.0.1 netmask 0xffffffff broadcast 10.0.0.255
    media: Ethernet 10baseT/UTP
    status: no carrier
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
    options=3<RXCSUM,TXCSUM>
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x4
    inet6 ::1 prefixlen 128
    inet 127.0.0.1 netmask 0xff000000
    nd6 options=8010<POINTOPOINT,MULTICAST> mtu 1500
```

Az előbbi parancs kimenetében a következő eszközök jelentek meg:

- `dc0`: az első Ethernet felület

- `dc1`: a második Ethernet felület
- `pl1lp0`: a párhuzamos port felülete (amennyiben található párhuzamos port a számítógépben)
- `lo0`: a loopback eszköz

A FreeBSD a kártyához tartozó meghajtó nevével és egy sorszámmal azonosítja a rendszermag indulása során talált eszközöket. Például az `sis2` a rendszerben található harmadik olyan eszköz, amely a `sis(4)` meghajtót használja.

A példában a `dc0` eszköz aktív és működőképes. Ennek legfontosabb jelei:

1. Az `UP` szó mutatja, hogy a kártyát sikerült beállítani és készen áll a használatra.
2. A kártya internet (`inet`) címe (jelen esetünkben ez `192.168.1.3`).
3. Érvényes hálózati maszkkal rendelkezik (`netmask`, ahol a `0xffffffff00` a `255.255.255.0` címnek felel meg).
4. Érvényes broadcast (üzenetszóró) címmel rendelkezik (ami itt most `192.168.1.255`).
5. A kártya MAC-címe (`ether`) `00:a0:cc:da:da:da`.
6. A hozzá tartozó fizikai eszköz kiválasztása automatikus (`media: Ethernet autoselect (100baseTX <full-duplex>)`). Láthatjuk, hogy a `dc1` eszközt egy `10baseT/UTP` típusú fizikai eszközhöz állítottuk be. Az egyes meghajtókhoz tartozó fizikai módokról a nekik megfelelő man oldalakon olvashatunk.
7. A kapcsolat állapota (`status`) `active` értékű, tehát van vonal. A `dc1` esetén láthatjuk, hogy a `status: no carrier` (nincs vonal). Ez teljesen normálisnak tekinthető minden olyan esetben, amikor a kártyába még nem dugtunk Ethernet-kábelt.

Amennyiben az `ifconfig(8)` kimenete valami ilyesmi:

```
dc0: flags=8843<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
      options=80008<VLAN_MTU,LINKSTATE>
      ether 00:a0:cc:da:da:da
      media: Ethernet autoselect (100baseTX <full-duplex>)
      status: active
```

akkor az arra utal, hogy a kártyát nem állítottuk be.

A kártya beállításához a `root` felhasználó jogosultságaira van szükségünk. A hálózati kártyák beállítása az `ifconfig(8)` segítségével elvégezhető parancssorból is, de a gép újraindításakor az így megadott értékek elvesznek. Ezért az `/etc/rc.conf` állományba kell felvennünk a hálózati kártyák érvényes beállításait.

A kedvenc szövegszerkesztőnkben nyissuk meg az `/etc/rc.conf` állományt. Minden egyes hálózati csatolóhoz fel kell vennünk benne egy sort, ennek megfelelően most a példához tartozó módon az alábbiakat:

```
ifconfig_dc0="inet 192.168.1.3 netmask 255.255.255.0"
ifconfig_dc1="inet 10.0.0.1 netmask 255.255.255.0 media 10baseT/UTP"
```

A `dc0` és `dc1` neveket kell a rendszerünkben ténylegesen megtalálható eszközök neveire kicserélni, valamint megadni a nekik megfelelő címeket. A kártya meghajtójának és az `ifconfig(8)` man oldalának elolvasásával kideríthetjük az itt megadható további beállításokat, valamint az `rc.conf(5)` man oldalán részletesebben megismerhetjük az `/etc/rc.conf` formai követelményeit.

Ha a telepítés során beállítottuk volna a hálózati kapcsolatokat, akkor tapasztalhatjuk, hogy egyes hálózati kártyák sorai itt már szerepelnek. Ellenőrizzük az `/etc/rc.conf` tartalmát, mielőtt bővítenénk!

Mindezek mellett az `/etc/hosts` állományba is be kell írunk a helyi hálózatunkon található különféle gépek neveit és IP-címeit, ha még nem szerepelnének ott. Erről további részleteket a `hosts(5)` man oldalról és az `/usr/share/examples/etc/hosts` állományból tudhatunk meg.

Megjegyzés: Ha a géppel szeretnénk majd csatlakozni az internetre, akkor ne felejtsük el manuálisan beállítani az alapértelmezett átjárót és a névfeloldáshoz szükséges kiszolgálót:

```
# echo 'defaultrouter="alapertelmezett_atjaro"' >> /etc/rc.conf
# echo 'nameserver DNS_kiszolgalo' >> /etc/resolv.conf
```

11.8.3. Tesztelés és hibaelhárítás

Miután az `/etc/rc.conf` állományban elvégeztük a szükséges változtatásokat, érdemes újraindítanunk a rendszerünket. Ennek révén érvényesítjük a csatolófelületekkel kapcsolatos változtatásainkat és ellenőrizzük, hogy így a rendszer mindenféle hibaüzenet nélkül képes elindulni. A másik lehetőség, ha csak magát a hálózati alrendszer konfigurációját indítjuk el újra:

```
# /etc/rc.d/netif restart
```

Megjegyzés: Ha az `/etc/rc.conf` állományban már beállítottuk az alapértelmezett átjárót, akkor elegendő csupán ez a parancs:

```
# /etc/rc.d/routing restart
```

Ahogy újrakonfiguráltuk a hálózati alrendszert, ki is tudjuk próbálni a hálózati felületeket.

11.8.3.1. Az Ethernet kártyák tesztelése

Az Ethernet kártyák helyes beállításának vizsgálatához két dolgot kell kipróbálnunk. Először is pingeljük magát a felületet, majd ezután pingeljük meg a helyi hálózaton egy másik számítógépet.

Elsőként tehát próbáljuk meg a helyi felületet:

```
% ping -c5 192.168.1.3
PING 192.168.1.3 (192.168.1.3): 56 data bytes
64 bytes from 192.168.1.3: icmp_seq=0 ttl=64 time=0.082 ms
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=0.108 ms
64 bytes from 192.168.1.3: icmp_seq=4 ttl=64 time=0.076 ms

--- 192.168.1.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.074/0.083/0.108/0.013 ms
```

Most pedig pingeljük meg egy másik számítógépet a helyi hálózaton:

```
% ping -c5 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=0.726 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.766 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.700 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.747 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.704 ms

--- 192.168.1.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.700/0.729/0.766/0.025 ms
```

Ha beállítottuk az `/etc/hosts` állományt, akkor a `192.168.1.2` helyett a gép nevét is megadhatjuk.

11.8.3.2. A hibák elhárítása

A hardverek és szoftverek beállításaiban mindig is valódi kín megtalálni a hibákat, és ezeket a kínokat többnyire úgy tudjuk enyhíteni, ha először az egyszerű hibaforrásokat szűrjük ki. Csatlakoztattuk a hálózati kábelt? Tisztességesen beállítottuk a hálózati szolgáltatásokat? Jól állítottuk be a tűzfalat? A FreeBSD képes kezelni a kártyát? A hibajelentések elküldése előtt mindig bújjuk át a támogatott hardvereszközök listáját. A FreeBSD verziókat frissítsük a legújabb STABLE változatra. Olvassuk át a levelezési listák archívumait vagy legalább keressünk rá a témára az interneten.

Ha a kártya működik, de a teljesítménye nem kielégítő, érdemes ennek utánanézni a tuning(7) man oldalon. Ilyenkor érdemes ellenőrizni a hálózati beállításainkat is, mivel a helytelen beállítások gyakran okoznak teljesítményvesztést.

Bizonyos esetekben láthatunk egy vagy két `device timeout` típusú hibát is, ami a kártyák egyes fajtáinál elfogadható. Ha azonban folyamatosan megjelennek vagy zavaróvá válnak, érdemes utánanéznünk, hogy az eszköz nem ütközik-e valamelyik másikkal. Mindenképpen győződjünk meg a kábelek épségéről és csatlakoztatásáról. Még az is elképzelhető, hogy egyszerűen csak egy másik hálózati kártyára van szükségünk.

Néha felbukkanak `watchdog timeout` jellegű hibák is. Ilyenkor elsőként mindig a hálózati kábelt ellenőrizzük. Egyes kártyáknak olyan PCI foglalatra van szükségük, ami támogatja a Bus Mastering opciót. Néhány régebbi alaplapon csak ilyen PCI bővítőhely található (ami általában a 0. foglalat). Olvassunk utána a hálózati kártya és az alaplapon dokumentációjában, hátha ezek okozzák a problémát.

A `No route to host` üzenet akkor jelenik meg, ha a rendszer képtelen megállapítani, milyen úton juttassa el a csomagokat a megadott célhoz. Ez többnyire olyankor történik meg, amikor nem adtunk meg alapértelmezett kézbesítési irányt (default route) vagy nem dugtuk be a hálózati kábelt. A `netstat -rn` kimenetéből meg tudjuk állapítani, hogy létezik-e érvényes út az elérni kívánt cél felé. Ha nincs, akkor haladjunk tovább a 31 fejezetre.

A `ping: sendto: Permission denied` jellegű üzeneteket többségében egy helytelenül beállított tűzfal okozza. Ha az `ipfw` működését engedélyeztük a rendszermagban, de nem adtunk meg hozzá szabályokat, akkor az alapértelmezett házirend szerint minden forgalmat blokkolni fog, tehát még a pingeket is! Ezzel kapcsolatban a 30 fejezet elolvasását ajánljuk.

Előfordulhat, hogy a kártya teljesítménye igen gyenge vagy az átlagos alatt van. Ilyenkor a fizikai eszköz `autoselect` (automatikus) típusú kiválasztása helyett érdemes megadnunk a konkrét eszköznek megfelelő típust. Habár ez a legtöbb hardver esetén beválik, nem mindenki számára jelent megoldást. Ismételten csak annyit tudunk ehhez hozzátenni, hogy ellenőrizzük a hálózati beállításainkat és olvassuk el a tuning(7) man oldalt.

11.9. Virtuális címek

A FreeBSD alkalmazása során igen gyakori a virtuális címek használata, aminek segítségével egyetlen szerver több szerverként képes látszódni a hálózaton. Ezt úgy érik el, hogy egyetlen felülethez több hálózati címet rendelnek hozzá.

Az adott hálózati csatolófelületnek van egy “valódi címe” és tetszőleges számú “álcíme”. Ezeket az álcímeket általában az `/etc/rc.conf` állományban kell feltüntetni.

Az `fxp0` felület esetén az álcímek megadása valahogy így néz ki:

```
ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

Figyeljük meg, hogy az álcímekhez tartozó bejegyzések az `alias0` névvel kezdődnek és szám szerint növekvőleg következnek egymás után (például, `_alias1`, `_alias2` és így tovább). A beállítás a sorozat első kimaradó tagjánál megszakad.

Az álcímek hálózati maszkjának pontos meghatározása nagyon fontos, de szerencsére nem különösebben bonyolult. Minden felület esetén lennie kell egy olyan címnek, amely helyesen reprezentálja a hálózat hálózati maszkját. Minden egyéb olyan címnek, ami ugyanabba az alhálózatba esik, végig 1-esekből álló hálózati maszkkal kell rendelkezniük (ami felírható `255.255.255.255` vagy `0xffffffff` formájában is).

Például vegyük azt, hogy az `fxp0` felületen keresztül két hálózathoz csatlakozunk, melyek közül az egyik a `10.1.1.0`, amelynek hálózati maszkja `255.255.255.0`, és a `202.0.75.16`, amelynek hálózati maszkja `255.255.255.240`. Azt szeretnénk elérni, hogy a rendszerünk a `10.1.1.1` címtől a `10.1.1.5` címig, valamint a `202.0.75.17` címtől a `202.0.75.20` címig jelenjen meg a nekik megfelelő hálózatokon. Ahogy arra már fentebb is utaltunk, az adott hálózati tartományban csak az első címnek (ebben az esetben ez a `10.0.1.1` és a `202.0.75.17`) kell valódi hálózati maszkkal rendelkeznie. Minden további címnek (a `10.1.1.2` és `10.1.1.5` között, valamint a `202.0.75.18` és `202.0.75.20` között) legyen `255.255.255.255` a hálózati maszkja.

Az alábbi `/etc/rc.conf` bejegyzések ennek az elrendezésnek megfelelően állítják be a kártyát:

```
ifconfig_fxp0="inet 10.1.1.1 netmask 255.255.255.0"
ifconfig_fxp0_alias0="inet 10.1.1.2 netmask 255.255.255.255"
ifconfig_fxp0_alias1="inet 10.1.1.3 netmask 255.255.255.255"
ifconfig_fxp0_alias2="inet 10.1.1.4 netmask 255.255.255.255"
ifconfig_fxp0_alias3="inet 10.1.1.5 netmask 255.255.255.255"
ifconfig_fxp0_alias4="inet 202.0.75.17 netmask 255.255.255.240"
ifconfig_fxp0_alias5="inet 202.0.75.18 netmask 255.255.255.255"
ifconfig_fxp0_alias6="inet 202.0.75.19 netmask 255.255.255.255"
ifconfig_fxp0_alias7="inet 202.0.75.20 netmask 255.255.255.255"
```

11.10. Konfigurációs állományok

11.10.1. Az `/etc` felépítése

A beállításokkal kapcsolatos információk számos könyvtárban tárolódnak. Többek közt:

<code>/etc</code>	Általános rendszerszintű beállítások. Az itt levő adatok a rendszer egészére vonatkoznak.
<code>/etc/defaults</code>	A rendszer konfigurációs állományainak alapértelmezett változatai.

<code>/etc/mail</code>	A <code>sendmail(8)</code> beállításához tartozó további állományok, egyéb levélküldéshez használt adatok.
<code>/etc/ppp</code>	A felhasználói és rendszermag szintű <code>ppp</code> programok beállításai.
<code>/etc/namedb</code>	A <code>named(8)</code> működéséhez szükséges adatok alapértelmezett helye. Általában a <code>named.conf</code> és a zónák leírását tároló állományok kerülnek ide.
<code>/usr/local/etc</code>	A telepített alkalmazások konfigurációs állományai. Néha alkalmazásonként külön könyvtárakba kerülnek a benne található állományok.
<code>/usr/local/etc/rc.d</code>	A telepített alkalmazások indításával és leállításával kapcsolatos szkriptek.
<code>/var/db</code>	Automatikusan generált rendszerszintű adatbázisok a csomagokkal, a programok helyével stb. kapcsolatosan.

11.10.2. Hálózati nevek

11.10.2.1. `/etc/resolv.conf`

Az `/etc/resolv.conf` határozza meg, hogy a FreeBSD névfeloldója miként fér hozzá az internet tartománynév rendszeréhez (a DNS-hez).

Az `resolv.conf` állományban leggyakrabban a következő bejegyzések fordulnak elő:

<code>nameserver</code>	Annak a névszernek az IP-címe, ahova a névfeloldó küldi a kéréseit. A névszervereket a felírás sorrendjében kérdezi meg, maximum hármat.
<code>search</code>	A hálózati nevek keresőlistája. Ezt általában a helyi hálózati nevek tartománya határozza meg.
<code>domain</code>	A helyi tartomány neve.

Egy átlagos `resolv.conf` tartalma:

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```

Megjegyzés: Csak egy `search` és `domain` opciót szabad megadni.

A DHCP használatakor a `dhclient(8)` felül szokta írni a `resolv.conf` tartalmát a DHCP szervertől kapott információkkal.

11.10.2.2. `/etc/hosts`

Az `/etc/hosts` az internet kezdeti napjaira emlékeztető egyszerű szöveges adatbázis. A nevek és IP-címek közti leképezéseket a DNS és NIS rendszerekkel karöltve oldja fel. Ide a helyi hálózaton csatlakozó számítógépek neveit lehet beírni ahelyett, hogy erre a célra beállítanánk egy külön `named(8)` szervert. Ezenkívül még az `/etc/hosts`

állományba internetes nevek rekordját is felvehetjük, amivel így csökkenthetjük a gyakran használt nevek feloldására irányuló külső kéréseket.

```
# $FreeBSD$
#
#
# A hálózati nevek adatbázisa
#
# Ebbe az állományba rakjuk a helyi hálózaton található címeket és
# a hozzájuk tartozó hálózati neveket, ahol szinte ugyanez az
# adatbázis megtalálható. A 'my.domain' helyére a saját gépünk
# nevét írjuk be.
#
# A DNS vagy NIS alkalmazása esetén ez az állomány nem feltétlenül kerül
# felhasználásra. A névfeloldás sorrendjét az /etc/nsswitch.conf
# állományban adhatjuk meg.
#
::1                localhost localhost.my.domain
127.0.0.1          localhost localhost.my.domain

#
# Egy képzeletbeli hálózat.
#10.0.0.2          myname.my.domain myname
#10.0.0.3          myfriend.my.domain myfriend
#
# Az RFC 1918-nak megfelelően a következő IP-címekkel rendelkező
# alhálózatok sosem csatlakozhatnak közvetlenül az internetre:
#
#      10.0.0.0      -   10.255.255.255
#      172.16.0.0    -   172.31.255.255
#      192.168.0.0   -   192.168.255.255
#
# Amikor csatlakozunk az internethez, egy valódi, hivatalosan
# kiosztott számra lesz szükségünk. Ne találjunk ki magunknak
# hálózati címeket, hanem használjuk az internetszolgáltatótól
# kapott címet (amennyiben rendelkezünk # ilyennel) vagy az
# regionális internetes nyilvántartásban szereplő címek közül
# valamelyiket (ARIN, APNIC, LACNIC, RIPE NCC vagy AfriNIC).
```

Az /etc/hosts formai felépítése igen egyszerű:

```
[internetes cím] [hivatalos hálózati név] [álnév1] [álnév2] ...
```

Tehát például:

```
10.0.0.1 azEnValodiNevem.aHalozaton.hu azEnValodiNevem izemize1 izemize2
```

A részletekért keressük fel a hosts(5) man oldalt.

11.10.3. A naplóállományok beállítása

11.10.3.1. syslog.conf

A `syslog.conf` állomány a `syslogd(8)` program beállításait tartalmazza. Segítségével megadhatjuk, hogy a `syslog` által generált üzenetek egyes típusait milyen naplóállományokba mentjük.

```
# $FreeBSD$
#
# Ebben az állományban HASZNÁLHATÓAK szóközzök a mezők elválasztására,
# habár a többi *nix-típusú rendszer inkább tabulátorokat használ
# erre a célra. Ha több rendszeren is használni akarjuk ezt az
# állományt, akkor ne használjunk szóközzöket.
#
# A többbit lásd a syslog.conf(5) man oldalon.
#
.err;kern.debug;auth.notice;mail.crit          /dev/console
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages
security.*                                      /var/log/security
mail.info                                      /var/log/maillog
lpr.info                                       /var/log/lpd-errs
cron.*                                         /var/log/cron
*.err                                          root
*.notice;news.err                             root
*.alert                                       root
*.emerg                                       *
# Tegyük vissza ezt a sort, ha a /dev/console eszközre kiírt
# üzeneteket át akarjuk irányítani az /var/log/console.log állományba.
#console.info                                /var/log/console.log
# Ha az összes üzenetet a /var/log/all.log állományba akarjuk menteni,
# akkor tegyük vissza ezt a sort.
#*.*                                          /var/log/all.log
# Ha egy "loghost" nevű gépre szeretnénk naplózni, akkor tegyük vissza
# ezt a sort.
#*.*                                          @loghost
# Az inn használatakor tegyük vissza ezeket a sorokat.
# news.crit                                  /var/log/news/news.crit
# news.err                                  /var/log/news/news.err
# news.notice                               /var/log/news/news.notice
!startslip
*.*                                          /var/log/slip.log
!ppp
*.*                                          /var/log/ppp.log
```

A `syslog.conf(5)` man oldalának elolvasásával tudhatunk meg többet ezekről.

11.10.3.2. newsyslog.conf

A `newsyslog.conf` a `newsyslog(8)` beállításait tároló állomány. Ez egy olyan program, amelyet általában a `cron(8)` futtat le. A `newsyslog(8)` dönti el, hogy mikor van szükség a naplók archiválására és átrendezésére. Ennek során a `logfile` állományból `logfile.0` lesz, a `logfile.0` állományból pedig `logfile.1` és így tovább. Beállíthatjuk

úgy is, hogy a naplóállományokat archiválja gzip(1) formátumban, aminek megfelelően ezek logfile.0.gz, logfile.1.gz és ehhez hasonló névvel jönnek létre.

A newsyslog.conf megadja, hogy melyik naplóállományokat kell felügyelni, mennyi példányt tartunk meg belőlük és mikor kell velük foglalkozni. A naplóállományok átrendezhetőek és/vagy archiválhatóak egy adott méret elérésekor vagy egy adott idő eltelte után.

```
# A newsyslog konfigurációs állománya
# $FreeBSD$
#
# állománynév      [tulajdonos:csoport]  mód  darab  méret  mikor [ZB] [/pid_állomány] [jelzés]
/var/log/cron      600  3      100  *      Z
/var/log/amd.log    644  7      100  *      Z
/var/log/kerberos.log 644  7      100  *      Z
/var/log/lpd-errs  644  7      100  *      Z
/var/log/maillog    644  7      *    @T00   Z
/var/log/sendmail.st 644  10     *    168    B
/var/log/messages   644  5      100  *      Z
/var/log/all.log     600  7      *    @T00   Z
/var/log/slip.log    600  3      100  *      Z
/var/log/ppp.log     600  3      100  *      Z
/var/log/security    600  10     100  *      Z
/var/log/wtmp        644  3      *    @01T05 B
/var/log/daily.log   640  7      *    @T00   Z
/var/log/weekly.log  640  5      1    $W6D0  Z
/var/log/monthly.log 640  12     *    $M1D0  Z
/var/log/console.log 640  5      100  *      Z
```

További információkat a newsyslog(8) man oldaláról nyerhetünk.

11.10.4. sysctl.conf

A sysctl.conf állomány leginkább az rc.conf állományhoz hasonlít, benne az értékeket változó=érték párokban adhatjuk meg. Az itt definiált értékek akkor kerülnek ténylegesen beállításra, amikor a rendszer többfelhasználós módba vált. Ezen a módon nem mindegyik változó értékét tudjuk átállítani.

A sysctl.conf állományban az alábbi érték beállításával tudjuk beállítani, hogy a rendszer ne naplózza, amikor a programok végzetes jelzéssel fejeződnek be, valamint azt, hogy a felhasználók láthassák egymás futó programjait:

```
# Ne naplózzuk a végzetes jelzésekhez (például sig 11) tartozó kilépéseket.
kern.logsigexit=0

# Ne engedjük a felhasználóknak, hogy lássák egy másik felhasználó
# azonosítójával futó programokat.
security.bsd.see_other_uids=0
```

11.11. Finomhangolás a sysctl használatával

A sysctl(8) egy olyan felület, amely lehetőséget biztosít egy működő FreeBSD rendszer megváltoztatására. Segítségével többek közt hozzáférhetünk a TCP/IP protokollkészlet és a virtuális memóriát kezelő alrendszer rengeteg apró opciójához, melyek megfelelő beállításával egy tapasztalt rendszergazda kezében drasztikusan növelhető a rendszer teljesítménye. A sysctl(8) alkalmazásával több mint ötszáz rendszerszintű változó kérdezhető le és állítható be.

A sysctl(8) két funkciót rejt magában: a rendszer beállításainak lekérdezését és módosítását.

Így nézhetjük meg az összes lekérdezhető változót:

```
% sysctl -a
```

Így kérhetjük egy konkrét változó, például a kern.maxproc értékét:

```
% sysctl kern.maxproc
kern.maxproc: 1044
```

Egy adott változó értékének módosításához pedig használjuk a *változó=érték* felírást:

```
# sysctl kern.maxfiles=5000
kern.maxfiles: 2088 -> 5000
```

A sysctl változók értékei lehetnek karakterláncok, számok és logikai értékek (ahol az 1 az igennek, a 0 a nemnek felel meg).

Ha a számítógép indításakor automatikusan be akarunk állítani bizonyos változókat, akkor vegyük fel ezeket az `/etc/sysctl.conf` állományba. Ennek pontosabb részleteit a sysctl.conf(5) man oldalon és a 11.10.4 Szakaszban találhatjuk meg.

11.11.1. A sysctl(8) írásvédett értékei

Írta: Tom Rhodes.

Egyes esetekben szükséges lehet a sysctl(8) írásvédett változóinak módosítása. Habár gyakran elengedhetetlen, ezt kizárólag csak a rendszer (újra)indításakor tudjuk megtenni.

Például egyes laptopoknál a cardbus(4) eszköz nem próbálkozik több memóriaterület használatával, ezért egy ehhez hasonló hibával leáll:

```
cbb0: Could not map register memory
device_probe_and_attach: cbb0 attach returned 12
```

Az ilyen és ehhez hasonló esetekben gyakran olyan sysctl(8) változók alapértelmezett értékeit kellene megváltoztatnunk, amelyek írásvédettek. Ilyenkor tegyük az érintett sysctl(8) változó "objektumazonosítóját" (OID) és a hozzá tartozó értéket a `/boot/loader.conf` állományunkba. Az alapértelmezéseket a `/boot/defaults/loader.conf` állományban találjuk meg.

A fentebb tárgyalt probléma megoldásához a felhasználónak a `hw.pci.allow_unsupported_io_range=1` értéket kell beállítania az előbb említett állományban. Ezután már a cardbus(4) megfelelően fog működni.

11.12. A lemezek finomhangolása

11.12.1. Sysctl változók

11.12.1.1. `vfs.vmiodirenable`

A `vfs.vmiodirenable` sysctl változó értéke lehet 0 (ki) vagy 1 (be, és ez az alapértelmezés is). Ez a változó vezérli a könyvtárak gyorsítótárazását a rendszerben. A könyvtárak többsége kis méretű, így az állományrendszerből csak egyetlen (általában 1 KB méretű) darabkát használnak és még ennél is kevesebbet (általában 512 byte-ot) a pufferben. A változó kikapcsolt (avagy 0) értéke mellett a puffer csak rögzített számú könyvtárat táraz be még abban az esetben is, amikor temérdek mennyiségű memória áll a rendelkezésére. Ha viszont (az 1 értékkel) engedélyezzük, akkor a rendszer a könyvtárak tárazására felhasználja a virtuális memóriában puffereelt lapokat is, amivel lényegében az összes elérhető memóriát a könyvtárak tárazására fordítja. Ilyenkor azonban az egyes könyvtárak tárazására használt legkisebb memóriaterület a fizikai lapmérettel egyezik meg (ami általában 4 KB) és nem 512 byte. Abban az esetben javasoljuk ennek a beállításnak a használatát, ha olyan szolgáltatásokkal dolgozunk, amelyek nagy számú állománnyal dolgoznak egyszerre. Ilyen szolgáltatások többek közt a webes gyorsítótárak, nagyobb levelezőrendszerek és hírendszerek. Az opció engedélyezése alapvetően nem veti vissza a rendszer teljesítményét még akkor sem, ha ezzel memóriát pazarlunk el, de ezt igazából érdemes kikísérletezni.

11.12.1.2. `vfs.write_behind`

A `vfs.write_behind` sysctl változó alapértelmezett értéke 1 (bekapcsolt). Ez arra utasítja az állományrendszert, hogy csak akkor küldje ki az adatokat az eszközre, ha belőlük teljes fűrtök gyűlte össze. Ez jellemző módon nagyobb szekvenciális állományok írása esetén kedvező. Arra szolgál, hogy segítségével el lehessen kerülni az I/O túlságosan gyakori módosítások okozta terhelését. Bizonyos körülmények közt ez azonban lassíthatja a futó programok működését, ezért ilyenkor érdemes megfontolni a kikapcsolását.

11.12.1.3. `vfs.hirunningspace`

A `vfs.hirunningspace` sysctl változó értéke azt adja meg, hogy tetszőleges számú példánynál rendszerszinten mekkora mértékű írási művelet irányítható át a lemezvezérlők soraiba. Az alapértelmezés többnyire elegendő, de olyan gépeken, ahol sok lemez dolgozik egyszerre, ez az érték négy vagy öt *megabyte-ra* is felszökhet! Hozzátennénk, hogy ha ezt az értéket túlságosan nagyra állítjuk (és így túllépjük a puffer írási küszöbértékét), akkor ezzel hihetetlenül gyenge fűrtözési teljesítményt nyerünk. Semmiképp se állítsuk túlzottan nagy értékre! A nagyobb írási értékek a velük párhuzamos olvasások számára késleltetést is jelentenek.

Találhatunk még más egyéb pufferelési és gyorsítótárazási sysctl változókat, azonban ezek megváltoztatását egyáltalán nem javasoljuk, mivel a virtuális memória alrendszer kiválóan tudja önállóan állítani ezeket a paramétereit.

11.12.1.4. `vm.swap_idle_enabled`

A `vm.swap_idle_enabled` sysctl változó módosítása olyan nagyobb többfelhasználós rendszerekben bizonyulhat hasznosnak, ahol sok felhasználó lép be és lép ki a rendszerbe és sok az üresjáratban futó program. Az ilyen jellegű rendszerek hajlamosak nagy mennyiségű folyamatos terhelést mérni a tartalékolt szabad memóriára. A beállítás engedélyezésével, valamint a `vm.swap_idle_threshold1` és a `vm.swap_idle_threshold2` változókon keresztül a kilapozás "reakcióidejének" alkalmas behangolásával a megszokottnál gyorsabban lenyomhatjuk az

üresjáratban dolgozó programokhoz tartozó memórialapok prioritását, amivel a kilapozásokat vezérlő démon kezére játszunk. Azonban tényleg csak akkor engedélyezzük ezt a lehetőséget, ha valóban szükségünk van rá, mivel így a memóriát jóval előbb lapozzuk ki és ezzel több lapozóállományt és lemezteljesítményt emésztenek fel. Kisebb rendszerekben jól behatárolható a hatása, azonban a nagyobb rendszerekben, ahol már eleve visszafogott mértékű lapozás történik, ez a beállítás lehetővé teszi a virtuális memóriát kezelő alrendszer számára, hogy könnyedén ki- és be rakosgasson komplett futó programokat a memóriába.

11.12.1.5. `hw.ata.wc`

A FreeBSD 4.3 egyszer már kacérkodott az IDE-lemezek írási pufferének kikapcsolásával. Ez ugyan csökkentette az IDE-lemezek írási sávszélességét, azonban bizonyos merevlemezgyártók gondatlanságából eredő súlyos adatvesztések miatt szükséges volt a használata. A gond ezzel kapcsolatban ott van, hogy egyes IDE-meghajtók hazudnak az íráskor teljesítéséről. A lemezek írási gyorsítótárazásának bekapcsolásával az IDE-meghajtók nem csak az íráskor sorrendjét rendezik át, hanem nagyobb terhelés esetén egyes blokkokat jóval később is rögzítenek. Ezért a rendszer esetleges összeomlása vagy egy áramkimaradás súlyos károkat okozhat az állományrendszerben. A FreeBSD úgy döntött, hogy a megbízhatóságot választja. Sajnos ez olyan nagyságú teljesítményvesztést okozott, hogy a következő kiadásban már kénytelenek voltunk alapértelmezés szerint is visszakapcsolni ezt a lehetőséget. A `hw.ata.wc` nevű `sysctl` változó vizsgálatával ellenőrizhetjük a rendszerünkön érvényes alapértelmezett beállítást. Amennyiben az IDE íráskor gyorsítótárazása nem engedélyezett, akkor ezt a változó értékének 1-re állításával állíthatjuk vissza. Ezt a rendszer indításakor a rendszerbetöltőben tehetjük meg. A rendszermag indítása után ennek már nincs hatása.

A részleteket a `ata(4)` man oldalon tudhatjuk meg.

11.12.1.6. `SCSI_DELAY` (`kern.cam.scsi_delay`)

A rendszermag `SCSI_DELAY` nevű beállítása a rendszer indulásának idejét hivatott mérsékelni. Az alapértelmezett értéke viszonylag magas, innen származik a rendszer indítása során keletkező 15 másodperces csúszás. Általában az is megfelelő, ha ezt visszavesszük az 5 értékre (főleg a modernebb meghajtók számára). A FreeBSD újabb (5.0 vagy későbbi) változataiban ez az érték már a `kern.cam.scsi_delay` `sysctl` változó értékével is megadható a rendszer indításakor. Azonban ügyeljünk rá, hogy mind a finomhangoláshoz használt változó, mind pedig rendszermag beállítása *ezredmásodpercben* és *nem másodpercben* értelmezi ezt az értéket.

11.12.2. Soft Updates

A `tunefs(8)` nevű program használható az állományrendszerek finomhangolására. Nagyon sok opciót találhatunk benne, de itt most csak a “Soft Updates” ki- és bekapcsolásával foglalkozunk, amit a következő módon tehetünk meg:

```
# tunefs -n enable /allomanyrendszer
# tunefs -n disable /allomanyrendszer
```

Amíg egy állományrendszer csatlakoztatott állapotban van, addig nem módosítható a `tunefs(8)` paranccsal. A Soft Updates bekapcsolására ezért az a legalkalmasabb időpont, amikor egyfelhasználós módban vagyunk és még egyetlen partíciót sem csatlakoztattunk.

A Soft Updates beállítás engedélyezése a memóriában pufferelt gyorsítótáron keresztül jelentős mértékben fokozza a metaadatok teljesítményét, elsősorban az állományok létrehozását és törlését. A Soft Updates használatát ezért minden állományrendszer esetén ajánljuk. A Soft Updates alkalmazásának két rossz oldalára kell tekintettel lennünk.

Először is a Soft Updates a rendszer összeomlása esetén ugyan garantálja az állományrendszer konzisztenciáját, de könnyen elképzelhető, hogy több másodperccel (vagy akár egy egész perccel!) hátrébb jár a fizikai lemez frissítésében. Másodszor a Soft Updates késlelteti az állományrendszer blokkjainak felszabadítását. Ha van egy olyan állományrendszerünk (mint például a rendszer indításához használt gyökér partíció), ami már majdnem betelt, akkor egy nagyobb frissítés, például a `make installworld` parancs kiadása, során az állományrendszer egyszerűen kifogy a helyből és így a frissítés meghiúsul.

11.12.2.1. Bővebben a Soft Updates működéséről

Két hagyományos megközelítés létezik az állományrendszerek metaadatainak visszaírására. (A metaadatok módosításakor olyan nem adatot tartalmazó blokkok változnak meg, mint például az állományokra vonatkozó információk vagy a könyvtárak.)

Eredetileg alapértelmezés szerint a metaadatok változásait szinkron módon írták ki. Amikor egy könyvtár megváltozott, a rendszer egészen addig várt, amíg ez a változás a lemezre nem íródott. Ugyanakkor az állományok adatait tartalmazó pufferek (az állományok tartalma) átkerültek a puffert gyorsítótárba, hogy majd később, aszinkron módon kerüljenek kiírásra. Ennek az implementációnak a biztonságos működés volt az előnye, mivel így a metaadatok még akkor is konzisztens állapotban maradtak, amikor valamilyen hiba következett be. Tehát egy állomány vagy teljesen létrejött vagy egyáltalán nem. Ha az állományhoz tartozó blokkok már nem tudtak kijutni a gyorsítótárból az összeomlás ideje előtt, akkor az `fsck(8)` felismerte ezt a helyzetet és az állományrendszer ilyen jellegű hibáját úgy orvosolta, hogy az adott állomány méretét nullára állította. Ezenkívül még az implementációs részletek is tiszták és egyszerűek maradtak. Ennek viszont hátránya, hogy a metaadatok kezelése lassú. Ha például kiadunk egy `rm -r` parancsot, akkor az a könyvtárban levő állományokat szekvenciálisan dolgozza fel, de minden egyes változtatást (az állományok törlését) csak szinkron módon rögzíti a lemezre. Ezek a frissítések érintik magát a könyvtárat, az állományokkal kapcsolatos információkat tároló táblázatot (az ún. inode táblát) és minden valószínűség szerint az állományok által lefoglalt blokkokat is közvetve. Hasonló megfontolások élnek a nagyobb könyvtárszerkezetek kibontása esetén is (`tar -x`).

A második lehetőség a metaadatok aszinkron frissítése. Ez az alapértelmezés a Linux ext2fs és BSD-k `mount -o async` opcióval csatlakoztatott UFS állományrendszerei esetén. Ilyenkor minden metaadattal kapcsolatos aktualizálás egyszerűen bekerült a puffert gyorsítótárba, tehát az állományok adatai és ezek a típusú frissítések keverednek. Ennek a megvalósításnak az az előnye, hogy nem kell megvárni, amíg a metaadatok is kiíródnak a lemezre, ezért a metaadatok óriási mennyiségű változásával járó műveletek sokkal gyorsabban hajódnak végre, mint a szinkron esetben. Sőt, maga az implementáció is tiszta és egyszerű marad, ezért a kódban megjelenő hibák beszivárgásának kockázata alacsony. A módszer hátránya, hogy egyáltalán semmilyen garanciát nem kapunk az állományrendszer konzisztenciájára. Ha tehát egy rengeteg metaadat megváltozásával együttjáró művelet közben történik valamilyen probléma (áramkimaradás, vagy valaki egyszerűen megnyomja a reset gombot), akkor az állományrendszer előre kiszámíthatatlan állapotba kerül. A rendszer újbóli indításakor ezért nincs lehetőségünk megvizsgálni az állományrendszer állapotát. Elképzelhető, hogy az állományokhoz tartozó adatok már kikerültek a lemezre, miközben a rá vonatkozó inode- vagy könyvtári bejegyzések még nem. Így lényegében lehetetlen olyan `fsck` implementációt készíteni, ami képes lenne eltüntetni ezt a káoszt (hiszen az ehhez szükséges adatok nem állnak rendelkezésre). Ha az állományrendszer helyrehozhatatlanul károsodott, akkor csak a `newfs(8)` és a biztonsági mentés visszaállítása segíthet rajta.

Ezt általában úgy küszöbölik ki, hogy az egészhez hozzáteszik még a *módosított területek feljegyzését*, amit gyakran csak *naplózásnak* (journaling) neveznek, habár ezt az elnevezést nem mindenhol ilyen értelemben használják, ezért a tranzakciók naplózásának más formáira is utalhat. A metaadatok frissítése ebben az esetben is csak szinkron módon történik, de csak a lemez egy kisebb területére. Később ez a megfelelő helyére kerül. Mivel a lemez naplózásra fordított része egy viszonylag kis méretű, folytonos terület, a lemez fejének még a megterhelőbb műveletek esetén sem kell sokat mozognia, ezért valójában ez a megoldás gyorsabb, mint a mezei szinkron frissítések. Az

implementáció bonyolultsága továbbra is jól behatárolható, a velejáró hibalehetőségek kockázata alacsony. Hátránya, hogy minden metaadat kétszer íródik ki (egyszer a naplózási területre, aztán a megfelelő helyre), ezért a hétköznapi használat során “visszaesés” tapasztalható a teljesítményben. Másrésztől azonban egy összeomlás esetén a naplózási terület segítségével minden függőben levő metaadattal kapcsolatos művelet könnyen visszafordítható vagy lezárható a rendszer következő indításakor, így ezzel egy gyors helyreállítást nyerünk.

Kirk McKusick, a Berkeley FFS fejlesztője ezt a problémát a Soft Updates segítségével hidalta át: a metaadatokkal kapcsolatos minden függőben levő frissítést a memóriában tart, majd ezeket rendezett sorrendben írja ki a lemezre (“a metaadatok rendezett frissítése”). Ennek következményeképpen a metaadatok komolyabb frissítése során a később érkező módosításoknak lehetőségük van “elkapni” a memóriában levő korábbi változataikat, ha azok még nem kerültek ki a lemezre. Így az összes, például könyvtárakon végzett, művelet a lemezre írás előtt általában először a memóriában játszódik le (az adatblokkok a pozíciójuknak megfelelően kerülnek rendezésre, ezért a rájuk vonatkozó metaadatok előtt nem jutnak ki a lemezre). Ha eközben a rendszer összeomlik, akkor így implicit módon a “napló visszalapozását” eredményezi: minden olyan művelet, ami már nem tudott kijutni a lemezre, meg nem történtnek számít. Ezen a módon az állományrendszernek egy 30 és 60 másodperc közti korábbi állapota marad fenn. Az algoritmus garantálja, hogy az összes használt erőforrás a nekik megfelelő bittérképekben helyesen jelölődik, a blokkokban és az inode-okban. Az összeomlás után az erőforrások kiosztásával kapcsolatban csak egyetlen hiba léphet fel: amikor olyan erőforrások jelölődnek “használatnak”, amelyek igazából “szabadok”. Az `fsck(8)` azonban képes felismerni ezeket a helyzeteket és felszabadítani a nem használt erőforrásokat. A `mount -f` parancs kiadásával minden további következmény nélkül figyelmen kívül hagyhatjuk az állományrendszer félkész állapotát és csatlakoztathatjuk az állományrendszereket. A használatban már nem levő erőforrások felszabadításához az `fsck(8)` parancsot később kell futtatni. Ez az alapötlet húzódik meg a *háttérben végzett lemezellenőrzés* mögött. A rendszer indításakor az állományrendszernek csupán egy *pillanatképet* rögzítjük, és az `fsck` tényleges lefuttatását későbbre toljuk. Mivel mindegyik állományrendszer csatlakoztatható “félkész” állapotban, ezért a rendszer képes elindulni többfelhasználós módban. Eközben a háttérben az `fsck` beütemezhető minden olyan állományrendszer számára, ahol arra szükség van, hogy szabadítsa fel az esetlegesen már nem használt erőforrásokat. (Így a Soft Updates opciót nem alkalmazó állományrendszerek esetén továbbra is szükség van az előtérben elvégzett `fsck` parancsra.)

A módszer előnye, hogy így a metaadatokkal kapcsolatos műveletek közel olyan gyorsak, mint az aszinkron módon végzett frissítések (tehát gyorsabb, mintha *naplózánánk*, ami ugye minden metaadatot kétszer ír ki). A hátránya a bonyolultabb kód (ami miatt növekszik az olyan hibák lehetősége, amelyek érzékenyen befolyásolhatják a felhasználói adatok elvesztését) és a nagyobb memóriaigény. Ezenkívül még van néhány olyan egyéni jellemzője, amelyet meg kell szokni. A rendszer összeomlása után az állományrendszer valamivel “régebbi” lesz. Amikor pedig megszokott szinkron megközelítés szerint az `fsck` lefutása után nulla méretű állományok jönnének létre, ezek az állományok a Soft Updates esetén egyáltalán meg sem jelennek, mivel sem a rájuk vonatkozó metaadatok, sem pedig a tartalmuk nem került ki a lemezre. Egy `rm` lefuttatása után a lemezterület addig nem kerül felszabadításra, amíg a frissítések teljesen rá nem kerülnek a lemezre. Ez nagyobb mennyiségű adat telepítésekor gondokat okozhat egy olyan állományrendszeren, ahol nincs elegendő hely az állományok kétszeri tárolására.

11.13. A rendszermag korlátainak finomhangolása

11.13.1. Az állományok és a futó programok korlátozásai

11.13.1.1. `kern.maxfiles`

A `kern.maxfiles` értéke a rendszerünk igényeinek megfelelően növelhető vagy csökkenthető. Ez a változó adja

meg a rendszerünkben levő állományleírók maximális számát. Amikor az állományleírókat tároló táblázat megtelik, a rendszer üzenetpufferében egy `file: table is full` üzenet jelenik meg, amit a `dmesg` paranccsal tudunk megnézni.

Minden megnyitott állomány, csatlakozás vagy FIFO elhasznál egy állományleíró. Egy nagyméretű szerver könnyen felemészthet több ezernyi állományleíró attól függően, hogy milyen és mennyi szolgáltatást futtat egymás mellett.

A FreeBSD korábbi kiadásában a `kern.maxfiles` a rendszermag beállításait tartalmazó állomány `maxusers` (a rendszerben egyszerre jelenlevő felhasználók maximumának) értékéből származott, tehát a `kern.maxfiles` a `maxusers` értékével arányosan növekszik. Amikor készítünk egy saját rendszermagot, mindig érdemes a rendszerünk használatának megfelelően beállítani ezt az értéket, mivel a rendszermag ebből a számból határozza meg a legtöbb előre meghatározott korlátait. Mivel még egy komoly szerveren sem jelentkeznek be egyszerre 256 felhasználónál többen, nagyjából ugyanannyi erőforrásra van szüksége, mint egy nagyobb webszervernek.

A `kern.maxusers` értéke a rendelkezésre álló memóriának megfelelően magától méreteződik a rendszer indításakor, és amit futás közben csak a `kern.maxusers sysctl` változó írásvédett értékének lekérdezéséből tudhatunk meg. Egyes oldalak üzemeltetése a `kern.maxusers` így megállapított értékétől nagyobb vagy éppen kisebb igényel, ezért a betöltéskor minden gond nélkül át lehet állítani 64, 128 vagy 256 értékűre. Senkinek sem ajánljuk, hogy 256 felé menjen, hacsak tényleg nincs szüksége ekkora mennyiségű állományleíróra. A `kern.maxusers` függvényében beállított alapértelmezett értékek tetszőleges módon átállíthatók a rendszer indításakor vagy futás közben a `/boot/loader.conf` módosításával (az ide kapcsolódó javaslatokról bővebben lásd a `loader.conf(5)` man oldalt vagy a `/boot/defaults/loader.conf` állományt) illetve a leírás más részén megadott módok szerint.

A korábbi kiadásokban úgy lehet önszabályozóra állítani a `maxusers` beállítást, ha explicit módon 0 értéket adtunk meg neki ¹. A `maxusers` paraméter beállításakor érdemes legalább 4-et megadni, különösen akkor, ha használjuk az X Window Systemet vagy szoftvereket fordítunk le. Azért van erre szükség, mert a `maxusers` értéke által szabályozott legfontosabb mennyiség az egyszerre futtatható programok táblázatának maximális mérete, amelyet így számolunk ki: $20 + 16 * \text{maxusers}$. Tehát ha a `maxusers` értékét 1-re állítjuk be, akkor az előbbi képlet értelmében csak 36 programunk futhat egymással párhuzamosan, beleértve mindazt a kb. 18 programot, amelyek a rendszerrel együtt indulnak, illetve még azt a további 15 programot, amelyeket az X Window System használatával indítunk el. Még egy olyan egyszerű dolog is, mint például egy man oldal megnézése, legalább kilenc programot indít el a szűréshez, kitömörítéshez és megnézéshez. Azonban ha a `maxusers` értékét 64-re állítjuk, akkor egyszerre akár már 1044 programot futtathatunk, ami szinte mindenre elegendő. Ha persze egy új program indításakor kapunk egy `proc table full` típusú üzenetet vagy nagy számú konkurens felhasználóval futtatunk szervert (ilyen például az `ftp.FreeBSD.org`), akkor érdemes növelni ezt a számot és újrafordítani a rendszermagot.

Megjegyzés: A `maxusers` *nem* korlátozza a számítógépre egyszerre bejelentkezni képes felhasználók számát. Egyszerűen csak beállítja néhány táblázat méretét és az egyszerre futtatható programok mennyiségét a rendszert egyidejűleg használni kívánó felhasználók maximális számának figyelembevételével.

11.13.1.2. kern.ipc.somaxconn

Az `kern.ipc.somaxconn sysctl` változó a beérkező TCP kapcsolatokat fogadó sor hosszát határozza meg. Ennek az alapértelmezett értéke 128, ami az új kapcsolatok megbízható kezeléséhez általában kevés egy erősen leterhelt webszerver számára. Ilyen helyzetekben ezt az értéket javasolt 1024-re vagy még annál is nagyobbra állítani. Az egyes szolgáltatások démonai ugyan szintén korlátozni szokták a fogadósoruk méretét (például a `sendmail(8)` vagy

az **Apache**), de gyakran találunk a beállításai között olyat, amivel ennek a sornak a mérete növelhető. A nagyobb fogadósorok mellesleg jó szolgálatot tesznek a Denial of Service (DoS) típusú támadásokkal szemben is.

11.13.2. Hálózati korlátozások

A rendszermag `NMBCLUSTERS` nevű beállítása szab határt a rendszer részére elérhető memóriapufferek mennyiségének. Egy nagyobb forgalmú szerver esetén a pufferek alacsony száma gátat szabhat a FreeBSD képességeinek. Minden klaszter nagyjából 2 KB memóriát takar, így az 1024-es érték azt jelenti, hogy a rendszermag memóriájából 2 megabyte-ot fordítunk a hálózati pufferekre. Egyszerűen kiszámítható, mennyire is van szükségünk: ha van egy webszerverünk, amely egyszerre legfeljebb 1000 párhuzamos kapcsolatot fogad, és minden kapcsolat lefoglal 16 KB-ot a fogadó-, valamint újabb 16 KB-ot a küldőpuffer számára, akkor megközelítőleg 32 MB-nyi hálózati pufferre lesz szükségünk a webszerver hatékony működéséhez. Ezt az értéket gyakran még érdemes megszorozni kettővel, így $2 \times 32 \text{ MB} / 2 \text{ KB} = 64 \text{ MB} / 2 \text{ KB} = 32768$. Több memóriával rendelkező számítógépek esetén egy 4096 és 32768 közti értéket javasolunk. Semmilyen körülmények között ne adjunk meg ennél nagyobb értéket, mert ezzel a rendszer már az indítása során összeomolhat. A `netstat(1)` `-m` beállításával ellenőrizhetjük a hálózati klaszterek kihasználtságát.

A `kern.ipc.nmbclusters` változó értékét a rendszer indításakor érdemes megváltoztatni. A FreeBSD korábbi változataiban ehhez a rendszermag `NMBCLUSTERS` nevű `config(8)` paraméterének módosítására van szükségünk.

Az olyan forgalmasabb szervereken, ahol sokat használják a `sendfile(2)` rendszerhívást, szükségünk lehet a `sendfile(2)` által használt pufferek számának növelésére a rendszermag `NFSBUFS` nevű konfigurációs paraméterén vagy a `/boot/loader.conf` állományon keresztül (lásd `loader(8)`). Amikor a futó programok közül sokan vannak `sfbufa` állapotban, akkor az egyértelműen annak a jele, hogy ezen a paraméteren állítanunk kell. A `kern.ipc.nsfbufs` egy írásvédett változót, amelyet a rendszermag állít be. Ez a paraméter névlegesen a `kern.maxusers` változó értékének megfelelően változik, de bizonyos esetekben ettől függetlenül önállóan kell behangolni.

Fontos: Annak ellenére, hogy egy socketet blokkolásmentesnek jelöltünk meg, a `sendfile(2)` meghívása egy blokkolásmentes socketre blokkolódást eredményezhet egészen addig, amíg a használatához elegendő `struct sf_buf` struktúra össze nem gyűlik.

11.13.2.1. `net.inet.ip.portrange.*`

A `net.inet.ip.portrange.*` `sysctl` változók vezérlik a TCP és UDP csatlakozásokhoz automatikusan hozzárendelt portszámok tartományát. Három ilyen tartomány létezik: az alsó, az alapértelmezett és a felső tartomány. A legtöbb hálózati program a `net.inet.ip.portrange.first` és `net.inet.ip.portrange.last` változók által rendre az 1024-től 5000-ig kijelölt alapértelmezett tartományt használja. A kimenő kapcsolatok is rögzített porttartományokat követnek, és adott körülmények mellett be lehet állítani úgy a rendszerünket, hogy ezen kívül osszon ki portokat. Ez a legtöbbször akkor fordul elő, amikor egy erősen leterhelt webproxyt működtetünk. A porttartományok nem okoznak gondot olyan szervereknél, ahol általában bejövő kapcsolatokra lehet számítani, tehát például webszerverek esetén, vagy ahol korlátozott a kimenő kapcsolatok száma, mint például a levelek továbbításánál. Ha olyan helyzetbe keverednénk, ahol már kifutunk a felhasználható portokból, a `net.inet.ip.portrange.last` mérsékelt növelésével javasolt kitörni. Ilyenkor a 10000, 20000 vagy 30000 értékek elfogadhatóak. Amikor megváltoztatjuk a porttartományok határait, előtte mindig gondoljuk át, milyen hatással lehet ez a tűzfalra. Egyes tűzfalak blokkolhatnak bizonyos tartományokat (általában az alacsonyabbakat) és

arra számítanak, hogy a rendszerek a kimenő kapcsolatokhoz a nagyobb számú portokat használják — ebből kifolyólag nem ajánlott csökkenteni a `net.inet.ip.portrange.first` értékét.

11.13.2.2. A TCP sávszélesség-késleltetés szorzat

A TCP sávszélesség-késleltetés szorzat korlátozása hasonlít a NetBSD-ben megtalálható TCP/Vegas implementációhoz. A `net.inet.tcp.inflight.enable` sysctl változó 1-re állításával lehet engedélyezni. A rendszer ilyenkor minden egyes kapcsolathoz megpróbálja kiszámítani a sávszélesség-késleltetés szorzatot és az optimális átviteli sebesség fenntartásához illeszkedően korlátozni a hálózat felé küldött adatok sorának hosszát.

Ez a lehetőség még olyankor bizonyulhat hasznosnak, amikor modemem, Gigabit Etherneten vagy nagysebességű WAN (vagy bármilyen más nagy sávszélesség-késleltetés szorzattal bíró) összeköttetéseken keresztül küldünk át adatokat, különösen abban az esetben, amikor ablakméretezést is használunk vagy nagy küldési ablakot állítottunk be. Az engedélyezésekor ne felejtjük el `net.inet.tcp.inflight.debug` változót sem beállítani 0-ra (amivel így kikapcsoljuk a nyomkövetést), éles használat esetén pedig előnyös lehet a `net.inet.tcp.inflight.min` változót legalább 6144-re állítani. Azonban hozzátesszük, hogy összeköttetéstől függően a nagy minimum értékek tulajdonképpen kikapcsolják a sávszélességkorlátozást. Ez a korlátozási lehetőség csökkenti a közbenső út adatainak és csomagváltásokhoz tartozó soroknak a méretét, miközben csökkenti a helyi számítógép felületén felépülő sorok méretét is. Ha kevesebb csomagot rakunk be a sorba, akkor az interaktív kapcsolatok, különösen a lassabb modemek esetében, kisebb *körbejárési idővel* (Round Trip Time) működnek. Továbbá megemlítenénk, hogy ez a lehetőség csak az adatok küldésére (feltöltésre, szerveroldalra) van hatással. Semmilyen befolyása nincs az adatok fogadására (letöltésre).

A `net.inet.tcp.inflight.stab` állítgatása *nem* ajánlott. A paraméter értéke alapértelmezés szerint 20, ami legfeljebb 2 csomag hozzáadását jelenti a sávszélesség-késleltetés szorzat ablakának kiszámításakor. Erre a kiegészítő ablakra azért van szükség, hogy stabilizálni tudjuk vele az algoritmust és javítani tudjuk a változó feltételekre adott reakciót, de lassabb összeköttetések esetében nagyobb ping időket is eredményezhet (habár ezek még így kisebbek, mint ha nem használnánk az algoritmust). Ilyen esetekben megpróbálhatjuk 15-re, 10-re vagy esetleg 5-re visszavenni a paraméter értékét, de ekkor a kívánt hatás eléréséhez minden bizonnyal a `net.inet.tcp.inflight.min` értékét is redukálunk kell majd (például 3500-ra). Ezen paraméterek megváltoztatását csak végső esetben ajánljuk!

11.13.3. Virtuális memória

11.13.3.1. kern.maxvnodes

A vnode egy állomány vagy könyvtár belső ábrázolása. Ennek megfelelően a vnode-ok számának növelésével az operációs rendszer spórolni tud a lemezműveletekkel. Ezt általában maga az operációs rendszer szabályozza, és nincs szükség a finomhangolására. Néhány esetben, amikor a lemezműveletek jelentik a rendszerben a szűk keresztmetszetet és kezdenek elfogyni a vnode-ok, szükség lehet ennek a számnak a növelésére. Ehhez az inaktív és szabad fizikai memória mennyiségét kell számításba vennünk.

Így kérhetjük le a pillanatnyilag használatban levő vnode-ok mennyiségét:

```
# sysctl vfs.numvnodes
vfs.numvnodes: 91349
```

Így tudhatjuk meg a vnode-ok maximális számát:

```
# sysctl kern.maxvnodes
kern.maxvnodes: 100000
```

Ha a vnode-ok aktuális kihasználtsága megközelíti a csúcértéket, nagyjából ezerrel javasolt megnövelni a `kern.maxvnodes` értékét. Ezután figyeljük továbbra is a `vfs.numvnodes` változását. Ha ismét felkúszik a maximális értékre, akkor növeljük megint egy keveset a `kern.maxvnodes` értékén. Eközben a `top(1)` használatával figyelhetjük a memória kihasználtságának növekedését is, ilyenkor tehát több memóriának kell használatban lennie.

11.14. A lapozóterület bővítése

Nem számít, mennyire tervezzük jól előre, mindig előfordulhat, hogy a rendszerünk mégsem teljesíti a kitűzött elvárásokat. Amennyiben további lapozóterület hozzáadására lenne szükségünk, azt igen könnyen megtehetjük. Háromféleképpen növelhetjük a lapozásra szánt területet: hozzáadunk a rendszerhez egy újabb merevlemez meghajtót, NFS-en keresztül lapozunk, vagy egy már meglévő partíción hozunk létre lapozóállományt.

A lapozóterület titkosításával, valamint annak lehetőségeivel és okaival kapcsolatban lapozzuk fel a kézikönyv 18.17 Szakaszát.

11.14.1. Lapozás egy új merevlemezre

A lapozóterület bővítésének legjobb módja természetesen remek indok egy új merevlemez beszerzésére is. Elvégre egy merevlemez mindig fel tudunk ilyen célra használni. Ha ezt a megoldást választjuk, előtte ajánlott (újra) elolvasni a kézikönyv 11.2 Szakaszában a lapozóterületek elrendezésére vonatkozó javaslatokat.

11.14.2. Lapozás NFS-en keresztül

NFS-en keresztül csak akkor lapozunk, ha ezt helyi lemezek segítségével nem tudjuk megtenni. Az NFS alapú lapozás hatékonyságát erősen behatárolja a rendelkezésre álló hálózati sávszélesség és további terheket ró az NFS szerverünkre is.

11.14.3. Lapozóállományok

Lapozóállománynak egy adott méretű állományt hozunk létre. Ebben a példában erre egy `/usr/swap0` nevű, 64 MB méretű állományt fogunk használni. Természetesen bármilyen más nevet is választhatunk.

Példa 11-1. Lapozóállomány létrehozása FreeBSD-ben

1. Győződjünk meg róla, hogy a rendszermagunk beállításai között megtalálható a memórialemez meghajtójának (`md(4)`) használata. Ez a `GENERIC` rendszermag alaphoz tartozik.


```
device md # Memória "lemezek"
```
2. Hozunk létre egy lapozóállományt (`/usr/swap0`):


```
# dd if=/dev/zero of=/usr/swap0 bs=1024k count=64
```
3. Állítsuk be rá a megfelelő engedélyeket (`/usr/swap0`):


```
# chmod 0600 /usr/swap0
```

4. Adjuk meg a lapozóállományt az `/etc/rc.conf` állományban:

```
swapfile="/usr/swap0" # Állítsuk be swapfile értékét, ha külső lapozóállományra van szükségű
```

5. Indítsuk újra a számítógépünket, vagy a lapozóállomány azonnali használatba vételéhez írjuk be:

```
# mdconfig -a -t vnode -f /usr/swap0 -u 0 && swapon /dev/md0
```

11.15. Energia- és erőforrásgazdálkodás

Írta: Hiten Pandya és Tom Rhodes.

Fontos a hardveres erőforrásaink hatékony kihasználása. Az ACPI megjelenése előtt az operációs rendszerek csak nehézkesen és rugalmatlanul tudták kezelni a rendszer energiafelhasználási és hőszabályzási lehetőségeit. A hardvert a BIOS kezelte, ezért a felhasználó kevesebbet tudott látni és irányítani az energiagazdálkodási beállításokból. Az *Fejlett energiagazdálkodás (Advanced Power Management, APM)* ehhez nyújtott egy erősen korlátozott felületet. Napjaink operációs rendszereiben az energia- és erőforráskezelés az egyik legfontosabb alkotóelem. Például, ha az operációs rendszerrel folyamatosan figyelni akarjuk a rendszer hőmérsékletének váratlan növekedését (és erről figyelmeztetést kérni).

A FreeBSD kézikönyvének ezen szakaszában az ACPI-ről adunk egy átfogó áttekintést, a végén pedig összefoglaljuk a témához tartozó irodalmat.

11.15.1. Mi az ACPI?

A speciális energia- és konfigurációs illesztő felület (Advanced Configuration and Power Interface, avagy ACPI) gyártók egy csoportja által létrehozott szabvány, amely a hardveres erőforrások és az energiagazdálkodás egységes felületét rögzíti (innen a neve). Döntő szerepet játszik a *Beállítások és az energiagazdálkodás operációs rendszerek áltai vezérlésében*, vagyis segítségével az operációs rendszer még nagyobb mértékben és rugalmassággal tudja irányítani ezeket a lehetőségeket. A modern operációs rendszerek az ACPI felbukkanásával "kitolták" a jelenleg meglevő Plug and Play felületek korlátait. Az ACPI az APM közvetlen leszármazottja.

11.15.2. A Fejlett energiagazdálkodás (APM) hiányosságai

A *Fejlett energiagazdálkodás (APM)* a rendszer által felhasznált energiát annak elfoglaltsága alapján vezérli. Az APM-et támogató BIOS-t a (rendszer) gyártó állítja elő és az adott hardverplatformra jellemző. Az APM operációs rendszerben levő meghajtója hozzáférést biztosít az *APM szoftveres felületéhez*, amivel lehetőség nyílik az energiaszintek kezelésére. Az APM-et 2000 előtt és körül még mindig használták egyes rendszerek gyártásánál.

Az APM használata négy nagyobb gondot rejt magában. Először is, az energiagazdálkodást a (gyártófüggő) BIOS végzi el, és az operációs rendszernek erről semmilyen ismerete nincsen. Ennek egyik példája az, amikor a felhasználó az APM-et ismerő BIOS-ban beállítja a merevlemezek automatikus kikapcsolásának idejét, majd amikor ez letelik, a BIOS az operációs rendszer tudta nélkül egyszerűen leállítja a lemezt. Másodszor: az APM működését a BIOS-ban programozták le, és teljesen az operációs rendszer hatáskörén túl tevékenykedik. Ez azt jelenti, hogy a felhasználó csak úgy tudja korrigálni az APM-es BIOS-ok problémáit, ha frissíti az alaplapi ROM-ot. Ez viszont egy nagyon kockázatos folyamat, amelynek hibája révén a rendszerünk helyrehozhatatlan állapotba kerülhet. Harmadszor: az APM alapvetően egy gyártófüggő megoldás, ami azt vonja maga után, hogy sok az átfedés (ugyanazt valósítják meg több módon), és ha az egyik gyártó BIOS-ában hibát találnak, akkor a másikéban az nem feltétlenül javítható. Végül,

de nem utolsósorban, az APM alapú BIOS-okban nincs elég hely az igazán kifinomult energiagazdálkodási sémák vagy bármi más kialakítására, amivel a felhasználók képesek lennének az igényeikhez alakítani a számítógépet.

A *Plug and Play BIOS (PNPBIOS)* sok szempontból megbízhatatlannak bizonyult. A PNPBIOS ráadásul egy 16 bites megoldás, ezért az operációs rendszereknek 16 bites emulációt kell használniuk a PNPBIOS eszközeinek "eléréséhez".

A FreeBSD APM meghajtójának dokumentációját az `apm(4)` man oldalon találjuk.

11.15.3. Az ACPI beállítása

Az `acpi.ko` meghajtó alapértelmezés szerint a `loader(8)` segítségével töltődik be, és *ne* is fordítsuk bele a rendszermagba. Ezt azzal tudnánk magyarázni, hogy modulokkal könnyebb dolgozni, például ha a rendszermag újrafordítása nélkül egy másik `acpi.ko` modult akarunk használni. Ezzel a lényegében a tesztelés is egyszerűbbé válik. Másik magyarázat, hogy a rendszer ACPI támogatása nem minden esetben működik rendesen. Ha a rendszer indítása során valamilyen problémát tapasztalunk, akkor próbálkozzunk meg az ACPI kikapcsolásával. Ezt a meghajtót nem lehet és nem is szabad kidobni a memóriából, mivel a hardverrel a rendszerbuszon keresztül tartja a kapcsolatot. Az ACPI a `hint.acpi.0.disabled="1"` sor megadásával kapcsolható a `/boot/loader.conf` állományban vagy a `loader(8)` parancssorában.

Megjegyzés: Az ACPI és az APM nem használató egyszerre. Közülük a később betöltött magától kilép, ha észreveszi, hogy a másikuk már működésbe lépett.

Az ACPI és az `acpicnf(8)` használatával a rendszerünk készenléti módba helyezhető az `-s` valamint az `1-5` paraméterek megadásával. Ezek közül is a legtöbb felhasználó számára csak az `1` vagy a `3` (állapot mentése a fizikai memóriába) érdekes. Az `5` opció egy szoftveres kikapcsolást eredményez, ehhez hasonlóan:

```
# halt -p
```

A további opciók a `sysctl(8)` man oldaláról érhetőek el. Ezen kívül még olvassuk el az `acpi(4)` és `acpicnf(8)` man oldalakat is.

11.16. A FreeBSD ACPI támogatásának használata és nyomkövetése

Írta: Nate Lawson. Segítségére volt még: Peter Schultz és Tom Rhodes.

Az ACPI az eszközök felderítésének, energiagazdálkodásának és a korábban a BIOS által kezelt hardverek szabványosított hozzáféréseinek alapjaiban új módja. Az ACPI folyamatosan fejlődik, de útját az egyes alaplapon *ACPI Machine Language (AML)* bytekód implementációjában megjelenő hibák, a FreeBSD rendszermag alrendszerének befejezetlensége és az Intel ACPI-CA értelmezőjében levő hibák lassítják.

Ez a leírás azzal a szándékkal készült, hogy segítsünk a felhasználóknak megtalálni az általuk tapasztalt problémák gyökerét és ezzel segíteni az ACPI fejlesztőket a nyomkövetésében és kijavításában. A fejlesztők köszönik, hogy ezt elolvassuk és segédkezünk a rendszerünkkel kapcsolatban felmerülő problémák orvosolásában!

11.16.1. A nyomkövetési információk beküldése

Megjegyzés: Mielőtt beküldenénk bármilyen problémát is, gondoskodjunk róla, hogy a BIOS-unk, és ha lehetséges, akkor a beágyazott vezérlők, legfrissebb verzióját használjuk.

Megkérnénk azokat, akik hibát akarnak bejelenteni, hogy a következő információkat küldjék a `frebsd-acpi@FreeBSD.org` (`mailto:frebsd-acpi@FreeBSD.org`) címre:

- A hibás működés leírása, beleértve a rendszer típusát és gyártmányát, illetve minden olyat, aminek köze lehet a hibához. Ha eddig még nem tapasztaltuk, igyekezzünk minél pontosabban leírni a hiba keletkezésének folyamatát.
- A `boot -v` paranccsal indított rendszer `dmesg(8)` kimenetét, beleértve a vizsgálni kívánt hiba által okozott összes hibaüzenetet.
- A `boot -v` paranccsal és az ACPI használata nélkül indított rendszer `dmesg(8)` kimenete abban az esetben, ha ez segít megoldani a problémát.
- A `sysctl hw.acpi` parancs kimenete. Ezzel egyébként kitűnően kideríthető, milyen lehetőségeket is kínál fel a rendszerünk.
- Az általunk használt *ACPI forrásnyelvének* (ACPI Source Language, ASL) elérhetősége az interneten. Mivel ezek akár igen nagyok is lehetnek, ezért a listára közvetlenül ne küldjünk ASL kódokat! Az ASL másolatát az alábbi parancs kiadásával hozhatjuk létre:

```
# acpidump -dt > név-rendszer.asl
```

(Adjuk meg a *név* helyett a bejelentkezéshez használt nevünket, a *rendszer* helyett pedig a gyártót/típust. Például: `njl-FooCo6000.asl`)

Habár a legtöbb fejlesztő a FreeBSD-CURRENT levelezési lista

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>)t figyeli, a problémáink leírását mindenképpen a `frebsd-acpi` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>) listára küldjük, hogy biztosan észrevegyék. A fejlesztők azt kérik, hogy legyünk türelmesek, hiszen emellett mindannyian teljes állásban is dolgoznak. Ha az általunk felfedezett hiba nem teljesen egyértelmű, akkor a fejlesztők valószínűleg meg fognak kérni arra, hogy a `send-pr(1)` használatával hozzunk róla létre egy hivatalos hibajelentést. A hibajelentés készítésekor lehetőleg a fentebb megadott információkat ugyanúgy adjuk meg. Ez segít a probléma szemmel tartásában és elhárításában. Az `frebsd-acpi` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>) lista kihagyása nélkül közvetlenül ne küldjünk hibajelentést, mivel a hibabejelentő rendszert elsősorban emlékeztetőnek használjuk, nem pedig a hibák tényleges bejelentésére. Gyakran előfordul, hogy valaki korábban már találkozott az adott problémával.

11.16.2. Háttér

Az ACPI minden olyan modern számítógépben megtalálható, mely megfelel az ia32 (x86), ia64 (Itanium) vagy amd64 (AMD) architektúrának. A teljes szabvány rengeteg lehetőséget biztosít, többek közt a processzor teljesítményének kezelését, az energiaszintek vezérlését, hőzónákat, különféle akkumulátor rendszereket, beágyazott vezérlők és a buszok felsorolását. A legtöbb rendszer általában nem a teljes szabványt valósítja meg. Például egy asztali rendszer általában csak a buszok felsorolásával kapcsolatos részeket tartalmazza, miközben egy laptop felajánlhatja a hűtés és az akkumulátor kezelését is. A laptopokban gyakorta találunk készenléti üzemmódot a maguk elbonyolított formájában.

Egy ACPI-nak megfelelő rendszert számos összetevő alkot. A BIOS-ok és chipkészletek gyártói a memóriában egy előre rögzített ponton helyeznek bizonyos táblázatokat (például FADT), amelyekkel megadják például az APIC összerendeléseit (ezt az SMP rendszerek használják), a konfigurációs regisztereket és az egyszerűbb konfigurációs értékeket. Itt ezenkívül még bytekódok egy táblázata (amit *Differenciált rendszerleíró táblának*, Differentiated System Description Table, DSDT nevezünk) is megtalálható, ahol az eszközök és módszerek nevei szerepelnek faszertű elrendezésben.

Az ACPI-hoz tartozó meghajtónak képesnek kell lennie értelmezni ezeket a rögzített táblázatokat, implementálni egy bytekód-értelmezőt, módosítani az eszközmeghajtókat és a rendszermagot az ACPI alrendszerből érkező információk befogadásához. A Linuxszal és a NetBSD-vel közösen a FreeBSD kapott egy ilyen értelmezőt az Inteltől (ACPI-CA). Az ACPI-CA forráskódja a rendszer forrásai között, a `src/sys/dev/acpica` könyvtárban található. A `src/sys/dev/acpica/0sd` könyvtárban található források pedig lehetővé teszik, hogy az ACPI-CA működhessen FreeBSD-n. Végezetül, az ACPI eszközöket megvalósító meghajtók a `src/sys/dev/acpica` könyvtárban találhatók.

11.16.3. Gyakori problémák

Az ACPI megfelelő működéséhez minden alkotórésznek helyesen kell működnie. A most következőkben előfordulásuk gyakorisága szerint felsorolunk néhány ismert problémát, valamint a hozzájuk tartozó javításokat vagy elkerülésük módszerét.

11.16.3.1. Gondok az egerrel

Egyes esetekben felfüggesztett állapotból visszatérve az egerünk nem hajlandó működni. Ezt úgy lehet elkerülni, ha `/boot/loader.conf` állományba beírjuk a `hint.psm.0.flags="0x3000"` sort. Ha ez nem segít, akkor a fentieknek megfelelően küldjünk be egy hibajelentést.

11.16.3.2. Felfüggesztés/Folytatás

Az ACPI három (STR) állapotban képes a fizikai memória segítségével készenléti módba váltani, ezek az S1-S3, és egy állapotban használja a lemezt (STD), amelyet S4-nek hívnak. Az S5 neve a “szoftveres kikapcsolás”, ami egy olyan állapotot takar, amikor a rendszerünk áram alatt van, de még nem üzemel. Az S4BIOS állapot a BIOS segítségével a lemezre menti a rendszert, az S4OS állapotot pedig teljes egészében az operációs rendszer valósítja meg.

A rendszerünk által ismert készenléti módokat a `sysctl hw.acpi` paranccsal ellenőrizhetjük. Íme mindez egy Thinkpad esetén:

```
hw.acpi.supported_sleep_state: S3 S4 S5
hw.acpi.s4bios: 0
```

Ez azt jelenti, hogy az `acpiconf -s` parancs kiadásával kipróbálhatjuk az S3, S4OS, és S5 állapotokat. Ha az `s4bios` értéke egy (1), akkor az S4BIOS támogatása helyett az S4 OS állapotot kapjuk.

A felfüggesztés és folytatás kipróbálása során kezdjük az S1 állapottal, már amennyiben az támogatott a rendszerünkön. Ez az állapot többnyire használható, mivel nem igényel túlságosan sok támogatást a meghajtó részéről. Eddig még senki sem implementálta az S2 állapotot, de ha ezt is tudja a rendszerünk, akkor az S1-hez hasonlót nyerünk vele. A következő próba az S3 állapoté. Ez a legmélyebb STR állapot, és a hardver megfelelő újraélesztéséhez rengeteg támogatás szükségeltetik a meghajtó részéről. Ha gondjaink lennének a rendszerünk felébresztésével, nyugodtan írjunk egy levelet a [freebsd-acpi](http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi) (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi)

listára, ám a probléma gyors megoldódásában ne reménykedjünk, hiszen ehhez még temérdek meghajtón és hardveren kell tesztelni és kell dolgozni.

Felfüggesztés és folytatás esetén gyakori probléma, hogy sok eszközmeghajtó nem menti el, nem állítja vissza vagy éppen nem hozza újra rendesen működésbe az adott eszközön található firmware-t, a regisztereket vagy memóriát. Az okok felderítéséhez először érdemes a következőket kipróbálni:

```
# sysctl debug.bootverbose=1
# sysctl debug.acpi.suspend_bounce=1
# acpiconf -s 3
```

Ezzel a módszerrel tesztelni tudjuk az összes meghajtó felfüggesztési és folytatási rutinjait anélkül, hogy ténylegesen S3 állapotba helyeznénk az eszközt. Bizonyos esetekben ezzel könnyen elcsíphető a hiba (például a firmware állapotának elvesztése, watchdog time out, megállás nélküli újrapróbálkozások). A rendszer ilyenkor nem vált S3 állapotra, vagyis az eszköz nem kerül energiatakarékos állapotba, és eltérően a valós S3 állapottól továbbra is működik még abban az esetben is, amikor a szükséges felfüggesztési és folytatási rutinok teljesen hiányoznak.

Komolyabb esetben további segédeszközökre lesz szükségünk, vagyis soros portra és kábelre a soros vonali nyomkövetéshez, vagy Firewire portra és kábelre a dcons(4) használatához, valamint némi tapasztalatra a rendszermagon belüli hibakeresésben.

A problémát nagy mértékben segíti különválasztani, ha igyekszünk minél több meghajtót kivenni a rendszermagból. Ha így javul a helyzet, akkor már könnyen le lehet szűkíteni arra a meghajtóra a kört, aminek betöltésével esetleg gondok akadhatnak. Általában ilyenek a bináris meghajtók, mint például az `nvidia.ko`, az X11 megjelenítésért felelős és az USB eszközök meghajtói, miközben az Ethernet eszközök remekül szoktak működni. Ha különösebb gond nélkül képesek vagyunk betölteni és eltávolítani ezeket a meghajtókat, akkor ezt a folyamatot önállósítani is tudjuk úgy, hogy az `/etc/rc.suspend` és `/etc/rc.resume` szkriptekbe beillesztjük az ehhez szükséges parancsokat. Ezekben egyébként találunk is egy megjegyzésbe rakott példát a meghajtók betöltéséről és eltávolításáról. Ha az ébresztés után elszemetelődik a képernyő tartalma, akkor állítsuk át a `hw.acpi.reset_video` változó értékét nullára (0). Sokat segíthet meg az is, ha a `hw.acpi.sleep_delay` értékét csökkentjük vagy növeljük.

Megpróbálhatjuk azt is, hogy elindítunk egy frissebb Linux disztribúciót ACPI támogatással és ugyanazon a hardveren kipróbáljuk az általa felkínált felfüggesztési és folytatási lehetőséget. Ha Linux alatt ez megbízhatóan működik, akkor nagy a valószínűsége, hogy ez FreeBSD alatt az egyik meghajtó hibájából fakadóan nem használható. Így fokozatosan le is tudjuk szűkíteni, hogy pontosan melyikkel lehet a gond, és ezzel a fejlesztők munkáját segítjük. Megjegyeznénk, hogy az ACPI-t karbantartó fejlesztők általában nem foglalkoznak más meghajtókkal (például hangkártya vagy ATA stb.), ezért az adott meghajtóval kapcsolatos hibáról javasolt értesíteni a `freebsd-current` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) listát és a meghajtóért felelős fejlesztőt is. Ha van egy kis kedvünk és időnk, mi magunk is belebiggyeszthetünk a meghajtóba néhány `printf(3)` függvényt annak kiderítésére, pontosan hol is fagy le a folytatási funkció.

Végül megpróbálkozhatunk az ACPI kikapcsolásával is, és áttérhetünk helyette az APM használatára. Ha az APM-mel működnek a készenléti állapotok, akkor érdemes inkább azzal dolgozni, különösen a régebbi (2000 előtti) hardverek esetében. A gyártóknak eltartott egy ideig, amíg rendes ACPI támogatást voltak képesek adni, ezért a régebbi hardvereknél inkább a BIOS-nak akadnak gondjai az ACPI-val.

11.16.3.3. A rendszer lemerevedik (ideiglenesen vagy teljesen)

A legtöbb rendszer olyankor akad meg, amikor sok megszakítás elveszik, vagy amikor éppen sok megszakítás érkezik egyszerre. A chipkészleteknek számos baja származik abból, hogy a BIOS milyen módon állítja be a

rendszer indítása előtt a megszakításokat, mennyire helyes az APIC (MADT) táblázata és hogyan vezérli a *Rendszervezrlő megszakítást* (System Control Interrupt, SCI).

A megszakítás-viharok a `vmstat -i` parancs kimenetében szereplő elveszett megszakításokból azonosíthatók be, ahol keressünk rá az `acpi0` sorra. Ha ez a számláló másodpercenként kettőnél többel növekszik, akkor a megszakításaink viharba keveredtek. Ha a rendszer látszólag lefagyott, próbáljuk meg előhívni a DDB-t (konzolban a **CTRL+ALT+ESC**) és gépeljük be, hogy `show interrupts`.

A megszakítási problémákkal kapcsolatban egyetlen reményünk az APIC támogatás kikapcsolása lehet a `loader.conf` állományban a `hint.apic.0.disabled="1"` sor hozzáadásával.

11.16.3.4. Végzetes hibák

Az ACPI-vel kapcsolatos végzetes hibák viszonylag ritkák, és javításuk a legfontosabb. Ilyenkor az első teendőnk elkülöníteni a hiba reprodukálásának egyes lépéseit és (ha lehetséges) lekérni a hívási láncot. Kövessük az `options` DDB és a soros vonali konzol beállításához adott tanácsokat (lásd 26.6.5.3 Szakasz) vagy hozzunk létre egy `dump(8)` partíciót. A DDB-ben a hívási láncot a `tr` parancs segítségével kérhetjük le. Ha kézzel írjuk le láncot, akkor legalább az alsó öt (5) és a felső öt (5) sorát mindenképpen jegyezzük fel!

Ezután próbáljuk meg úgy szűkíteni a probléma lehetőségét, hogy az ACPI használata nélkül indítjuk a rendszert. Ha ezzel nincs semmi gond, akkor a `debug.acpi.disable` változó értékének megfelelő beállításával egyenként meg tudjuk figyelni az ACPI alrendszer egyes részeit. Ehhez példákat az `acpi(4)` man oldalon találunk.

11.16.3.5. Felfüggesztés vagy leállítás után elindul a rendszer

Először is próbáljuk meg a `hw.acpi.disable_on_poweroff` változó értékét 0-ra állítani a `loader.conf(5)` állományban. Ezzel távoltartjuk az ACPI alrendszert a rendszer leállítási folyamatától. Egyes rendszereknek valamilyen okból kifolyólag szükségük van itt az 1 (az alapértelmezett) értékre. Ez többnyire megoldja a problémát, amikor a rendszer váratlanul elindul a készenléti mód aktiválásakor vagy kikapcsoláskor.

11.16.3.6. Egyéb problémák

Ha más gondjaink lennének az ACPI-val (dokkoló állomásunk van, egyes eszközöket nem vesz észre stb.), akkor természetesen erről is küldjünk egy leírást a levelezési listára. Azonban vegyük figyelembe, hogy egyes problémák a ACPI alrendszer eddig még nem implementált, befejezetlen részeihez kötődnek, ezért azok megoldása még várat magára. Kérünk mindenkit, hogy legyen türelemmel és álljon készen a kiküldött javítások tesztelésére!

11.16.4. ASL, `acpidump` és IASL

A problémák leggyakoribb forrása, hogy a BIOS-gyártók rossz (vagy kifejezetten hibás!) bytekódokat adnak. Ez általában a következőhöz hasonló rendszerüzenetből derül ki:

```
ACPI-1287: *** Error: Method execution failed [\\_SB_.PCI0.LPC0.FIGD._STA] \\
(Node 0xc3f6d160), AE_NOT_FOUND
```

Az ilyen jellegű hibákat gyakran úgy lehet orvosolni, ha a BIOS-unkat frissítjük a legújabb verzióra. A legtöbb ilyen üzenet teljesen ártalmatlan, de ha vannak más problémáink is, például az akkumulátor állapota nem olvasható le, akkor először az AML környékén érdemes kutakodnunk. A bytekód, más néven AML, az ASL elnevezésű

forrásnyelvből származik. Az AML egy DSDT néven ismert táblázatban található meg. Az ASL másolatát az `acpidump(8)` paranccsal készíthetjük el. Paraméterként egyaránt adjuk meg a `-t` (megmutatja a rögzített táblák tartalmát) és `-d` (visszafejtí az AML kódokat az ASL nyelvére) kapcsolókat. A felírás pontos formátumát a `A nyomkövetési információk beküldése` című szakaszban olvashatjuk.

Elsőként próbáljuk meg újrafordítani az így nyert ASL programot és keressünk benne hibákat. A figyelmeztetések általában nyugodtan figyelmen kívül hagyhatóak, azonban a hibák olyan implementációs hibákra utalnak, amelyek akadályozzák az ACPI helyes működését. Az ASL újrafordítását az alábbi paranccsal tudjuk elvégezni:

```
# iasl saját.asl
```

11.16.5. Az ASL kijavítása

Végeredményben az a célunk, hogy az ACPI megfelelő működéséhez senkinek se kelljen hozzányúlania semmihez. Azonban még mindig szükség van BIOS-gyártók által elkövetett gyakori hibák elkerülésének kifejlesztésére. A Microsoft értelmezője (`acpi.sys` és `acpiec.sys`) nem ellenőrzi szigorúan a szabvány szerinti megfelelést, ezért számos olyan BIOS-gyártó, akik csak Windows alatt tesztelik az ACPI implementációjukat, soha nem fogják kijavítani a ASL kódjukban ejtett hibáikat. Reménykedünk, hogy folyamatosan sikerül felderíteni és dokumentálni a Microsoft értelmezője által eltűrt szabványon kívüli viselkedést és leutánozni FreeBSD alatt is, hogy így ne kelljen a felhasználóknak kézzel a saját ASL forrásaikat javítgatni. Az ebből fakadó hibákat úgy tudjuk elkerülni és segíteni a fejlesztőknek azonosítani a hozzá társuló viselkedést, hogy magunk javítjuk az ASL-ben felfedezett hibákat. Ha ez beválik, akkor küldjük el a régi és új ASL közti diff(1)-et a fejlesztőknek, akik így majd az ACPI-CA-ban ki tudnak dolgozni egy megoldást a hibás viselkedésre, ezzel a javításunk szükségtelenné válik.

Most pedig következzenek a legismertebb hibaüzenetek, az okaik és javításuk:

11.16.5.1. Operációs rendszeri függőségek

Néhány AML úgy gondolja, hogy a világ csak a különböző Windows verziókból áll. A FreeBSD-nek megadható, hogy másik operációs rendszernek adja ki magát, és ezzel talán meg is szüntethető pár hiba. Ezt a legegyszerűbb úgy tudjuk megtenni, ha a `/boot/loader.conf` állományhoz hozzáfűzzük a `hw.acpi.osname="Windows 2001"` sort, vagy itt egy olyan karakterláncot adunk meg, amit az ASL forrásban láttunk.

11.16.5.2. Hiányzó visszatérési érték

Bizonyos módszerek a szabvány szerint elvártaktól eltérően nem adnak vissza explicit módon értéket. Mivel az ACPI-CA ezt nem kezeli le, ezért a FreeBSD részéről tartalmaz egy olyan módosítást, amivel implicit módon is vissza lehet adni értéket. Ha biztosak akarunk lenni a visszaadni kívánt értékben, akkor helyezzünk el a megfelelő helyekre explicit Return utasításokat. Az `iasl` a `-f` paraméterrel kényszeríthető az ilyen ASL források lefordítására.

11.16.5.3. Az alapértelmezett AML felülbírlása

Miután módosítottuk a `saját.asl` állományunkat, így tudjuk lefordítani:

```
# iasl saját.asl
```

Az `-f` kapcsoló megadásával kikényszeríthetjük az AML létrehozását még abban az esetben is, amikor hibákat tartalmaz. Ügyeljünk rá, hogy bizonyos hibákat (például a hiányzó visszatérési értékeket) a fordító magától kikerül.

Az `iasl` alapértelmezett kimenete a `DSDT.aml` állomány. A `/boot/loader.conf` átírásával így tudjuk ezzel helyettesíteni a BIOS-unk hibás változatát (ami még mindig megtalálható a flash memóriában):

```
acpi_dsdt_load="YES"
acpi_dsdt_name="/boot/DSDT.aml"
```

Ehhez ne felejtsük el a saját `DSDT.aml` állományunkat bemásolni a `/boot` könyvtárba.

11.16.6. Nyomkövetési információk kinyerése az ACPI-ből

Az ACPI meghajtója nagyon rugalmas nyomkövetési lehetőségekkel rendelkezik. Ennek révén ugyanúgy megadhatjuk a nyomkövetni kívánt alrendszert, mint ahogy annak mélységét is. A nyomkövetni kívánt alrendszereket "rétegekként" adjuk meg, valamint ezek ACPI-CA komponensekre (`ACPI_ALL_COMPONENTS`) és ACPI hardvertámogatásra (`ACPI_ALL_DRIVERS`) bomlanak le. A nyomkövetéskor keletkező kimenet részletességét a "szintként" adjuk meg, ami az `ACPI_LV_ERROR`-tól (csak a hibák) `ACPI_LV_VERBOSE`-ig (minden) terjedhet. A "szint" itt egy bitmaszk, ezért szóközzel elválasztva egyszerre több beállítás megadható. Ha túlságosan sok üzenet érkezik a konzol üzenetpufferébe, akkor szükségünk lehet a soros konzol keresztüli nyomkövetésre is. Az összes szint és réteg az `acpi(4)` man oldalon található meg.

A nyomkövetés alapértelmezés szerint nem engedélyezett. Az engedélyezéséhez hozzá kell adnunk az `options ACPI_DEBUG` sort a rendszermagunk beállításait tartalmazó állományhoz, amennyiben a rendszermagba fordítjuk az ACPI támogatást. Ha az `/etc/make.conf` állományba írjuk bele az `ACPI_DEBUG=1` sort, akkor azt globálisan engedélyezhetjük. Ha modulként használjuk, elegendő csak a következő módon újrafordítani az `acpi.ko` modult:

```
# cd /sys/modules/acpi/acpi
&& make clean &&
make ACPI_DEBUG=1
```

Telepítsük fel a `acpi.ko` modult a `/boot/kernel` könyvtárba és állítsuk be a számunkra megfelelő szintet és réteget a `loader.conf` állományban. Az alábbi példában engedélyezzük az összes ACPI-CA komponens és az összes ACPI hardvermeghajtó (processzor, LID stb.) nyomkövetését. Csak a hibaüzeneteket írja ki részletesen.

```
debug.acpi.layer="ACPI_ALL_COMPONENTS ACPI_ALL_DRIVERS"
debug.acpi.level="ACPI_LV_ERROR"
```

Ha az általunk keresett információt egy adott esemény váltja ki (például egy felfüggesztés vagy egy ébresztés), akkor nem is fontos átírunk hozzá a `loader.conf` állományt, hanem helyette a rendszer indítása után használjuk a `sysctl` parancsot a réteg és a szint megadására akkor, amikor a rendszert felkészítjük az eseményre. A `sysctl` változókat ugyanúgy nevezték el, mint a `loader.conf` állományban található beállításokat.

11.16.7. Hivatkozások

Az ACPI-ről az alábbi helyeken találunk részletesebb információkat:

- A FreeBSD ACPI levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>)
- Az ACPI levelezési lista archívuma: <http://lists.freebsd.org/pipermail/freebsd-acpi/>
- A korábbi ACPI levelezési lista archívuma: <http://home.jp.FreeBSD.org/mail-list/acpi-jp/>

- Az ACPI 2.0 specifikációja: <http://acpi.info/spec.htm>
- A FreeBSD következő man oldalai: `acpi(4)`, `acpi_thermal(4)`, `acpidump(8)`, `iasl(8)`, `acpidb(8)`
- A DSDT nyomkövetése (angolul) (http://www.cpqlinux.com/acpi-howto.html#fix_broken_dsdt). (Példának a Compaqot hozza fel, de általánosságban véve hasznos.)

Megjegyzések

1. Az önszabályozó algoritmus a `maxusers` értékét a rendszerben található memóriának megfelelően legalább 32-re, legfeljebb 384-re állítja.

Fejezet 12. A FreeBSD rendszerindítási folyamata

12.1. Áttekintés

A számítógép indulását és a rajta található operációs rendszer betöltődését “rendszerindítási folyamatnak” nevezzük, vagy egyszerűen csak “bootolásnak”. A FreeBSD rendszerindítási folyamata nagymértékű rugalmasságot kínál a rendszer indulását követő események vezérlését illetően, legyen az a számítógépre telepített különféle operációs rendszerek egyikének kiválasztása, vagy pedig ugyanazon operációs rendszer valamelyik változatának vagy rendszermagjának kiválasztása.

Ez a fejezet részleteiben bemutatja a rendszerindításhoz kapcsolódó konfigurációs opciókat, illetve a FreeBSD bootolásának testreszabhatóságát. Ebbe minden beleértendő, ami a FreeBSD rendszermag beindulása és az eszközök keresése során történik, majd az `init(8)` elindításával zárul. Ha nem vagyunk teljesen biztosak benne, ez pontosan mikor is következik be, figyeljük, amikor a szöveg színe fehérről szürkére vált.

A fejezet elolvasása során megismerjük:

- milyen elemekből áll a FreeBSD rendszertöltő alrendszere, és ezek miként kapcsolódnak egymáshoz;
- melyek azok a FreeBSD rendszerindításában résztvevő elemeknek átadható opciók, amelyekkel vezérelhető ez a folyamat;
- a `device.hints(5)` alapjait.

Csak x86: Ez a fejezet kizárólag csak az Intel x86 típusú architektúráján futó FreeBSD rendszerindítási folyamatát mutatja be.

12.2. A rendszerindítás problémája

Az operációs rendszer elindítása a számítógép bekapcsolása után egy felettébb érdekes problémát vet fel. Definíció szerint a számítógép ugyanis egy lépést sem tud megtenni az operációs rendszer elindulása nélkül. Például nem tud programokat futtatni a lemezeiről. Eszerint ha a számítógépünk nem képes programokat futtatni a lemezeiről az operációs rendszer segítségével, viszont az operációs rendszer programjai a lemezen vannak, mégis hogyan képes elindulni maga az operációs rendszer?

Maga a probléma a *Münchhausen báró kalandjai* c. könyvben leírtakhoz hasonló. A történet szerint ugyanis a főszereplő egy mocsárban ragadt derék lovával, azonban sikerült kihúznia magát belőle a saját hajánál fogva. Ez a motívum vált a számítógépek hősorában a *rendszerbetöltés* alapjává, vagyis ahogyan betöltötték az operációs rendszereket. (Ford.: ezt az angolban *bootstrapping*nek hívják, mivel a történet angol változata szerint a csizmáján (*boot*) emelkedett ki. Ebből alakult ki később az elterjedt bootolás szó is.)

Az x86-os konfigurációkon a BIOS (Basic Input/Output System, avagy “alapvető be- és kimeneti rendszer”) felelős az operációs rendszer betöltéséért. Ehhez a BIOS először megkeresi a merevlemez egy speciális helyén található Master Boot Record-ot (MBR). A BIOS elegendő tudással rendelkezik az MBR beolvasásához és lefuttatásához, és

feltételezi, hogy az MBR majd elvégzi az operációs rendszer betöltéséhez szükséges további feladatokat, helyenként a BIOS közreműködésével.

Az MBR-ben található programkódot hívják általában *boot manager*nek, kiváltképp abban az esetben, amikor az a felhasználóval is kommunikál. Ilyenkor a boot manager többnyire további kódot tartalmaz a lemez első sávján vagy az egyik állományrendszerben. (A boot managereket néha *boot loader*nek is nevezzük, de a FreeBSD-s terminológia ezt a kifejezést a rendszerindítás egy későbbi fokozatára használja.) Népszerűbb boot managerek: **boot0** (avagy **Boot Easy**, a FreeBSD alapvető boot manager), **GRUB**, **GAG** és a **LILO**. (Ezek közül egyedül csak a **boot0** fér el az MBR-ben.)

Amennyiben merevlemezeinken csupán egyetlen operációs rendszer foglal helyet, akkor egy szabványos MBR tökéletesen megfelelő. Ez az MBR megkeresi az első indítható (más néven aktív) slice-ot a lemezen, majd lefuttatja a benne található indítókédot az operációs rendszer többi részének felélesztéséhez. Az fdisk(8) által alapértelmezés szerint telepített MBR pontosan ilyen. Ennek alapja a /boot/mbr állomány.

Ha viszont több operációs rendszert is telepítettünk a lemezeinkre, akkor egy ettől eltérő boot managert érdemes használnunk, olyat, amely képes felsorolni a rendelkezésre álló operációs rendszereket, lehetővé téve, hogy választani lehessen az indításuk között. Ezek közül kettőről esik szó a következő alfejezetekben.

A FreeBSD rendszertöltő alrendszerének fennmaradó része három fokozatra bontható. Az első fokozatot az MBR indítja el, amely pontosan eleget tud ahhoz, hogy a számítógépet egy előre megadott állapotba hozza és lefuttassa rajta a második fokozatot. A második fokozat ennél már egy kicsivel többre képes, majd ezt követi a harmadik fokozat. Ez a fokozat zárja le végül az operációs rendszer betöltésének feladatát. A munka tehát ezen három fokozat között oszlik meg, mivel a PC-szabványok komoly korlátozásokat tesznek az első, illetve második fokozatban futtatható programok méretére. Ha így fűzzük össze a feladatokat, akkor a FreeBSD számára egy sokkal rugalmasabb betöltőt kapunk.

Ezután beindul a rendszermag (más néven kernel), és nekilát a számítógépben rendelkezésre álló hardvereszközök keresésének, majd előkészíti őket a használatra. Ahogy a rendszermag beindításának folyamata véget ért, az átadja a vezérlést az init(8) nevű felhasználói programnak, amely megbizonyosodik a lemezek használhatóságáról. Az init(8) ezt követően megkezdje az erőforrások felhasználói szintű konfigurálását: csatlakoztatja az állományrendszereket, beállítja a hálózati kártyá(ka)t, és elindítja mindazon programokat, amelyeknek egy FreeBSD rendszer indulásakor futnia kell.

12.3. A boot manager és az indulás fokozatai

12.3.1. A boot manager

Az MBR-ben található programkódot, avagy boot managert, sokszor csak a rendszerindítás *nulladik fokozataként* emlegetik. Ez az alfejezet a korábban említett két boot managert tárgyalja: a **boot0**-t és a **LILO**-t.

A boot0 boot manager: A FreeBSD telepítője vagy a boot0cfg(8) által kialakított MBR alapértelmezett állapotban a /boot/boot0 állományon alapszik. (A **boot0** program nagyon egyszerű, hiszen az MBR-ben elhelyezhető kód csak 446 byte hosszúságú lehet, mert a végében még el kell férnie a slice-táblának és az 0x55AA azonosítónak.) Ha telepítettük a **boot0**-t és a lemezeinken több operációs rendszer is megtalálható, akkor a rendszerindítás során egy hasonló képet kell látnunk:

Példa 12-1. A boot0 munkában

F1 DOS

```
F2 FreeBSD
F3 Linux
F4 ??
F5 Drive 1
```

```
Default: F2
```

Más operációs rendszerek, különösen a Windows, telepítésük során felülírják a már meglevő MBR-t a sajátjukkal. Ha ez történne, vagy egyszerűen csak szeretnénk a meglevő MBR-t lecserélni a FreeBSD MBR-jével, adjuk ki a következő parancsot:

```
# fdisk -B -b /boot/boot0 eszköznev
```

ahol az *eszköznev* annak az eszköznek a neve, ahonnan a rendszert indítani szeretnénk, tehát például *ad0* az első IDE-lemez esetén, vagy *ad2* a második IDE-vezérlőn található első IDE-lemez esetén, illetve *da0* az első SCSI-lemez esetén, és így tovább. Ha testre akarjuk szabni az MBR-t, használjuk a *boot0cfg(8)*-t.

A LILO boot manager: Ezen boot manager telepítéséhez és beállításához elsőként indítsuk el a Linuxot és vegyük hozzá az alábbi sort a rendszerünkben található */etc/lilo.conf* konfigurációs állományhoz:

```
other=/dev/hdXY
table=/dev/hdX
loader=/boot/chain.b
label=FreeBSD
```

A fenti sablont kiegészítve, a linuxos konvenciók szerint adjuk meg a FreeBSD elsődleges partícióját és meghajtóját úgy, hogy az *X*-et átírjuk a linuxos meghajtó betűjelére és az *Y*-t átírjuk a Linux elsődleges partíciójának számára. Ha SCSI-meghajtót használunk, a */dev/hd* részt is át kell írunk az előbbiek mellett */dev/sd*-re. A

loader=/boot/chain.b sor elhagyható abban az esetben, ha mind a két operációs rendszer ugyanazon a meghajtón található. Ha befejeztük a módosítást, futtassuk le a */sbin/lilo -v* parancsot a változtatásaink életbe léptetéséhez. Ezt ellenőrizhetjük is a képernyőn megjelenő üzenetek alapján.

12.3.2. Az első fokozat (*/boot/boot1*) és a második fokozat (*/boot/boot2*)

Az első és a második fokozat fogalmilag ugyanannak a programnak a része, a lemezen ugyanott helyezkedik el. A tárbeli megszorítások miatt ugyan el kellett választani őket egymástól, de a telepítésük mindig egy helyre történik. A telepítő vagy a **bsdlabel** (lásd lentebb) használata során a */boot/boot* nevű kombinált állományból másolódnak ki.

Az állományrendszereken kívül találhatóak, az aktív slice első sávjában, annak első szektorától kezdődően. Ez az a hely, ahol a *boot0*, illetve a többi boot manager is keresi a rendszerindítás folytatására alkalmas programot. A felhasznált szektorok száma könnyedén kideríthető a */boot/boot* méretéből.

Legfeljebb 512 byte-os méreténél fogva a *boot1* állomány nagyon egyszerű felépítésű, és éppen csak annyit tud a slice-ra vonatkozó információkat tároló FreeBSD *bsdlabel*-ről, hogy megtalálja a *boot2*-t és elindítsa.

A *boot2* már egy kicsivel ügyesebb, és eléggé ismeri a FreeBSD állományrendszerét ahhoz, hogy megtalálja rajta állományokat, valamint képes egy egyszerű felületet nyújtani a rendszermag vagy a betöltő megválasztásához.

Mivel a betöltő pedig már ennél is okosabb, és egy könnyen használható rendszerindítási konfigurációt tud a felhasználó számára nyújtani, ezért a *boot2* általában ezt indítja el, de előtte közvetlenül a rendszermag futtatását végzi el.

Példa 12-2. A boot2 működés közben

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

Ha le kellene váltani a korábban telepített boot1 és boot2 fokozatokat, használjuk a bsdlabel(8)-t:

```
# bsdlabel -B lemezslice
```

ahol a *lemezslice* annak a lemeznek és slice-nak a kombinációja, ahonnan indítjuk a rendszerünket, például az első IDE-lemez első slice-a esetén ez az `ad0s1`.

A veszélyesen dedikált mód (Dangerously Dedicated Mode) Amikor a bsdlabel(8) meghívásakor csak a lemez nevét használjuk, például `ad0`-t, a parancs egy veszélyesen dedikált lemezt hoz létre, slice-ok nélkül! Szinte biztos, hogy nem ez az, amire szükségünk lenne, ezért mindig ellenőrizzük kiadása előtt a bsdlabel(8) parancsot!

12.3.3. A harmadik fokozat (/boot/loader)

A betöltő a három fokozatú rendszertöltés utolsó állomása. Az állományrendszerben `/boot/loader` néven található meg.

A rendszertöltőt az egyszerű konfigurálhatóságot támogató, felhasználóbarát eszköznek tervezték, és könnyen megtanulható, beépített parancsokat használ, melyek mögött egy összetettebb parancsokat ismerő, erősebb értelmező áll.

12.3.3.1. A rendszertöltő működése

Az inicializálás során a rendszertöltő megpróbálja megkeresni a konzolt, és a lemezek közül igyekszik megtalálni azt, amelyikről elindult a rendszer. A keresések eredményének megfelelően beállítja a változókat, majd elindul egy értelmező, ahol vagy szkriptből olvasva, vagy pedig interaktívan feldolgozásra kerülnek a parancsok.

A rendszertöltő ezt követően beolvassa a `/boot/loader.rc` állományt, az pedig alapértelmezés szerint feldolgozza a `/boot/defaults/loader.conf` állományt, ahol a változók értelmes kezdőértéket kapnak, valamint feldolgozza még a `/boot/loader.conf` állományt is, ahol a változók értékeit változtathatjuk meg. Miután ez lezajlott, a `loader.rc` a változók értékeinek megfelelően cselekszik, betöltve az ily módon kiválasztott rendszermagot és a hozzá választott modulokat.

Végezetül, a rendszertöltő beiktat egy, alapértelmezés szerint 10 másodperces várakozási szünetet, majd elindítja a rendszermagot, ha azt meg nem szakítjuk egy billentyű lenyomásával. Ha megszakítjuk ezt a várakozást, a rendszertöltő egy parancssort ad, amelyen keresztül egyszerű parancsokat adhatunk ki neki: állíthatjuk a változók értékeit, modulokat távolíthatunk el a memóriából, modulokat tölthetünk be, elindíthatjuk a rendszert vagy újraindíthatjuk a számítógépet.

12.3.3.2. A rendszertöltő beépített parancsai

Következzenek a leggyakrabban használt parancsok a rendszertöltőben. Az összes itt elérhető parancsot a loader(8) man oldalon találjuk meg.

autoboot másodperc

Megkezdí a rendszermag betöltését, ha nem szakítjuk meg a várakozást másodpercekben megadott időtartam alatt. Ekkor egy visszaszámlálást láthatunk, ami az alapértelmezés szerint 10 másodperctől indul.

boot [-opciók] [rendszermag]

Amennyiben léteznek, a megadott opciókkal azonnal megkezdí a megadott rendszermag betöltését. A *rendszermag* paraméter csak abban az esetben adható meg, ha előtte kiadtunk egy *unload* parancsot, máskülönbén a korábban betöltött rendszermaggal indul a rendszer.

boot-conf

Végigmegy a modulok ugyanazon automatikus konfigurációján, ahogy az a normális rendszerindítás során is történik. Ezen parancs használatának csak akkor van értelme, ha előtte az *unload* parancsot használjuk, megváltoztatunk egy-két változót, általában a *kernel*-t.

help [témakör]

A */boot/loader.help* állományban fellelhető súgóüzeneteket mutatja meg. Ha témakörnek indexet adunk meg, akkor az elérhető témakörök listáját kapjuk meg.

include állománynév ...

Feldolgozza a megnevezett állományt: beolvassa, majd sorról sorra értelmezi. Hiba esetén azonnal megállítja a feldolgozást.

load [-t típus] állománynév

A név alapján betölti a rendszermagot, modult vagy az adott típusú állományt. Az állománynév után megadott további paraméterek az állománynak adódnak át.

ls [-l] [elérési útvonal]

Kilistázza a megadott elérési útvonalon található állományokat, vagy ennek hiányában a gyökér tartalmát. Ha hozzátesszük a *-l* kapcsolót, az állományok mérete is látható válik.

lsdev [-v]

Kilistázza az összes olyan eszközt, ahonnan modulokat tölthetünk be. Amennyiben a *-v* kapcsolót is megadjuk, további részleteket tudhatunk meg róluk.

lsmod [-v]

Kilistázza a betöltött modulokat. Ha többet szeretnénk megtudni róluk, adjuk meg a *-v* kapcsolót.

more állománynév

Megmutatja a megadott állomány tartalmát, minden *LINES* számú sor után szünetet tartva.

reboot

Azonnal újraindítja a számítógépet.

set változó

set változó=érték

Beállítja a rendszertöltő környezeti változójának értékét.

unload

Eltávolítja a memóriából az összes betöltött modult.

12.3.3.3. Rendszertöltő példák

Íme néhány konkrét példa a rendszertöltő használatára:

•

Így indíthatjuk egyfelhasználós módban az általunk használt rendszermagot:

```
boot -s
```

- Távolítsuk el a betöltött rendszermagot és a moduljait, és töltsük be helyettük a korábbi (vagy egy másik) rendszermagot:

```
unload
```

```
load kernel.old
```

Itt használhatjuk a `kernel.GENERIC` nevet is, amely a telepítőlemezen található általános rendszermagra utal, vagy a `kernel.old` nevet, amely a korábban használt rendszermagot rejti (például amikor rendszermagot frissítettünk vagy készítettünk magunknak).

Megjegyzés: A következőképpen lehet betölteni a szokásos moduljainkat egy másik rendszermaggal:

```
unload
```

```
set kernel="kernel.old"
```

```
boot-conf
```

- Egy rendszermag-konfigurációs szkript (automatizált szkript, amely ugyanazokat a beállításokat végzi el, amelyeket mi magunk tennénk akkor, amikor a rendszermagot indítjuk) betöltése:

```
load -t userconfig_script /boot/kernel.conf
```

12.3.3.4. Rendszerbetöltő képernyők

Készítette: Joseph J. Barbish.

A rendszertöltés során megjelenő rendszerüzenetek megjelenítése helyett egy sokkal megnyerőbb, látványosabb rendszerindítást tudunk elérni betöltő képernyők használatával. Egy ilyen képet egészen a konzolos bejelentkezésig, vagy az X felett futó valamelyik bejelentkező képernyő megjelenéséig láthatunk.

FreeBSD alatt alapvetően két típusú környezet létezik. Ezek közül az egyik a hagyományos virtuális konzolos parancssoros felület. Ekkor a rendszertöltés befejeződésekor egy szöveges parancssori bejelentkező promptot kapunk. A másik környezet az X11 által felkínált grafikus felület. Miután telepítettük az X11 szervert és valamelyik munkakörnyezetet, tehát például a **GNOME**, a **KDE** vagy az **XFce** környezetek valamelyikét, a `startx` paranccsal indíthatjuk el a grafikus felületet.

Némely felhasználók a megszokott szöveges bejelentkezés helyett is inkább valamelyik X11 alapú grafikus bejelentkezést szeretnék használni. A különböző bejelentkező képernyők, mint amilyen az Xorg esetén az **XDM**, a **GNOME** esetén a **gdm**, vagy a **KDE** esetén a **kdm** (illetve a Portgyűjteményből származó egyéb megoldások)

alapvetően a konzolos bejelentkezés helyett nyújtanak egy grafikus bejelentkező felületet. Ilyenkor a sikeres bejelentkezést követően a felhasználó közvetlenül egy grafikus környezetbe kerül.

A parancssoros felület esetén a rendszertöltő képernyő elrejt az összes rendszerüzenetet és a rendszer indításakor futtatott programok üzeneteit. Az X11 használata esetén azonban a felhasználók ezzel együtt már a többi, alapértelmezés szerint grafikus felülettel rendelkező rendszerhez (Microsoft Windows vagy más nem-UNIX operációs rendszer) hasonló élményt nyernek.

12.3.3.4.1. A rendszerbetöltő képek támogatása

A FreeBSD csak BMP (.bmp) vagy ZSoft PCX formátumú, 256 színű rendszerbetöltő képek megjelenítését támogatja. Emellett szabványos VGA kártyákon csak akkor fog működni, ha a kép 320x200 vagy annál kisebb felbontású.

Nagyobb méretű képek esetén, egészen az 1024x768-as felbontásig, a FreeBSD VESA támogatására lesz szükségünk. Ezt vagy a rendszer indításakor a VESA modul betöltésével engedélyezhetjük, vagy ha a rendszermag konfigurációs állományában megadjuk a VESA sort és készítünk egy saját rendszermagot (lásd 8 fejezet). A VESA támogatáson keresztül a felhasználók a teljes képernyőt betöltő rendszerbetöltő képeket is meg tudnak így jeleníteni.

A rendszerbetöltő képernyő a rendszer indítása közben bármikor tetszőlegesen kikapcsolható egy tetszőleges billentyű lenyomásával.

A megadott betöltőképernyő alapértelmezés szerint a képernyővédő szerepét is betölti az X11 felületén kívül. Ha tehát egy ideig nem használjuk a számítógépünket, akkor a képernyő átvált a betöltőképre és folyamatosan változtatni kezdi az intenzitását, a nagyon világosból a nagyon sötétbe, majd újrakezdi. Az alapértelmezett képernyővédő az `/etc/rc.conf` állományban a `saver=` sor megadásával állítható át. Ehhez a beállításhoz több különböző beépített képernyővédő tartozik, ezek teljes listáját a `splash(4)` man oldalon olvashatjuk. Ezek közül az alapértelmezett a "warp". Az `/etc/rc.conf` állományban megadható `saver=` csak a virtuális konzolokra vonatkozik, az X11 bejelentkező képernyőire semmilyen hatással sincs.

A rendszerbetöltő néhány üzenete, valamint a rendszerindítási opciókat tartalmazó menü és a hozzá tartozó visszaszámlálás még a rendszerbetöltő képernyő használata során is meg fog jelenni.

A <http://artwork.freebsdgr.org> (<http://artwork.freebsdgr.org/node/3/>) címen találhatunk néhány ilyen betöltőképernyőt. A `sysutils/bsd-splash-changer` port telepítésével pedig a rendszer egyes indításakor egy előre megadott gyűjteményből tudunk véletlenszerűen választani egyet.

12.3.3.4.2. A rendszerbetöltő képek használata

A betöltőképet tartalmazó (.bmp vagy .pcx kiterjesztésű) állományt a rendszerindító partícióra, például a `/boot` könyvtárba kell tennünk.

A normál (256 szín, legfeljebb 320x200-as felbontású) képek esetén a következő sorokat adjuk hozzá a `/boot/loader.conf` állományhoz:

```
splash_bmp_load="YES"
bitmap_load="YES"
bitmap_name="/boot/betöltőkép.bmp"
```

Nagyobb felbontás esetén (legfeljebb 1024x768-as méretig) pedig a `/boot/loader.conf` állománynak a következőket kell tartalmaznia:

```
vesa_load="YES"
```

```
splash_bmp_load="YES"
bitmap_load="YES"
bitmap_name="/boot/betöltőkép.bmp"
```

Az iménti példában feltételeztük, hogy a `/boot/betöltőkép.bmp` állományt használjuk betöltőképként. Amikor azonban PCX állományokat akarunk használni, a következő sorokat kell megadnunk, a felbontástól függően a `vesa_load="YES"` sorral kiegészítve:

```
splash_pcx_load="YES"
bitmap_load="YES"
bitmap_name="/boot/betöltőkép.pcx"
```

Természetesen a kép neve sem csak “betöltőkép” lehet. Tetszőlegesen elnevezhetjük, egyedül csak arra kell ügyelnünk, hogy BMP vagy PCX formátumú legyen: `splash_640x400.bmp` vagy például `blue_wave.pcx`.

További érdekes beállítások a `loader.conf` állományból:

```
beastie_disable="YES"
```

Ennek megadásakor nem jelenik meg a rendszerindítási lehetőségeket felkínáló menü, de a visszaszámlálás megmarad. Hiába tiltjuk le a menüt, ilyenkor továbbra is választanunk kell a lehetőségek közül.

```
loader_logo="beastie"
```

Ezzel a beállítással a menüben látható “FreeBSD” feliratot cserélhetjük le a korábbi kiadásokban szereplő színes démonos emblémára.

12.4. Kapcsolat a rendszermaggal a rendszerindítás folyamán

Ahogy sikerült betölteni (a szokásos módon) a rendszertöltővel vagy (a rendszertöltő átugrásával) a boot2 segítségével, a rendszermag megvizsgálja az esetlegesen átvett rendszerindítási paramétereket, és azoknak megfelelően viselkedik.

12.4.1. A rendszermag paraméterei

A rendszermag leginkább használt paraméterei:

–a

a rendszermag inicializálása során rákérdez a gyökér állományrendszerként csatlakoztatandó eszközre.

–C

a rendszer indítása CD-ről.

–c

a UserConfig, a rendszerindítás során használt rendszermag-beállító, futtatása.

-s

a rendszer indítása egyfelhasználós módban.

-v

részletesebb információk megjelenítése a rendszermag indítása során.

Megjegyzés: Ezeken kívül még számos paraméter létezik, a teljes listát a boot(8) man oldalon találhatjuk meg.

12.5. Eszköz útmutatók (device.hints)

Írta: Tom Rhodes.

Megjegyzés: Ez a lehetőség csak a FreeBSD 5.0 vagy annál későbbi verziókban jelenik meg.

A rendszerindítás kezdeti szakaszában a loader(8) beolvassa a device.hints(5) állományt. Ebben az állományban tárolódnak a gyakran csak “eszköz útmutatóknak” nevezett változók, amelyek a rendszermag számára nyújtanak hasznos információkat az indulás során. Ezeket az “útmutatókat” az eszközmeghajtók hasznosítják az általuk ismert eszközök beállítása során.

Az eszközökre vonatkozó ilyen jellegű útmutatások a harmadik fázisban megjelenő parancssorban is megadhatóak. A változókat a set (beállít) parancs segítségével tudjuk felvenni, míg az unset (eltávolít) parancssal tudunk törölni, valamint a show (megmutat) parancssal megjeleníteni az értéküket. Sőt, ezen a ponton a /boot/device.hints állománnyal már beállított változókat is felülbírálnak. A rendszerindító parancssorában elvégzett módosítások viszont nem fognak megmaradni, és a következő rendszerindítás alkalmával elvesznek.

Ahogy a rendszerünk használatra kész állapotba került, a kenv(1) parancs használható a változók értékeinek listázásához.

A /boot/device.hints állományban soronként egy-egy változót tudunk megadni, illetve a kettőskeresztrel (“#”) bevezetve megjegyzéseket illeszthetünk bele. A sorok szerkezete az alábbi:

```
útmutató.meghajtó.egység.kulcsszó="érték"
```

A harmadik fázisban pedig így adhatjuk meg:

```
set útmutató.meghajtó.egység.kulcsszó=érték
```

Itt a meghajtó az eszközmeghajtó neve, az egység az eszközmeghajtó által kezelt egyik egység sorszáma, a kulcsszó pedig az útmutatáshoz tartozó kulcsszó. Ez a következők egyike lehet:

- `at`: az útmutatás az eszköz által használt buszra vonatkozik.
- `port`: az útmutatás az eszköz által használt I/O-címre vonatkozik.
- `irq`: az útmutatás az eszköz által használt megszakítás sorszáma vonatkozik.
- `drq`: az útmutatás az eszköz által használt DMA-csatorna sorszáma vonatkozik.

- `maddr`: az útmutatás az eszköz által használt fizikai memóriaterület kezdőcímére vonatkozik.
- `flags`: az eszközhöz tartozó bitek beállítása.
- `disabled`: ha az értéke 1, akkor az adott eszköz használatát letiltjuk.

Az eszközmeghajtók elfogadhatnak (vagy várhatnak) olyan útmutatásokat is, amelyek itt nem szerepelnek, ezért mindegyik esetében érdemes áttekinteni a hozzájuk tartozó man oldalt. Bővebb információért lásd a `device.hints(5)`, `kenv(1)`, `loader.conf(5)` és `loader(8)` man oldalakat.

12.6. Init: A folyamatirányítás elindítása

Miután a rendszermag sikeresen elindult, átadja a vezérlést az `init(8)` felhasználói folyamatnak, amely vagy az `/sbin/init`, vagy pedig a rendszerindítóban megadott `init_path` változó által mutatott program.

12.6.1. Az automatikus újraindulási folyamat

Az automatikus újraindulási folyamat gondoskodik róla, hogy az indulást követően rendelkezésre álló állományrendszerek ne legyenek sérültek. Amennyiben mégis sérültek és a `fsck(8)` nem tudja megjavítani őket, az `init(8)` a rendszert egyfelhasználós módba állítja, ahol a rendszergazdának kell közvetlenül megoldania a fennálló problémákat.

12.6.2. Egyfelhasználós mód

Ezt a módot az automatikus újraindítási folyamat során érhetjük el, vagy akkor, ha a rendszert a `-s` kapcsolóval indítjuk, esetleg a rendszerindítóban beállítjuk a `boot_single` változót.

Ezt a módot többfelhasználós módban, a `shutdown(8)` hívásával is aktiválhatjuk, ha nem adjuk meg az újraindítást (`-r`) vagy leállítást (`-h`) kérő opciók egyikét sem.

Ha az `/etc/ttys` állományban a `console` értékét `insecure` (nem biztonságos)ra állítjuk, a rendszer az egyfelhasználós módba lépés előtt kérni fogja a `root` felhasználó jelszavát.

Példa 12-3. Nem biztonságos konzol megadása az `/etc/ttys`-ben

```
# name  getty                                type    status      comments
#
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                                unknown off insecure
```

Megjegyzés: Az `insecure` (nem biztonságos) konzol az, ahol nem tekintjük megbízhatónak a rendszerkonzol fizikai biztonságát, és biztosak akarunk lenni benne, hogy csak az képes használni a rendszert egyfelhasználós módban, aki ismeri a `root` felhasználó jelszavát. Ez tehát nem arra utal, hogy magát a konzolt akarjuk nem biztonságos módban működtetni. Szóval, ha biztonságot akarunk, az `insecure`-t válasszuk, ne pedig a `secure`-t.

12.6.3. Többfelhasználós mód

Ha az `init(8)` mindent rendben talál, vagy ha a felhasználó kilépett az egyfelhasználós módból, a rendszer többfelhasználós módba lép át, ahol megkezdzi az erőforrások konfigurálását.

12.6.3.1. Az erőforrások konfigurációja (`rc`)

Az erőforrásokat konfiguráló alrendszer beolvassa a folyamathoz kapcsolódó változók alapértelmezett értékeit az `/etc/defaults/rc.conf` állományból, majd módosítja őket a rendszer egyéni beállításai szerint, amit a `/etc/rc.conf` állományból olvas ki. Ezután elvégzi az `/etc/fstab` alapján az állományrendszerek csatlakoztatását, elindítja a hálózati szolgáltatásokat, egyéb rendszerdaemonokat, és végezetül lefuttatja a telepített csomagok indítószkriptjeit.

Az erőforrásokat konfiguráló alrendszerről magáról az `rc(8)` man oldalon, valamint az érintett szkriptek tanulmányozásával tudhatunk meg többet.

12.7. A leállítási folyamat

A `shutdown(8)` paranccsal vezérelt leállítás során az `init(8)` megpróbálja lefuttatni az `/etc/rc.shutdown` szkriptet, majd ezt követően `TERM` (befejeztetés) jelzést küld az aktuálisan futó folyamatoknak, kis idő múlva pedig `KILL` (leállítás) jelzést azoknak, amelyek még nem álltak le addig a pillanatig.

Azokon az architektúrákon és rendszereken, ahol elérhető a fejlett energiagazdálkodás támogatása, a FreeBSD-t a `shutdown -p now` paranccsal állíthatjuk le, amit közvetlenül a számítógép automatikus kikapcsolása követ. A FreeBSD-s rendszer újraindításához egyszerűen csak adjuk ki a `shutdown -r now` parancsot. Fontos tudni, hogy alapértelmezés szerint a `shutdown(8)` használatához `root` felhasználónak, vagy legalább az `operator` csoport tagjának kell lennünk. Ezekre a feladatokra egyébként a `halt(8)` és `reboot(8)` parancsok is használhatóak. Alkalmazásukról bővebben a hozzájuk, valamint a `shutdown(8)`-hoz tartozó man oldalakon találhatunk bővebben információkat.

Megjegyzés: Az energiagazdálkodás használatához a rendszermagnak beépítve vagy a megfelelő modul betöltésével biztosítani kell az `acpi(4)` támogatást.

Fejezet 13. Felhasználók és hozzáférések alapvető kezelése

Írta: Neil Blakey-Milner.

13.1. Áttekintés

A FreeBSD lehetővé teszi, hogy egyazon időben egyszerre több felhasználó is dolgozhasson a számítógépen. Közülük nyilvánvalóan csak egy képes előtte ülni ¹, de rajta kívül még sok más felhasználó is be tud jelentkezni a munkájához hálózaton keresztül. A rendszer használatához minden egyes felhasználónak hozzáféréssel kell rendelkeznie.

A fejezet elolvasása során megismerjük:

- a FreeBSD rendszerben megtalálható különféle felhasználói hozzáférések közti különbségeket;
- hogyan készítsünk új felhasználói hozzáféréseket;
- hogyan töröljünk felhasználói hozzáféréseket;
- hogyan változtassuk meg a hozzáférés adatait, mint például a felhasználók teljes nevét vagy a választott parancsértelmezőt;
- hogyan korlátozzuk az egyes hozzáféréseket vagy hozzáférések egy csoportját az olyan erőforrások, mint például a memória vagy a processzoridő védelmében;
- hogyan használjuk csoportokat a hozzáférések karbantartásának megkönnyítésére.

A fejezet elolvasásához ajánlott:

- a UNIX és a FreeBSD alapjainak ismerete (3 fejezet).

13.2. Bevezetés

A rendszert bármilyen fajta módon csak hozzáféréseken keresztül tudjuk elérni, minden programot felhasználók futtatnak, ezért a felhasználók és hozzáférések kezelése a FreeBSD rendszerek szerves része.

A FreeBSD rendszerben minden hozzáférés rendelkezik bizonyos információkkal az azonosításhoz.

Felhasználó neve

A felhasználónevet a `login:` felirat megjelenésekor kell megadni. A felhasználók neveinek egyedinek kell lenni a számítógépen, tehát két felhasználó nem használhatja ugyanazt a nevet. A `passwd(5)` man oldalon megtalálhatjuk azokat a szabályokat, amelyek az érvényes felhasználónevek létrehozására vonatkoznak. Általánosságban elmondható, hogy a felhasználóneveknek kisbetűseknek kell lenniük és legfeljebb nyolc karakterből állhatnak.

Jelszó

Minden hozzáféréshez tartozik egy jelszó is. Ez a jelszó lehet akár üres is, ebben az esetben nincs szükség jelszóra a hozzáféréshez. Ez viszont többnyire nagyon rossz ötlet: minden hozzáférést erősen ajánlott jelszóval védeni.

Felhasználó azonosítója (User ID, UID)

Az UID egy szám, amely hagyományosan 0-tól 65535-ig terjed ², és a felhasználó rendszeren belüli egyedi azonosítására használatos. A FreeBSD az UID-ot a felhasználók azonosítására használja — bármelyik parancs, amely lehetővé teszi felhasználónevek megadását, át fogja alakítani UID-dé, mielőtt ténylegesen dolgozni kezdene vele. Ez tehát azt jelenti, hogy több hozzáférésünk is lehet több különböző felhasználónévvel, de ugyanazzal az UID-del. Legalább is a FreeBSD ezeket egyetlen felhasználónak tekinti, de nem is valószínű, hogy ilyenre valaha szükségünk is lenne.

Csoportazonosító (Group ID, GID)

A csoportazonosító (Group ID, GID) egy szám, amely általában 0-tól 65535-ig terjed ², és azt az elsődleges csoportot azonosítja be egyedileg, amelyikhez a felhasználó tartozik. A csoportok segítségével az erőforrások hozzáféréseinek vezérlését tudjuk megoldani a felhasználók GID-jével az UID-dek helyett. Ezzel jelentős mértékben csökkenthető egyes konfigurációs állományok mérete. Egy felhasználó egyszerre több csoport tagja is lehet.

Bejelentkezési osztály

A bejelentkezési osztályok a csoportszervezés kibővítését célozzák meg, további rugalmasságot nyújtanak, amikor a rendszert az egyes felhasználók igényeihez szabjuk.

Jelszóváltási idő

Alapértelmezés szerint a FreeBSD nem kényszeríti rá a felhasználókat, hogy rendszeresen megváltoztassák a jelszavukat. Ezt felhasználónként kikényszeríthetjük, és így az egyes, vagy akár az összes felhasználót kötelezhetjük az adott időközönként jelszóváltásra.

A hozzáférés lejáratási ideje

A FreeBSD-ben alapértelmezés szerint nem évülnek el a hozzáférések. Ha azonban olyan hozzáféréseket kell létrehoznunk, melyeknek korlátoznunk kell az élettartamukat, mint például egy iskolában a diákok számára, akkor ilyenkor meg tudjuk adni a lejáratuk idejét. Ezen dátum után a hozzáféréssel már nem lehet bejelentkezni a rendszerbe, viszont a hozzá tartozó könyvtárban tárolt állományok továbbra is megmaradnak.

Felhasználó teljes neve

Míg a felhasználónév tökéletesen azonosítja a FreeBSD számára a hozzáférést, nem feltétlenül tükrözi a felhasználó valódi nevét. Ezt az információt is meg lehet adni a hozzáféréshez.

Felhasználói könyvtár

A felhasználói könyvtár a rendszerben található azon könyvtár teljes elérési útvonala, ahová a felhasználó a bejelentkezést követően kerül. Elterjedt megszokás, hogy az összes felhasználó könyvtárát a `/home/felhasználónév` vagy a `/usr/home/felhasználónév` könyvtárba teszik. A felhasználók ezekben a könyvtárakban tárolják a személyes állományait, és tetszőleges könyvtárakat hozhatnak létre benne.

Felhasználói parancsértelmező

A parancsértelmező biztosítja azt az alapértelmezett környezetet, amelyben a felhasználó kapcsolatba tud lépni a rendszerrel. Többféle parancsértelmező is akad, és a tapasztaltabb felhasználók ragaszkodnak is némelyikükhöz, ami gyakran látható is a hozzáférésük beállításában.

Három fő típusa van a hozzáféréseknek: az adminisztrátori, a rendszer- és a felhasználói hozzáférések. Az adminisztrátori hozzáférés, amelyre gyakran `root`-ként hivatkoznak, használatos a rendszer karbantartására, és semmilyen korlátozás nem érvényes rá. A rendszerhozzáférések szolgáltatásokat futtatnak. Végezetül a felhasználói hozzáféréseket használják a valódi emberek, akik bejelentkeznek, leveleket olvasnak és így tovább.

13.3. Az adminisztrátori hozzáférés

Az adminisztrátori hozzáférés, amelyet általában csak `root`-nak nevezünk, a rendszeradminisztrációs feladatok elvégzéséhez van igazítva, és nem ajánlott az olyan hétköznapi tevékenységek elvégzéséhez, mint például a levelek olvasása és írása, a rendszer bejárása vagy a programozás.

Ezért az adminisztrátor, eltérően az átlagos felhasználói hozzáférésektől, képes mindenféle határok nélkül tevékenykedni, és az adminisztrátori hozzáférés helytelen használata látványos katasztrófákat idézhet elő. A felhasználói hozzáférések képtelenek merő véletlenségből tönkretenni a rendszert, ezért általánosságban véve az a legjobb, ha egyszerű felhasználói hozzáféréseket használunk, amint módunk van rá, hacsak nincs szükségünk kifejezetten különleges jogosultságokra.

Minden esetben érdemes alaposan megfontolni az adminisztrátorként kiadott parancsokat, mivel egyetlen hiányzó szóköz vagy más egyéb karakter helyrehozhatatlan károkat okozhat a rendszerben.

Ezért, ha még nem tettük volna meg korábban, legyen az első dolgunk a fejezet elolvasása után, hogy létrehozunk egy kiemelt jogosultságokkal nem rendelkező felhasználót saját magunk számára a hétköznapi feladatok lebonyolítására. Ez ugyanúgy vonatkozik a többfelhasználós és az egyfelhasználós módban futó rendszerekre is. A fejezet egy későbbi részében leírjuk, hogyan lehet további hozzáféréseket létrehozni, és hogyan kell váltani egy mezei felhasználó és az adminisztrátor hozzáférése között.

13.4. Rendszerhozzáférések

A rendszer általi hozzáférések azok, amelyek olyan szolgáltatások futtatásáért felelősek, mint például a DNS, a levelezés, a webserverek és így tovább. Ennek oka a biztonság: ha minden szolgáltatást adminisztrátorként futtatnánk, bármit meg tudnának tenni a rendszerben.

Ilyen rendszerfelhasználók a `daemon`, `operator`, `bind` (a névfeloldáshoz), `news`, és a `www`.

A `nobody` ("senki") egy általános jogosultságok nélküli rendszerfelhasználó. Mindazonáltal nem szabad elfelejtenünk, hogy minél több szolgáltatást bízunk a `nobody`-ra, annál több állomány és program kerül vele kapcsolatba, ennél fogva annál erősebbé válik a rendszer számára ez a felhasználó.

13.5. Felhasználói hozzáférések

A felhasználói hozzáférések a valós felhasználók elsődleges eszközei a rendszer felé, és ezek a hozzáférések szigetelik el a felhasználókat és a környezeteket, megakadályozva, hogy a felhasználók kárt okozzanak akár a

rendszerben, akár egymásnak, valamint lehetővé teszik a felhasználók számára a környezeteik testreszabását anélkül, hogy a többiekét módosítani kellene.

Minden olyan személynek, aki hozzá akar férni a rendszerünkhöz, rendelkeznie kell felhasználói azonosítóval. Ezáltal meg tudjuk állapítani, ki mivel foglalkozik éppen a rendszerben, és meg tudjuk akadályozni, hogy a felhasználók elérjék egymás beállításait, olvassák egymás leveleit és így tovább.

Minden felhasználó alakítani tudja a saját környezetét, és ezzel mintegy berendezkedik a rendszerünkben, különféle parancsértelmezők, szövegszerkesztők, billentyű-hozzárendelések és nyelvek használatával.

13.6. A hozzáférések módosítása

Egy UNIX-os környezetben több különböző parancs közül választhatunk a felhasználói hozzáférések módosításakor. A legáltalánosabb parancsokat az alábbiakban foglaljuk össze, amit ezután a használatukat részletesebben bemutató példák követnek.

Parancs	Leírás
<code>adduser(8)</code>	az új felhasználók felvételére ajánlott parancssoros alkalmazás
<code>rmuser(8)</code>	a felhasználók eltávolítására ajánlott parancssoros alkalmazás
<code>chpass(1)</code>	rugalmas eszköz a felhasználói adatbázis információinak megváltoztatására
<code>passwd(1)</code>	egy egyszerű parancssoros segédprogram a felhasználói jelszavak megváltoztatásához
<code>pw(8)</code>	egy erőteljes és rugalmas segédeszköz a felhasználói hozzáférések teljeskörű módosításához

13.6.1. `adduser`

Az `adduser(8)` a felhasználók hozzáadására használható egyszerű program. Bejegyzéseket hoz létre a rendszer `passwd` és `group` állományában. Ezen kívül még létrehozza az új felhasználó könyvtárát is, odamásolja az alapértelmezett konfigurációs állományokat a `/usr/share/skel` könyvtárból (ezek a felhasználóknál ponttal kezdődően jelennek meg, de az említett könyvtárban „dot” előtaggal szerepelnek), és opcionálisan küld egy üdvözlőlevelet az újdonsült felhasználónak.

Példa 13-1. Felhasználó hozzáadása a FreeBSD-ben

```
# adduser
Username: jantyyik
Full name: Jantyyik Zsolt
Uid (Leave empty for default):
Login group [jantyyik]:
Login group is jantyyik. Invite jantyyik into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh zsh nologin) [sh]: zsh
Home directory [/home/jantyyik]:
Use password-based authentication? [yes]:
Use an empty password? (yes/no) [no]:
Use a random password? (yes/no) [no]:
Enter password:
```

```

Enter password again:
Lock out the account after creation? [no]:
Username      : jantyk
Password      : ****
Full Name     : Jantyk Zsolt
Uid           : 1001
Class         :
Groups        : jantyk wheel
Home          : /home/jantyk
Shell         : /usr/local/bin/zsh
Locked        : no
OK? (yes/no): yes
adduser: INFO: Successfully added (jantyk) to the user database.
Add another user? (yes/no): no
Goodbye!
#

```

Megjegyzés: A jelszó a beírás során egyáltalán nem jelenik meg, még csillagokat sem láthatunk a karakterek helyén. Ezért vigyázzunk, nehogy elgépeljük véletlenül a jelszót!

13.6.2. rmuser

Az `rmuser(8)` használható a felhasználók teljes eltávolítására a rendszerből. Az `rmuser(8)` az alábbi lépéseket hajtja végre:

1. Eltávolítja a felhasználó `crontab(1)` bejegyzéseit (amennyiben léteznek).
2. Eltávolítja az `at(1)` felhasználóhoz tartozó munkáit.
3. Leállítja a felhasználó által birtokolt összes futó programot.
4. Eltávolítja a felhasználót a rendszer helyi jelszó állományából.
5. Eltávolítja a felhasználó könyvtárát (amennyiben az a felhasználó birtokában van).
6. Eltávolítja a felhasználóhoz tartozó beérkező leveleket tartalmazó állományt a `/var/mail` könyvtárból.
7. Eltávolítja a felhasználó tulajdonában levő összes állományt az olyan ideiglenes tárhelyekről, mint például a `/tmp` könyvtár.
8. Végezetül eltávolítja a felhasználó nevét az összes olyan csoportból, amelyhez az `/etc/group` szerint tartozik.

Megjegyzés: Ha menet közben egy csoport üressé válik, és a csoport neve megegyezik a felhasználó nevével, a csoport is eltávolításra kerül. Ez kiegészíti az `adduser(8)` eszközzel létrehozott felhasználónkénti egyedi csoportokat.

A `rmuser(8)` nem használható adminisztrátori hozzáférések törlésére, mivel az szinte majdnem mindig a teljes összeomlást vonja maga után.

Alapértelmezés szerint interaktív módban működik, melynek során megpróbál megbizonyosodni róla, hogy tényleg a megfelelő dolgot cselekedjük.

Példa 13-2. `rmuser` Hozzáférések interaktív eltávolítása

```
# rmuser jantyk
Matching password entry:
jantyk:*:1001:1001::0:0:Jantyk Zsolt:/home/jantyk:/usr/local/bin/zsh
Is this the entry you wish to remove? y
Remove user's home directory (/home/jantyk)? y
Updating password file, updating databases, done.
Updating group file: trusted (removing group jantyk -- personal group is empty) done.
Removing user's incoming mail file /var/mail/jantyk: done.
Removing files belonging to jantyk from /tmp: done.
Removing files belonging to jantyk from /var/tmp: done.
Removing files belonging to jantyk from /var/tmp/vi.recover: done.
#
```

13.6.3. `chpass`

A `chpass(1)` segítségével meg tudjuk változtatni a felhasználói adatbázisban található információkat, mint például a jelszavakat, parancsértelmezőket és a személyes adatokat.

Csak a rendszeradminisztrátoroknak, mint például magának az adminisztrátornak, szabad megváltoztatnia a felhasználók adatait a `chpass(1)` programmal.

Amikor az opcionálisan megadható felhasználói névtől eltekintve nem adunk át neki paramétereket, a `chpass(1)` egy szövegszerkesztőben megnyitja az érintett felhasználó adatait. Miután kiléptünk belőle, a felhasználói adatbázist a megváltoztatott adatoknak megfelelően frissíti.

Megjegyzés: Ha nem adminisztrátorként hívjuk meg, akkor a rendszer kérni fogja a jelszavunkat, miután kiléptünk a szövegszerkesztőből.

Példa 13-3. A `chpass` interaktív használata adminisztrátorként

```
# A jantyk nevű felhasználó adatainak módosítása.
Login: jantyk
Password: *
Uid [#]: 1001
Gid [# or name]: 1001
Change [month day year]:
Expire [month day year]:
Class:
Home directory: /home/jantyk
Shell: /usr/local/bin/zsh
Full Name: Jantyk Zsolt
Office Location:
Office Phone:
Home Phone:
```

Other information:

Egy átlagos felhasználó a bemutatott adatoknak csak igen kis részét képes módosítani, és azokat is csak saját maga számára.

Példa 13-4. A `chpass` interaktív használata normál felhasználóként

```
# A jantyk nevű felhasználó adatainak megváltoztatása.
Shell: /usr/local/bin/zsh
Full Name: Jantyk Zsolt
Office Location:
Office Phone:
Home Phone:
Other information:
```

Megjegyzés: A `chfn(1)` és `chsh(1)` parancsok csupán linkek a `chpass(1)` parancsra, akárcsak a `ypchpass(1)`, `ypchfn(1)` és az `ypchsh(1)`. A NIS támogatása teljesen magától működik, ezért az `yp` előtag használata nem kötelező. Ha ez nem érthető, nem kell megijedni, a NIS-t majd a 29 fejezetben bemutatjuk.

13.6.4. `passwd`

Felhasználóként a saját jelszavunkat, adminisztrátorként pedig bármelyik felhasználó jelszavát a `passwd(1)` segítségével változtathatjuk meg a megszokott módon.

Megjegyzés: A véletlen balesetek és az illetéktelen változtatások ellen védelmet nyújt, hogy az eredeti jelszót is meg kell adnunk az új jelszó beállításához.

Példa 13-5. A jelszavunk megváltoztatása

```
% passwd
Changing local password for jantyk.
Old password:
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

Példa 13-6. Egy másik felhasználó jelszavának megváltoztatása adminisztrátorként

```
# passwd jantyk
Changing local password for jantyk.
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

Megjegyzés: Ahogy a `chpass(1)`, az `yppasswd(1)` is csak egy link a `passwd(1)` parancsra, így a NIS mind a két megadási módban működik.

13.6.5. pw

A `pw(8)` egy olyan parancssori segédprogram, amellyel felhasználókat és csoportokat tudunk létrehozni, törölni, módosítani és megjeleníteni. Ez tulajdonképpen a rendszer felhasználókat és csoportokat tároló állományainak egyfajta kezelőfelülete. A `pw(8)` kiválóan paraméterezhető, aminek köszönhetően remekül kiaknázható tudása a különféle parancsértelmezők szkriptjeiben, habár a kezdő felhasználók nehezkesebbnek érezhetik a kezelését a korábban mutatott parancsokhoz képest.

13.7. A felhasználók korlátozása

Ha már vannak felhasználóink, gyakran szóba kerülhet esetükben a rendszer használatának korlátozása. A FreeBSD rengeteg módon engedi korlátozni a rendszergazdának az egyénenként használható erőforrások mennyiségét a rendszerben. Ezek a korlátok két részre oszthatóak: a lemezkvótákra és egyéb erőforráskorlátokra.

A lemezkvóták a felhasználók lemezhasználatát korlátozzák, és lehetővé teszik, hogy állandó újraszámolás nélkül, gyorsan ellenőrizni tudjuk ennek mértékét. A kvótákat a 18.15 Szakaszban részletezzük.

A többi erőforrás korlátozása magában foglalja a processzoridő, memória és minden olyan erőforrás behatárolását, amihez a felhasználó csak hozzá tud férni. Ezeket bejelentkezési osztályokon keresztül határozzuk meg, ezekről esik itt most szó.

A bejelentkezési osztályokat az `/etc/login.conf` állományban adhatjuk meg. Ennek pontos ismertetése nem tárgya ennek a szakasznak, de ezt megtalálhatjuk a `login.conf(5)` man oldalon. Elegendő csak annyit mondanunk, hogy minden felhasználóhoz tartozik egy bejelentkezési osztály (alapértelmezés szerint a `default` nevű), és minden egyes bejelentkezési osztályhoz tulajdonságok egy halmaza társul. Ezek a bejelentkezési tulajdonságok *név=érték* párosokból állnak, ahol *név* egy jól ismert azonosító, illetve az *érték* egy tetszőleges sztring, melyet a nevtől függő módon dolgozunk fel. A bejelentkezési osztályok és tulajdonságok beállítása eléggé magától értetődő, és a `login.conf(5)` man oldal is jól leírja.

Megjegyzés: A rendszer általában nem magát az `/etc/login.conf` állományban található beállításokat olvassa be, hanem az `/etc/login.conf.db` állományt, amiben gyorsabban lehet keresni. Az `/etc/login.conf` állományból az `/etc/login.conf.db` állományt az alábbi paranccsal tudjuk legyártani:

```
# cap_mkdb /etc/login.conf
```

Az erőforrások korlátozása két irányban is eltér a sima hétköznapi bejelentkezési tulajdonságoktól. Először is minden korláthoz létezik egy gyenge (aktuális) és egy erős korlát. A gyenge korlátok a felhasználók vagy az alkalmazások részéről még finomíthatóak, de az erős korláton túl már nem. Ez utóbbit mindig tudja csökkenteni a felhasználó, de sose tudja növelni. Másodsorban a legtöbb erőforráskorlát az adott felhasználó által futtatott programokra egyenként vonatkozik, nem pedig az összesre együttesen. Megjegyezzük azonban, hogy ezeket az eltéréseket a korlátok különleges kezelése indokolja, nem pedig a bejelentkezési tulajdonságok rendszerének megvalósítása (tehát a korlátok *valójában nem* ezen tulajdonságok speciális esetei.)

Így aztán, minden további magyarázkodás nélkül, felsoroljunk alant a leggyakrabban alkalmazott erőforráskorlátokat (a többi, más egyéb bejelentkezési tulajdonságokkal együtt, megtalálható a `login.conf(5)` man oldalon).

`coredumpsize`

A program által létrehozott memóriakivonat maximális méretét határolja be ez a korlát, értelemszerűen a többi lemezterületre vonatkozó korlátnak (például a `filesize` vagy a lemezkvóták) alárendelt módon. Mindazonáltal ezt gyakran használjuk egyfajta enyhébb lemezfoglalási korlátként. Mivel nem maguk a felhasználók hozzák létre ezeket az állományokat és sokszor nem is törlik le ezeket, ez a beállítás azonban megmentheti ezeket a nagyobb programok (mint például az **emacs**) összeomlása során keletkező memóriakivonatok felesleges helyfoglalásától.

`cputime`

Az a maximális processzoridő, amit a felhasználó által futtatott programok egyenként fogyaszthatnak. A vétkező programok futását a rendszermag leállítja.

Megjegyzés: Ez a korlát a processzoridőre vonatkozik, nem pedig a processzor kihasználtságának százalékára, ahogy a `top(1)` és a `ps(1)` szokta megjeleníteni. Ez utóbbi alapján korlátozni ugyanis, még ezen leírás készítésének pillanataiban nem lehetséges, és meglehetősen hasztalan is lenne: egy fordítóprogram — ami minden bizonnyal egy szabályosan futó program — könnyen fel tudja emészteni majdnem az egész processzort egy időre.

`filesize`

A felhasználó által birtokolható állományok maximális mérete. Eltérően a lemezkvótáktól, ez a korlát az egyes állományokra vonatkozik, nem pedig a felhasználó összes állományára együttesen.

`maxproc`

A felhasználó által egyidőben, az előtérben és a háttérben futtatható programok maximális száma. Érthető okokból ez az érték nem lehet nagyobb, mint a rendszerben a `sysctl(8)` által definiált `kern.maxproc` (a rendszermag által maximálisan futtatható programok számának) értéke. Érdemes még továbbá megjegyezni, hogy ez a beállítás gátolhatja a felhasználó munkáját: gyakran hasznos lehet egyszerre több példányban is bejelentkezni a rendszerbe vagy csövekkel összekapcsolt programokat futtatni. Bizonyos feladatok, mint például egy nagyobb program lefordítása, több program futására is szétterjedhetnek (például a `make(1)`, `cc(1)` és egyéb köztes feldolgozókra).

`memorylocked`

Ezzel korlátozhatjuk az egyes futó programok által zárolható memóriaterület méretét a központi memóriában (lásd `mlock(2)`). Egyes rendszerkritikus programok, mint például az `amd(8)`, zárolják magukat a központi

memóriában, és ezért soha nem lapozódnak ki onnan. Ennek köszönhetően nem érinti ezeket a rendszer lapozásból eredő esetleges lelassulása.

memoryuse

Ez az a maximális memóriamennyiség, amelyet egy futó program egyszerre használhat. Ebbe együttesen beleértendő a központi memóriában és a lapozóállományban elfoglalt hely. Ez ugyan nem minden szempontból korlátozza egy program memóriahasználatát, de indulásnak megfelelő.

openfiles

A felhasználó egyes futtatott programjai által egy időben megnyitható állományok maximális száma. FreeBSD-ben az állományok közé a foglalatok és az IPC-csatornák is beszámítanak. Ezért vigyázzunk, nehogy véletlenül túlságosan alacsonyra állítsuk ezt az értéket. Ezt rendszerszinten a `kern.maxfiles` `sysctl(8)` érték határozza meg.

sbsize

A korlátozás a felhasználó által egyszerre maximálisan elérhető hálózati memória és így a rendszermag puffereire vonatkozik. Eredetileg a régebbi, sok csatlakozást felemésztő DoS (Denial of Service) támadások ellen nyújtana védelmet, de általánosságban alkalmazható a hálózati kommunikáció korlátozására is.

stacksize

Ez a felhasználó által működtetett egyes programok vermeinek maximális mérete. Önmagában nem elegendő a programok által használt memóriamennyiség korlátozására, így emiatt inkább a többi korláttal együtt érdemes alkalmazni.

Van néhány tényező, amelyekre érdemes odafigyelni az erőforrások korlátainak beállítása során. Most következik pár tipp, javaslat és egyéb megjegyzés a témához.

- A rendszerindítás során az `/etc/rc` által indított programok a `daemon` bejelentkezési osztályba tartoznak.
- Habár a rendszerrel érkező `/etc/login.conf` állományban remekül be van állítva a legtöbb korlát, de nekünk, mint rendszergazdáknak, kell ismernünk a saját rendszerünk korlátait. Ezen korlátok túlzott tágításával a rendszerünk könnyen leterhelhetővé válik, míg a túlzott szűkítésével akadályozhatjuk a hatékony használatát.
- Az X Window System (X11) felhasználóinak a többi felhasználónál valószínűleg jóval több erőforráshoz kell tudniuk hozzáférni. Az X11 már önmagában sok erőforrást eszik, de egyben bátorítja is a felhasználókat több program párhuzamos futtatására.
- Ne felejtjük el, hogy sok korlát az egyes különállóan futó programokra vonatkozik, nem pedig a felhasználó összes futtatott programjára. Például ha beállítjuk 50-re az `openfiles` értékét, a felhasználó által elindított programok mindegyike legfeljebb 50 állományt tud majd megnyitni. Emiatt a felhasználó által egyszerre ténylegesen megnyitható állományok száma az `openfiles` és a `maxproc` aktuális értékeinek szorzatából adódik. Ugyanez igaz a memóriahasználatra is.

Az erőforrások korlátozásáról, a bejelentkezési osztályokról és tulajdonságaikról a hozzájuk tartozó man oldalakon olvashatunk: `cap_mkdb(1)`, `getrlimit(2)` és `login.conf(5)`.

13.8. Csoportok

Egy csoport nem több felhasználók összességénél. A csoportokat a nevük és az azonosítójuk (Group ID, GID) azonosítja be. A FreeBSD-ben (és a legtöbb UNIX-szerű rendszerben) a rendszermag két tényező alapján dönt arról, mit szabad tennie egy futó programnak: ezek közül az egyik a tulajdonosának azonosítója (UID), a másik azon csoportok listája, melyeknek tagja a tulajdonos. Eltérően a UID-től, egy futó programhoz csoportok listája tartozik. Amikor egy felhasználó vagy egy futó program "csoportazonosítójára" hivatkoznak, általában csak a lista első elemére gondolnak.

A csoportok nevei és azonosítói közti megfeleltetéseket az `/etc/group` állományban találjuk. Ez lényegében egy szimpla szöveges állomány, négy kettősponttal elválasztott mezőt tartalmaz. Ezek közül az első a csoport neve, a második a titkosított jelszó, a harmadik a csoport azonosítója, a negyedik pedig a tagok vesszővel tagolt felsorolása. Akár kézzel is nyugodtan szerkeszthető (feltételezve persze, hogy nem vétünk benne szintaktikai hibát!). A szintaxis teljes leírását a `group(5)` man oldalon találhatjuk meg.

Ha nem akarjuk magunk szerkeszteni az `/etc/group` állományt, használhatjuk a `pw(8)` parancsot is csoportok létrehozására és törlésére. Például hozzuk létre a `pg_csoport` nevű csoportot és vizsgáljuk meg, valóban létrejött-e:

Példa 13-7. A csoportok tagjainak beállítása a `pw(8)` használatával

```
# pw groupadd pg_csoport
# pw groupshow pg_csoport
pg_csoport:*:1100:
```

A fent szereplő 1100-as érték a `pg_csoport` csoportazonosítója. Ebben a pillanatban a `pg_csoport`-nak még egyetlen tagja sincs, ami miatt lényegében haszontalan. Így hát hívjuk meg a `pg_csoport`-ba a korábban létrehozott `jantyyik` nevű felhasználót.

Példa 13-8. A csoport tagjainak beállítása a `pw(8)` használatával

```
# pw groupmod pg_csoport -M jantyyik
# pw groupshow pg_csoport
pg_csoport:*:1100:jantyyik
```

Az `-M` kapcsoló paramétere a csoportba sorolandó felhasználók neveinek vesszőkkel tagolt listája. A korábbi szakaszok alapján már tudjuk, hogy a jelszavakat tároló állomány egyben azokat a csoportokat is tartalmazza, ahova az egyes felhasználók tartoznak. Az utóbbiakat (a felhasználókat) automatikusan beleteszi a rendszer a csoportlistába, de az érintett felhasználó nem fog megjelenni tagként a `pw(8)` parancs `groupshow` utasításával, azonban az `id(1)` és a hozzá hasonló eszközökkel már látható lesz. Más szavakkal élve, a `pw(8)` csak az `/etc/group` állományt módosítja, és soha nem próbál meg további adatokat kiolvasni az `/etc/passwd` állományból.

Példa 13-9. Egy új tag felvétele a csoportba a `pw(8)` használatával

```
# pw groupmod pg_csoport -m kisati
# pw groupshow pg_csoport
pg_csoport:*:1100:jantyyik,kisati
```

Az `-m` kapcsoló paramétere azon felhasználók vesszővel tagolt listája, akiket fel akarunk venni a csoportba. Tehát eltérően az előző példától, ezeket a felhasználókat felvesszük a csoportba, nem pedig átírjuk velük a csoport jelenlegi tagjainak listáját.

Példa 13-10. Az id(1) használata a csoporttagság megállapítására

```
% id jantyk
uid=1001(jantyk) gid=1001(jantyk) groups=1001(jantyk), 1100(pg_csoport)
```

Ahogy láthatjuk is, a jantyk nevű felhasználó tagja a jantyk nevű csoportnak és a pg_csoportnak is.

A pw(8) működéséről a saját man oldalán, az `/etc/group` formátumáról pedig a group(5) man oldalon találhatunk több információt.

Megjegyzések

1. Hacsak nem kapcsolunk hozzá több terminált. De ennek a leírását a 26 fejezetre tartogatjuk.
2. Lehetséges akár 4294967295-ig is számozni az UID/GID értékét, de az ekkora nagyságú azonosítók komoly gondokat okozhatnak az olyan szoftvereknek, melyek bizonyos feltételezésekkel élnek az értékeikkel kapcsolatban.

Fejezet 14. Biztonság

A fejezet legnagyobb részét a security(7) man oldal alapján írta: Matthew Dillon.

14.1. Áttekintés

Ez a fejezet egy alapvető bevezetés a rendszerek biztonsági fogalmaiba, ad néhány általános jótanácsot és a FreeBSD-vel kapcsolatban feldolgoz néhány komolyabb témát. Az itt megfogalmazott témák nagy része egyaránt ráhúzható rendszerünk és általánosságban véve az internet biztonságára is. A internet már nem az “békés” hely, ahol mindenki a kedves szomszéd szerepét játssza. A rendszerünk bebiztosítása elkerülhetetlen az adataink, szellemi tulajdonunk, időnk és még sok minden más megvédésére az internetes banditák és hasonlók ellen.

A FreeBSD segédprogramok és mechanizmusok sorát kínálja fel a rendszerünk és hálózatunk sértetlenségének és biztonságának fenntartására.

A fejezet elolvasása során megismerjük:

- az alapvető rendszerbiztonsági fogalmakat, különös tekintettel a FreeBSD-re;
- milyen olyan különböző titkosítási mechanizmusok érthetőek el a FreeBSD-ben, mint például a DES és az MD5;
- hogyan állítsunk be egyszeri jelszavas azonosítást;
- hogyan burkoljunk az **inetd** segítségével TCP kapcsolatokat;
- hogyan állítsuk be a **KerberosIV**-t a FreeBSD 5.0-nál korábbi változatain;
- hogyan állítsuk be a **Kerberos5**-t a FreeBSD-n;
- hogyan állítsuk be az IPsec-et és hozzunk létre VPN-t FreeBSD/Windows gépek között;
- hogyan állítsuk be és használjuk az **OpenSSH**-t, a FreeBSD SSH implementációját;
- mik azok az ACL-ek az állományrendszerben és miként kell ezeket használni;
- hogyan kell használni a **Portaudit** segédprogramot a Portgyűjteményből telepített külső szoftvercsomagok biztonságosságának ellenőrzésére;
- hogyan hasznosítsuk a FreeBSD biztonsági tanácsait tartalmazó leírásokat
- mit jelent a futó programok nyilvántartása és hogyan engedélyezzük azt FreeBSD-n.

A fejezet elolvasásához ajánlott:

- az alapvető FreeBSD és internetes fogalmak ismerete.

A könyvben további biztonsági témákról is szó esik, például a 16 fejezetben a Kötelező hozzáférés-vezérlésről (MAC) és a 30 fejezetben pedig az internetes tűzfalakról.

14.2. Bevezetés

A biztonság egy olyan funkció, ami a rendszergazdától indul és nála is végződik. Míg az összes többfelhasználós BSD UNIX rendszer önmagában is valamennyire biztonságos, a felhasználók “fegyelmezéséhez” szükség további biztonsági mechanizmusok kiépítésére és karbantartására, ami minden bizonnyal egy rendszergazda egyik

legfontosabb kötelessége. A számítógépek csak annyira biztonságosak, mint amennyire beállítjuk, és a biztonsági megfontolások állandó versenyben vannak az emberi kényelemmel. A UNIX rendszerek általánosságban véve órási mennyiségű program párhuzamos futtatására képesek, melyek többsége kiszolgálóként fut — ez azt jelenti, hogy hozzájuk kívülről érkező egyedek csatlakozhatnak és társaloghatnak velük. Ahogy a tegnap kicsi és nagy számítógépei napjaink asztali gépeivé váltak és ahogy a számítógépek egyre többen csatlakoznak hálózatra és az internetre, a biztonság fontossága is egyre jobban növekszik.

A rendszerek biztonsága a támadások különböző formáival is foglalkozik, többek közt olyan támadásokkal, amelyek a rendszer összeomlását vagy használhatatlanságát célozzák meg, de nem próbálják meg veszélybe sodorni a `root` felhasználó hozzáférését (“feltörni a gépet”). A biztonsággal kapcsolatos problémák több kategóriára oszthatóak:

1. A szolgáltatások működésképtelenné tételére irányuló (DoS, Denial of Service) támadások.
2. A felhasználói fiókok veszélyeztetése.
3. Rendszergazdai jogok megszerzése a közeli szervereken keresztül.
4. Rendszergazdai jogok megszerzése a felhasználói fiókokon keresztül.
5. Kiskapuk létrehozása a rendszerben.

A szolgáltatások működésképtelenné tételére irányuló támadások olyan tevékenységre utalnak, amelyek képesek megfosztani egy számítógépet az erőforrásaitól. A DoS támadások többnyire nyers erővel kivitelezett technikák, melyek vagy a rendszer összeomlasztását vagy pedig a használhatatlanná tételét veszik célba úgy, hogy túlterhelik az általa felkínált szolgáltatásokat vagy a hálózati alrendszert. Egyes DoS támadások a hálózati alrendszerben rejtőző hibákat igyekeznek kihasználni, amivel akár egyetlen csomaggal is képesek romba dönteni egy számítógépet. Ez utóbbit csak úgy lehet orvosolni, ha a hibát kijavítjuk a rendszermagban. A szerverekre mért csapásokat gyakran ki lehet védeni a paramétereik ügyes beállításával, melyek segítségével korlátozni tudjuk az ezeket ért terhelést egy kellemetlenebb helyzetben. A nyers erőt alkalmazó hálózati támadásokkal a legnehezebb szembenézni. Például az álcázott támadások, melyeket szinte lehetetlen megállítani, remek eszközök arra, hogy elvágják gépünket az internettől. Ezzel viszont nem csak azt iktatják ki, hanem az internet-csatlakozásunkat is eldugítják.

A DoS támadásoknál még gyakrabban előfordul, hogy feltörik a felhasználók fiókjait. A rendszergazdák többsége még mindig futtat **telnetd**, **rlogin**, **rshd** és **ftpd** szervereket a gépen. Ezek a szerverek alapértelmezés szerint nem titkosított kapcsolaton keresztül működnek. Ebből következik, hogy ha nincs annyira sok felhasználónk és közülük néhányan távoli helyekről jelentkeznek be (ami az egyik leggyakoribb és legkényelmesebb módja ennek), akkor előfordulhat, hogy valami megneszeli a jelszavaikat. A körületekintő rendszergazdák mindig ellenőrzik a bejelentkezéseket tartalmazó naplókat és igyekeznek kiszűrni a gyanús címeket még abban az esetben is, amikor a bejelentkezés sikeres volt.

Mindig arra kell gondolni, hogy ha a támadónak sikerült megszerezni az egyik felhasználó hozzáférését, akkor akár képes lehet a `root` felhasználó fiókjának feltörésére is. Azonban a valóságban egy jól őrzött és karbantartott rendszer esetén a felhasználói hozzáférések megszerzése nem feltétlenül adja a támadó kezére a `root` hozzáférést. Ebben fontos különbséget tenni, hiszen a `root` felhasználó jogai nélkül a támadó nem képes elrejteni a nyomait és legjobb esetben sem tud többet tenni, mint tönkretenni az adott felhasználó állományait vagy összeomlasztani a rendszert. A felhasználói fiókok feltörése nagyon gyakran megtörténik, mivel a felhasználók messze nem annyira elővigyázatosak, mint egy rendszergazda.

A rendszergazdáknak mindig észben kell tartani, hogy egy számítógépen több módon is meg lehet szerezni a `root` felhasználó hozzáférését. A támadó megtudhatja a `root` jelszavát, hibát fedezhet fel az egyik rendszergazdai jogosultsággal futó szerverben és képes feltörni a `root` hozzáférést egy hálózati kapcsolaton keresztül, vagy a támadó olyan programban talál hibát, aminek segítségével el tudja érni a `root` fiókját egy felhasználói hozzáféréseken keresztül. Miután a támadó megtalálta a rendszergazdai jogok megszerzésének módját, nem feltétlenül kell

kiskapukat elhelyeznie a rendszerben. Az eddig talált és javított, rendszergazdai jogok megszerzését lehetővé tevő biztonsági rések egy része esetében viszont a támadónak akkora mennyiségű munkát jelentene eltüntetni maga után a nyomokat, hogy megéri neki egy kiskaput telepíteni. Ennek segítségével a támadó ismét könnyedén hozzájuthat a `root` felhasználó hozzáférésehez a rendszerben, de ezen keresztül egy okos rendszergazda képes is a behatolót leleplezni. A kiskapuk lerakásának megakadályozása valójában káros a biztonság szempontjából nézve, mert ezzel nem szüntetjük meg azokat a lyukakat, amin keresztül a támadó először bejutott.

A támadások elleni védelmet mindig több vonalban kell megvalósítani, melyeket így oszthatunk fel:

1. A rendszergazda és a személyzet hozzáféréseinek védelme.
2. A rendszergazdai jogokkal futó szerverek és a `suid/sgid` engedélyekkel rendelkező programok védelme.
3. A felhasználói hozzáférések védelme.
4. A jelszavakat tároló állomány védelme.
5. A rendszermag belsejének, a nyers eszközök és az állományrendszerek védelme.
6. A rendszert ért szabálytalan módosítások gyors észlelése.
7. Állandó paranoia.

A fejezet most következő szakaszában az imént felsorolt elemeket fejtjük ki részletesebben.

14.3. A FreeBSD védelme

Parancs kontra protokoll: A dokumentumban a **félkövéren** fogjuk szedni az alkalmazásokat, és egyenszerűségű betűkkel pedig az adott parancsokra hivatkozunk. A protokollokat nem különböztetjük meg. Ez a tipográfiai elkülönítés hasznos például az `ssh` egyes vonatkozásainak esetén, mivel ez egyben egy protokoll és egy parancs is.

A most következő szakaszok a FreeBSD védelmének azon módszereit ismertetik, amelyekről a fejezet előző szakaszában már írtunk.

14.3.1. A rendszergazda és a személyzet hozzáféréseinek védelme

Először is: ne törjük magunkat a személyzeti fiókok biztonságossá tételével, ha még a rendszergazda hozzáférést sem tettük eléggé biztonságossá. A legtöbb rendszerben a `root` hozzáféréshez tartozik egy jelszó. Elsőként fel kell tennünk, hogy ez a jelszó *mindig* megszerezhető. Ez természetesen nem arra utal, hogy el kellene távolítanunk. A jelszó szinte mindig szükséges a számítógép konzolon keresztüli eléréséhez. Valójában arra szeretnénk rávilágítani, hogy a konzolon kívül sehol máshol ne lehessen használni ezt a jelszót, még a `su(1)` paranccsal sem. Például gondoskodjunk róla, hogy az `/etc/ttys` állományban megadott pszeudó terminálokat "insecure" (nem biztonságos) típusúnak állítottuk be, és így a `telnet` vagy az `rlogin` parancsokon keresztül nem lehet rendszergazdaként bejelentkezni. Ha más szolgáltatáson keresztül jelentkeznünk be, például az **ssh**d segítségével, akkor ebben az esetben is gondoskodjunk róla, hogy letiltottuk a közvetlen rendszergazdai bejelentkezés lehetőségét. Ezt úgy tudjuk megtenni, ha megnyitjuk az `/etc/ssh/ssh_config` állományt és a `PermitRootLogin` paramétert átállítjuk a `no` értékre. Vegyünk számba minden lehetséges hozzáférési módot — az FTP és a hozzá hasonló módok gyakran átszivárognak a repedéseken. A rendszergazdának csak a rendszerkonzolon keresztül szabad tudnia bejelentkeznie.

Természetesen egy rendszergazdának valahogy el kell érnie a `root` hozzáférést, ezért ezzel felnyitunk néhány biztonsági rést. De gondoskodjunk róla, hogy ezek a rések további jelszavakat igényelnek a működésükhöz. A `root` hozzáférés eléréséhez érdemes felvenni tetszőleges személyzeti (staff) hozzáféréseket a `wheel` csoportba (az `/etc/group` állományban). Ha a személyzet tagjait a `wheel` csoportba rakjuk, akkor innen a `su` paranccsal fel tudjuk venni a `root` felhasználó jogait. A személyzet tagjait létrehozásukkor közvetlenül sose vegyük fel a `wheel` csoportba! A személyzet tagjai először kerüljenek egy `staff` csoportba, és majd csak ezután az `/etc/group` állományon keresztül a `wheel` csoportba. A személyzetnek csak azon tagjait tegyük ténylegesen a `wheel` csoportba, akiknek valóban szükségük van a `root` felhasználó hozzáférésére. Ha például a Kerberost használjuk hitelesítésre, akkor megcsinálhatjuk azt is, hogy a Kerberos `.k5login` állományában engedélyezzük a `ksu(1)` parancson keresztül a `root` hozzáférés elérését a `wheel` csoport alkalmazása nélkül. Ez a megoldás talán még jobb is, mivel a `wheel` használata esetén a behatolónak még mindig lehetősége van hozzájutni a `root` hozzáféréséhez olyankor, amikor a kezében van a jelszavakat tároló állomány és meg tudja szerezni a személyzet valamelyik tagjának hozzáférését. A `wheel` csoport által felkínált megoldás ugyan jobb, mint a semmi, de kétségtelenül nem a legbiztonságosabb.

A hozzáférések teljes körű letiltásához a `pw(8)` parancsot érdemes használni:

```
# pw lock személyzet
```

Ezzel meg tudjuk akadályozni, hogy a felhasználó akármilyen módon, beleértve az `ssh(1)` használatát is, hozzá tudjon férni a rendszerünkhöz.

A hozzáférések blokkolásának másik ilyen módszere a titkosított jelszó átírása egyetlen “*” karakterre. Mivel ez a karakter egyetlen titkosított jelszóra sem illeszkedik, ezért a felhasználó nem lesz képes bejelentkezni. Ahogy például a személyzet alábbi tagja sem:

```
izemize:R9DT/Fa1/LV9U:1000:1000::0:0:Ize-Mize:/home/izemize:/usr/local/bin/tcsh
```

Erre cseréljük ki:

```
izemize::1000:1000::0:0:Ize-Mize:/home/izemize:/usr/local/bin/tcsh
```

Ezzel megakadályozzuk, hogy az `izemize` nevű felhasználó a hagyományos módszerekkel be tudjon jelentkezni. Ez a megoldás azonban a **Kerberost** alkalmazó rendszerek esetén nem működik, illetve olyan helyzetekben sem, amikor a felhasználó az `ssh(1)` paranccsal már létrehozott magának kulcsokat.

Az ilyen védelmi mechanizmusok esetében mindig egy szigorúbb biztonsági szintű gépről jelentkezünk be egy kevésbé biztonságosabb gépre. Például, ha a szerverünk mindenféle szolgáltatásokat futtat, akkor a munkaállomásunknak egyetlen egyet sem lenne szabad. A munkaállomásunk biztonságossá tételéhez a lehető legkevesebb szolgáltatást szabad csak futtatnunk, de ha lehet, egyet sem, és mindig jelszóval védett képernyővédőt használjuk. Természetesen ha a támadó képes fizikailag hozzáférni a munkaállomásunkhoz, akkor szinte bármilyen mélységű védelmet képes áttörni. Ezt mindenképpen számításba kell vennünk, azonban ne felejtjük el, hogy a legtöbb betörési kísérlet távolról, hálózaton keresztül érkezik olyan emberektől, akik fizikailag nem férnek hozzá a munkaállomásunkhoz vagy a szervereinkhez.

A Kerberos és a hozzá hasonló rendszerek használatával egyszerre tudjuk a személyzet tagjainak jelszavát letiltani vagy megváltoztatni, ami egyből érvényessé válik minden olyan gépen, ahová az adott felhasználónak bármilyen hozzáférése is volt. Nem szabad lebecsülnünk ezt a gyors jelszóváltási lehetőséget abban az esetben, ha a személyzet valamelyik tagjának hozzáférését megszerezték. Hagyományos jelszavak használatával a jelszavak megváltoztatása N gépen igazi káosz. A Kerberosban jelszóváltási megszorításokat is felállíthatunk: nem csak a Kerberos által adott jegyek járnak le idővel, hanem a Kerberos rendszer meg is követelheti a felhasználóktól, hogy egy adott idő (például egy hónap) után változtasson jelszót.

14.3.2. A rendszergazdai jogokkal futó szerverek és SUID/SGID engedélyekkel rendelkező programok védelme

A bölcs rendszergazda mindig csak akkor futtat szervereket, amikor szüksége van rá, se többet, se kevesebbet. Az egyéb fejlesztőktől származó szerverekkel bányunk különösen óvatosan, mivel gyakran hajlamosak hibákat tartalmazni. Például az **imapd** vagy a **popper** használata olyan, mintha az egész világnak ingyenjegyet osztogatnánk a rendszerünk `root` hozzáférésehez. Soha ne futtassunk olyan szervert, amelyet nem vizsgáltunk át kellő alapossággal. Sok szervert nem is feltétlenül kell `root` felhasználóként futtatni. Például az **ntalk**, **comsat** és **finger** démonok egy speciális *járókában* (sandbox) futnak. Ezek a járókák sem teljesen tökéletesek, hacsak erre külön figyelmet nem fordítunk. Ilyenkor a többvonalas védelem eszménye még mindig él: ha valakinek sikerült betörnie a járókába, akkor onnan ki is tud törni. Minél több védelmi vonalat húzunk a támadó elé, annál jobban csökken a sikerének valószínűsége. A történelem során lényegében minden `root` jogokkal futó szerverben, beleértve az alapvető rendszerszintű szervereket is, találtak már biztonsági jellegű hibát. Ha a gépünkre csak az **sshd** szolgáltatáson keresztül tudnak belépni, és soha nem használja senki a **telnetd**, **rshd** vagy **rlogind** szolgáltatásokat, akkor kapcsoljuk is ki ezeket!

A FreeBSD most már alapértelmezés szerint járókában futtatja az **ntalkd**, **comsat** és **finger** szolgáltatásokat. Másik ilyen program, amely szintén esélyes lehet erre, az a **named(8)**. Az `/etc/defaults/rc.conf` megjegyzésben tartalmazza a **named** járókában futtatásához szükséges paramétereket. Attól függően, hogy egy új rendszert telepítünk vagy frissítjük a már meglévő rendszerünket, a járókákhoz tartozó speciális felhasználói hozzáférések nem feltétlenül jönnek létre. Amikor csak lehetséges, az előrelátó rendszergazda kikísérletez és létrehoz ilyen járókákat.

Vannak más olyan szerverek, amelyek tipikusan nem járókákban futnak. Ilyen többek közt a **sendmail**, **popper**, **imapd**, **ftpd** és még sokan mások. Léteznek rájuk alternatívák, de a telepítésük valószínűleg több munkát igényel, mint amennyit megérné számunkra vesződni velük (és itt megint lesújt a kényelmi tényező). Ezeket a szervereket többnyire `root` felhasználóként kell futtatnunk és a rajtuk keresztül érkező betörési kísérleteket más módokra támaszkodva kell észlelnünk.

A `root` felhasználó keltette biztonsági rések másik nagy csoportja azok a végrehajtható állományok a rendszerben, amelyek a `suid` és `sgid` engedélyekkel rendelkeznek, futtatásuk rendszergazdai jogokkal történik. Az ilyen binárisok többsége, mint például az **rlogin**, a `/bin` és `/sbin`, `/usr/bin` vagy `/usr/sbin` könyvtárakban található meg. Habár semmi sem biztonságos 100%-ig, a rendszerben alapértelmezetten `suid` és `sgid` engedéllyel rendelkező binárisok ebből a szempontból meglehetősen megbízhatónak tekinthetők. Alkalmanként azonban találunk a `root` felhasználót veszélyeztető lyukakat az ilyen binárisokban is. Például 1998-ban az `xlib`-ben volt egy olyan rendszergazdai szintű hiba, amellyel az **xterm** (ez általában `suid` engedéllyel rendelkezik) sebezhetővé vált. Mivel jobb félni, mint megijedni, ezért az előretekinő rendszergazda mindig igyekszik úgy csökkenteni az ilyen engedélyekkel rendelkező binárisok körét, hogy csak a személyzet tagjai legyenek képesek ezeket futtatni. Ezt egy olyan speciális csoport létrehozásával oldhatjuk meg, amelyhez csak a személyzet tagjai férhetnek hozzá. Az olyan `suid` binárisoktól pedig, amelyeket senki sem használ, igyekszik teljesen megszabadulni (`chmod 000`). A monitorral nem rendelkező szervereknek általában nincs szükségük az **xterm** működtetésére. Az `sgid` engedéllyel rendelkező binárisok is legalább ugyanennyire veszélyesek. Ha a behatoló képes feltörni egy `kmem` csoporthoz tartozó `sgid` binárist, akkor képes lesz olvasni a `/dev/kmem` állomány tartalmát, ezáltal hozzájut a titkosított jelszavakhoz és így megszerezheti magának akármelyik hozzáférést. Sőt, a `kmem` csoportot megszerző behatolók figyelni tudják a pszeudó terminálokon keresztül érkező billentyűleütéseket, még abban az esetben is, amikor a felhasználók egyébként biztonságos módszereket használnak. A `tty` csoportot bezsebelő támadók szinte bármelyik felhasználó termináljára képesek írni. Ha a felhasználó valamilyen terminál programot vagy terminál emulátort használ a billentyűzet szimulációjával, akkor a behatoló tud olyan adatokat generálni, amivel a felhasználó nevében adhat ki parancsokat.

14.3.3. A felhasználói hozzáférések védelme

A felhasználók hozzáféréseit szinte a legnehezebb megvédeni. Míg a személyzet tagjaival szemben lehetünk kíméletlenül szigorúak és “ki is csillagozhatjuk” a jelszavakat, addig a felhasználók hozzáféréseivel általánosságban véve ezt nem tehetjük meg. Ha a kezünkben van a megfelelő mértékű irányítás, akkor még győzhetünk és kényelmesen biztonságba helyezethetjük a felhasználók hozzáféréseit. Ha nincs, akkor nem tehetünk mást, mint állandóan őrködünk a hozzáférések felett. Az ssh és Kerberos használata a felhasználók esetén sokkalta problematikusabb, mivel ilyenkor jóval több adminisztrációra és műszaki segítségnyújtásra van szükség, de még mindig jobb megoldás a titkosított jelszavakhoz képest.

14.3.4. A jelszavakat tároló állomány védelme

Az a legbiztosabb, ha minél több jelszót kicsillagozunk és a hozzáférések hitelesítésére ssh-t vagy Kerberost használunk. Igaz, a titkosított jelszavakat tároló állományt (`/etc/spwd.db`) csak a `root` képes olvasni, de a támadó meg tudja szerezni ezt a jogot még olyankor is, ha `root` felhasználóként nem feltétlenül tud írni.

A rendszerünkben futó biztonsági szkripteknek a jelszavakat tároló állomány változását folyamatosan tudnia kell figyelnie és jelentie (lásd lentebb a Az állományok sértetlenségének ellenőrzése című fejezetet).

14.3.5. A rendszermag belsejének, a nyers eszközök és az állományrendszerek védelme

Ha a támadó megszerzi a `root` hozzáférést, akkor szinte bármit képes megtenni, de vannak bizonyos előnyei. Például a mostanság fejlesztett legtöbb rendszermag tartalmaz valamilyen beépített csomaglehallgatót, amit FreeBSD alatt a `bpf` eszköz valósít meg. A támadók szinte mindig megpróbálnak valamilyen csomaglehallgatót használni a feltört gépen. A legtöbb rendszeren azonban nem kell feltétlenül megadnunk ezt az örömet, ezért nem is kell beépítenünk a rendszermagba a `bpf` eszközt.

De ha még ki is iktatjuk a `bpf` eszközt, még aggódhatunk a `/dev/mem` és `/dev/kmem` miatt. Egyébként ami azt illeti, a behatoló még így is képes írni a nyers eszközökre. Sőt, a rendszermagba képesek vagyunk modulokat is betölteni a `kldload(8)` használatával. A vállalkozó kedvű támadó a rendszermag moduljaként képes telepíteni és használni a saját `bpf` eszközét vagy bármilyen más, a csomagok lehallgatására alkalmas eszközt. Az ilyen problémák elkerülése érdekében a rendszermagot a legmagasabb védelmi szinten kell üzemeltetni, tehát legalább egyes szinten.

A rendszermag védelmi szintjét több különböző módon lehet állítani. A védelmi szintet úgy lehet a legegyszerűbben növelni, ha a `sysctl` paranccsal beállítjuk a `kern.securelevel` nevű, rendszerszintű változó értékét:

```
# sysctl kern.securelevel=1
```

A FreeBSD rendszermag alapértelmezés szerint a `-1` védelmi szinten indul. Ez egészen addig `-1` marad, amíg a rendszergazda vagy valamelyik `init(8)` során hívott rendszerindító szkript ezt meg nem változtatja. A rendszer indítása során úgy tudjuk beállítani a megfelelő védelmi szintet, ha az `/etc/rc.conf` állományban megadjuk a `kern.securelevel_enable` változót a `YES` értékkel, illetve `kern.securelevel` értékeként a kívánt védelmi szintet.

A FreeBSD alapértelmezett védelmi szintje közvetlenül a rendszerindító szkriptek lefutása után `-1`. Ezt “nem biztonságos módnak” nevezik, mivel az állományok írásáért felelős állományjelzők nem feltétlenül működnek, mindegyik eszköz írható, olvasható és a többi.

Miután a védelmi szintet `1` vagy annál magasabb értékre állítottuk, akkor a rendszer figyelembe veszi a csak hozzáfűzést (`append-only`) és módosíthatatlanságot (`immutable`) megszorító állományjelzőket, nem engedélyezi a

tiltásukat és az eszközök közvetlenül nem érhetőek el. A különböző védelmi szintek részletesebb bemutatását a `security(7)` man oldalon olvashatjuk (vagy a FreeBSD 7.0 előtti változataiban a `init(8)` man oldalon).

Megjegyzés: Az 1 és az afeletti védelmi szinteken többek közt az X11 nem feltétlenül lesz futtatható (mivel a `/dev/io` eszköz elérése blokkolt), illetve a rendszer frissítése is akadályokba fog ütközni (a `installworld` futtatása során ideiglenesen ki kell kapcsolni az append-only és immutable állományjelzőket). Az X11 esetében ezt valahogy még ki lehet kerülni úgy, hogy ha az `xm(1)` démont még a rendszerindítás elején aktiváljuk (amikor a védelmi szint még kellően alacsony). Az összes védelmi szint és megszorítás esetén azonban nem mindig adható ilyen jellegű javaslat, ezért ilyenkor mindig érdemes előre tervezni egy keveset. Emellett fontos alaposan megismerni a különböző védelmi megszorításokat, mivel jelentős mértékben visszafoghatják a rendszer használhatóságát. Ez segít az adott helyzetben az egyszerűbb megoldást választani és ezáltal elkerülni a kellemetlen meglepetéseket.

Ha a rendszermag védelmi szintjét az 1 érték vagy afelé emeljük, akkor hasznos lehet a fontosabb (lényegében minden olyan programnak, amely a védelmi szint helyes beállítódása előtt lefut) programoknak, könyvtáraknak és szkripteknek beállítani az `schg` állományjelzőt. Ilyenkor azonban vegyük figyelembe, hogy a rendszer frissítése is nehezebbé válik a magasabb védelmi szinteken. Egy működőképesebb megoldás lehet, ha rendszerünket egy magasabb védelmi szinten használjuk, de nem állítjuk be mindegyik rendszerszintű állományra az `schg` állományjelzőt. Másik lehetőség még a `/` és `/usr` partíciók írásvédett csatlakoztatása. Ne felejtjük el azonban, hogy ha túlságosan szigorúak vagyunk magunkhoz, akkor azzal egyúttal a behatolás észlelését is meg tudjuk nehezíteni!

14.3.6. Az állományok sértetlenségének ellenőrzése: binárisok, konfigurációs állományok stb.

Ha arról van szó, csak a legfontosabb rendszerszintű konfigurációs- és vezérlőállományokat tudjuk megvédeni, még mielőtt a korábban emlegetett kényelmi tényező kimutatná a foga fehérjét. Például, ha a `chflags` paranccsal beállítjuk az `schg` állományjelzőt a `/` és `/usr` állományrendszereken található legtöbb állományra, akkor az minden bizonnyal csökkenti a hatékonyságunkat, hiszen az állományok védelmének növekedésével csökken az észlelés lehetősége. A védelmi vonalaink közül ugyanis az utolsó talán az egyik legfontosabb — a detektálás. A felépített biztonsági rendszerünk legnagyobb része szinte teljesen hasztalan (vagy ami még rosszabb, a biztonság hamis érzetét kelti), ha nem vagyunk képesek észrevenni a betörési kísérleteket. A védelmi rendszer egyik részére nem a támadó megállításához, hanem a lelassításához van szükség, hogy így majd munka közben érhezzük tetten.

A betörés tényét legjobban a megváltozott, hiányzó vagy éppen váratlanul felbukkanó állományok utáni kutatással tudjuk felismerni. A módosított állományokat általában egy másik (gyakran központosított) korlátozott hozzáférésű rendszerből ellenőrizhetjük a legjobban. Fontos, hogy ha egy korlátozott hozzáférésű, kiemelten védett rendszeren írjuk a védelemért felelős szkripteket, akkor azok szinte teljesen láthatlanok lesznek a támadó számára. A legjobb kihasználás érdekében a korlátozott hozzáférésű gépnek jelentős mértékű rálátással kell rendelkeznie az összes többi gépre, amit írásvédett NFS exportok vagy ssh kulcspárok felhasználásával érhetünk el. A hálózati forgalmat leszámítva az NFS látszik a legkevésbé — segítségével lényegében észrevétlenül tudjuk figyelni az egyes gépek állományrendszereit. Ha a megfigyelésre használt szerver a kliensekhez switchen keresztül csatlakozik, akkor az NFS gyakran jobb választásnak bizonyul. Ha a szerver hubon vagy több hálózati elemen keresztül éri el a megfigyelni kívánt klienseket, akkor az NFS nem eléggé biztonságos (és hatékony), ezért ilyen esetekben az ssh választása lehet a kedvező még az ssh által hagyott nyomokkal együtt is.

Miután a korlátozott hozzáférésű gépünk legalább látja a hozzá tartozó kliensek rendszereit, el kell készítenünk a tényleges monitorozást végző szkripteket. Ha NFS csatlakozást tételezünk fel, akkor az olyan egyszerű rendszerezőkkel, mint például a `find(1)` és `md5(1)` képesek vagyunk összerakni ezeket. A szemmel tartott

kliensek állományait naponta legalább egyszer érdemes ellenőrizni md5-tel, valamint még ennél gyakrabban is tesztelni az `/etc` és `/usr/local/etc` könyvtárakban található konfigurációs és vezérlőállományokat. Ha valamilyen eltérést tapasztal az ellenőrzést végző szervertől és a rajta levő md5 információk is helyesek, akkor értesítenie kell a rendszergazdát. Egy jó védelmi szkript képes megkeresni az oda nem illő suid binárisokat, valamint az új vagy törölt állományokat a `/` és a `/usr` partíciókon.

A védelmi szkriptek megírása valamivel nehezebb feladat, ha ssh-t használunk az NFS helyett. A futtatásukhoz a szkripteket és az általuk használt eszközöket (például `find`) az `scp` paranccsal lényegében át kell másolni a kliensekre, amivel így láthatóvá válnak. Ne feledjük továbbá, hogy az **ssh** kliens már eleve feltört lehet. Szó, ami szó, ha nem megbízható összeköttetésekről beszélünk, akkor az ssh használata elkerülhetetlen, de nem feltétlenül egyszerű.

Egy jó védelmi szkript észreveszi a felhasználók és a személyzet tagjainak hozzáférését vezérlő állományokban, mint például az `.rhosts`, `.shosts`, `.ssh/authorized_keys` és társaiban keletkezett változásokat is, amelyek esetleg elkerülhetik egy MD5 alapú ellenőrzés figyelmét.

Ha netalán órasi mennyiségű tárterülettel rendelkezünk, akkor eltarthat egy ideig, amíg végigsöprünk az összes partíció összes állományán. Ebben az esetben érdemes olyan beállításokat megadni az állományrendszerek csatlakoztatásánál, amivel le tudjuk tiltani a suid engedéllyel rendelkező binárisok futtatását. Ezzel kapcsolatban a `mount(8)` parancs `nosuid` opcióját nézzük meg. Hetente legalább egyszer azért mégis érdemes átnézni az ilyen partíciókat is, mivel ez a réteg a betörési kísérletek felderítésével foglalkozik, függetlenül a sikerességüktől.

A futó programok nyilvántartása (lásd `accton(8)`) egy olyan viszonylag kevés költséggel járó lehetőség az operációs rendszerben, ami segítségünkre lehet a betörés utáni események kiértékelésében. Különösen hasznos olyankor, amikor megpróbáljuk modellezni, miképp is sikerült a támadónak bejutnia a rendszerünkbe, természetesen feltételezve, hogy az ehhez felhasznált feljegyzések a betörés után is érintetlenek maradtak.

Végül a védelmet ellátó szkripteknek javasolt feldolgozni a naplóállományokat is, valamint a naplókat magukat is a lehető legbiztonságosabb formában generálni — ilyenkor nagyon hasznos lehet, ha egy távoli gépre naplózunk. A behatoló megpróbálja majd eltüntetni a nyomait, a naplóállományok viszont nagyon fontosak a rendszergazda számára a betörési kísérletek idejének és módjának megállapításában. A naplókat úgy tudjuk tartósan rögzíteni, ha a rendszerkonzol üzeneteit soros porton keresztül gyűjtjük össze a konzolok felügyeletéért felelős biztonságos gépen.

14.3.7. Állandó paranoia

Egy kis paranoia sosem árt. Elmondható, hogy a rendszergazda tetszőleges számú biztonsági intézkedéssel élhet egészen addig, amíg az nincs hatással a kényelmére, és a kényelmet *befolyásoló* biztonsági intézkedéseket pedig megfelelő mérlegelés mellett tegye meg. Ami még ennél is fontosabb, hogy mindig változtassunk valamit a biztonsági hálónkon — mivel ha egy az egyben követjük a dokumentumban leírtakat, akkor ezzel együtt kiadjuk a bejutás receptjét annak a leendő támadónknak, aki szintén elolvasta ugyanezt.

14.3.8. A szolgáltatások működésképtelenné tételét célzó támadások

Ez a szakasz a szolgáltatások működésképtelenségét elérni kívánó, más néven “Denial of Service” típusú támadásokkal foglalkozik. Noha nem tudunk túlságosan sokat tenni a manapság felbukkanó álcázott, a hálózatunk totális leterhelését célbavevő támadások ellen, akadnak olyan általános érvényű eszközök, amelyekkel elejét vehetjük a szervereink szétbomzásának:

1. A létjövő szerverpéldányok korlátozása.
2. Az ugródeszkaszerű támadások (támadás ICMP-válasszal, pingszórás stb.) korlátozása.

3. A rendszermag útválasztási gyorsítótárának túlterhelése.

A DoS támadások egyik jellemző sémája szerint egy sokszorozódni képes szervert támadnak meg, amelynek igyekeznek minél több példányát legyártatni, míg végül az ezt futtató rendszer ki nem fogy a memóriából, állományleíróból satöbbiből és megállásra nem kényszerül. Az **inetd** (lásd `inetd(8)`) számos lehetőséget kínál ennek megakadályozására. Ezzel kapcsolatban szeretnénk megjegyezni, hogy bár ezzel el tudjuk kerülni a gépünk leállítását, semmilyen garanciát nem ad arra, hogy a szolgáltatás a támadás során is zavartalanul üzemel tovább. Alaposan olvassuk el az **inetd** man oldalát és legyünk különös tekintettel a `-c`, `-C` és `-R` kapcsolóira. Vigyázzunk, hogy az **inetd** `-C` kapcsolóját képesek kijátszani az álcázott IP-vel érkező támadások, ezért inkább az előbbi kapcsolók valamilyen kombinációja az ajánlott. Egyes szerverprogramoknál be lehet állítani a példányainak maximális számát.

A **Sendmail** rendelkezik egy `-OMaxDaemonChildren` beállítással, ami a terhelésben levő késleltetése miatt néha mintha jobban beválna, mint a **Sendmail** terheléskorlátozó paraméterei. A **Sendmail** indításakor tehát a `MaxDaemonChildren` paramétert javasolt megadni egy olyan értékkel, amely elegendő a **Sendmail** számára betervezett terhelés kiszolgálására, de még kevés ahhoz, hogy a **Sendmail** fűbe harapjon tőle. Továbbá bölcs dolog a **Sendmail** várakozási sorral (`-ODeliveryMode=queued`) és démonként (`sendmail -bd`), külön feldolgozási menetekkel (`sendmail -q15m`) futtatni. Ha továbbra is valós idejű kézbesítést akarunk, akkor a feldolgozást kisebb időközökkel is lefuttathatjuk (például `-q1m`), de arra *mindig ügyeljünk*, hogy a `MaxDaemonChildren` beállítása ne okozzon kaszkádosítási hibákat a **Sendmail** működésében.

A **Syslogd** közvetlenül is támadható, ezért határozottan javasoljuk a `-s` használatát, amikor csak lehet, minden más esetben pedig a `-a` beállítást.

Fordítsunk kellő figyelmet a TCP kapcsolatok burkolását végző **TCP Wrapper** “reverse-ident” lehetőségére, ami szintén közvetlenül támadható. Ebből az okból kifolyólag valószínűleg nem is akarjuk a **TCP Wrapper** által felkínált reverse-ident-et használni.

Jól járunk el abban az esetben, ha a belső szolgáltatásainkat az útválasztóink mentén tűzfal segítségével védjük meg a külső hozzáféréstől. Ezzel lényegében a helyi hálózatunkat kívülről fenyegető támadások ellen védekezünk, de ez nem nyújt elegendő védelmet a belső szolgáltatásaink esetén a `root` hozzáférés megszerzésére irányuló kísérletek ellen. Mindig egy exkluzív, tehát zárt tűzfalat állítsunk be, vagyis “tűzfalazzunk mindent *kivéve* az A, B, C, D és M-Z portokat”. Ezen a módon ki tudjuk szűrni az összes alacsonyabb portot, kivéve bizonyos eseteket, mint például a **named** (ha az adott zónában ez az elsődleges gép), **ntalkd**, **sendmail** vagy más interneten keresztül elérhető szolgáltatásokat. Ha másképpen állítjuk a tűzfalat — inkluzív, nyílt vagy megengedő módon, akkor jó eséllyel elfelejtünk “lezárni” egy csomó szolgáltatást, vagy úgy adunk hozzá egy új belső szolgáltatást, hogy közben elfelejtjük frissíteni a tűzfalat. Ennél még azon is jobb, ha a tűzfalon nyitunk egy magasabb portszámú tartományt, és ott valósítjuk meg ezt a megengedő jellegű működést, az alacsonyabb portok veszélybe sodrása nélkül. Vegyük azt is számításba, hogy a FreeBSD-ben a kiosztott portokat dinamikusan állíthatjuk a `net.inet.ip.portrange` sysctl változókon keresztül (`sysctl -a | fgrep portrange`), ami nagyságrendekkel megkönnyíti a tűzfal beállítását. Ennek megfelelően például meg tudjuk adni, hogy a 4000-től 5000-ig terjedő porttartomány a 49152-től 65535-ig húzódó tartományba kerüljön át, majd a 4000 alatti összes portot blokkoljuk (természetesen az internetről szándékosan hozzáférhető portok kivételével).

A DoS támadások másik elterjedt fajtája az ún. “ugródeszka támadás” — ilyenkor a szervert úgy próbálják túlterhelni, hogy folyamatosan válaszokat kérnek tőle a helyi hálózatról vagy egy másik számítógépről. Az ilyen természetű támadások közül is a legnépszerűbb az *ICMP pingszórásos támadás*. A támadó olyan ping csomagokat küld szét a helyi hálózaton, amelyek forrásának azt a gépet jelöli meg, amelyiket meg akarja támadni. Ha a hálózatokat elválasztó útválasztók nem fogják meg a pingszórást, akkor a helyi hálózatról összes gépe nekilát válaszolni a meghamisított forrás címére, amivel így teljesen leterhelik az áldozatot. Ez különösen akkor hatásos, amikor a támadó ugyanezt a trükköt eljátssza egyszerre több tucat különböző hálózatban is. Az üzenetszórással járó

támadások akár százhusz megabitnyi forgalmat is képesek generálni másodpercenként. A második legelterjedtebb ugródeszkás támadás az ICMP hiba-visszajelzési rendszere ellen irányul. Ilyenkor a támadó ICMP hibaüzeneteket kiváltó csomagok készítésével képes eltömíteni egy szerver bejövő hálózati kapcsolatát és az ICMP válaszokkal pedig a szerver maga dugítja el a kimenő hálózati kapcsolatát. Ez a fajtájú támadás képes kinyomni az összes memóriát a szerverből és ezzel összeomlasztani, különösen olyankor, amikor a szerver nem tudja elég gyorsan elnyelni az általa generált ICMP válaszokat. A `net.inet.icmp.icmplim sysctl` változóval tudunk gátat szabni a támadások ezen fajtájának. Az ugródeszkás támadások utolsó nagyobb osztálya az **inetsd** olyan szolgáltatásait szemeli ki, mint például az `udp echo`. A támadó ilyenkor egyszerűen küld a helyi hálózatunkon található A és B szerverünknek egy olyan UDP csomagot, ahol forrásként az A szerver `echo` portját adja meg, célként pedig a B szerver `echo` portját. Ezután a két szerver elkezd egymás között passzolgatni ezt az egyetlen csomagot. A támadó még több ilyen csomag befecskendezésével pillanatok alatt képes leterhelni a két szervert és helyi hálózatot. Hasonló problémák vannak a belső **chargen** portjával is. Egy hozzáértő rendszergazda ezért kikapcsolja az összes ilyen `inetsd`-alapú belső tesztelő szolgáltatást.

Az álcázott csomagok felhasználhatóak a rendszermag útválasztó gyorsítótárának túlterhelésére is. Ezzel kapcsolatban nézzük meg a `net.inet.ip.rtexpire`, `rtminexpire` és `rtmaxcache sysctl` változókat. A véletlenszerű IP-címekkel megcímezett álcázott csomagok hatására a rendszermag létrehoz mindegyikükhöz egy ideiglenesen puffertelt utat az útválasztó táblázatában, amelyet a `netstat -rna | fgrep W3` paranccsal tudunk lekérdezni. Az ilyen útvonalak nagyjából 1600 másodperc múlva elévülnek. Ha a rendszermag észleli, hogy a gyorsítótárazott útválasztási táblázat mérete túlságosan megnövekedett, akkor automatikusan csökkenti az `rtexpire` értékét, de soha nem megy a `rtminexpire` alá. Ebből két probléma adódik:

1. A rendszermag nem reagál elég gyorsan amikor egy alig terhelt szervert hirtelen megtámadnak.
2. Az `rtminexpire` nem elég kicsi ahhoz, hogy a rendszermag túléljen egy tartósabb rohamot.

Ha a szervereink az internethez T3 (kb. 45 Mbit/s) vagy gyorsabb összeköttetésen keresztül csatlakoznak, akkor határozottan javasolt kézzel behangolni a `sysctl(8)` segítségével az `rtexpire` és az `rtminexpire` értékeket. Soha ne állítsuk egyiket sem nullára (hacsak nem akarjuk összeomlasztani a gépünket). Ha például mind a kettőt 2 másodpercre állítjuk, akkor az többnyire elegendő az útválasztási táblázat megvédéséhez.

14.3.9. Hozzáférés Kerberoszal és SSH-val

Van néhány dolog, amit a Kerberos és az ssh esetén ajánlatos tisztázni, mielőtt használjuk ezeket. A Kerberos 5 egy kifogástalan hitelesítési protokoll. A **telnet** és **rlogin** Kerberos által módosított változatában vannak olyan hibák, amelyek alkalmatlanná teszik ezeket a bináris adatfolyamok helyes kezelésére. Sőt, alapértelmezés szerint a Kerberos nem titkosítja a kapcsolatot, csak ha megadjuk neki a `-x` kapcsolót. Az **ssh** alapértelmezés szerint mindent titkosít.

Az ssh minden szempontból nagyon jól teljesít kivéve, hogy alapértelmezés szerint átküldi a kulcsokat is. Ez azt jelenti, hogy ha van egy olyan biztonságos munkaállomásunk, ahol a rendszer többi részéhez tartozó kulcsainkat tartjuk és egy nem biztonságos gépre akarunk vele ssh-n keresztül belépni, akkor a kulcsaink használatává válnak. A tényleges kulcsokat ugyan nem látja senki, de a bejelentkezés során az ssh megnyit egy közvetítéshez használt portot, amit a nem biztonságos gépen a támadó egy feltört `root` hozzáférés birtokában ki tud használni úgy, hogy a kulcsaink segítségével hozzá tudjon férni egy másik olyan géphez, amelyet a kulcsok nyitnak.

Ha lehetséges, akkor a személyzet bejelentkeztetéséhez az ssh-t és Kerberost együttesen használjuk. Az **ssh** lefordítható Kerberos támogatással. Ezzel csökkentjük a potenciálisan kiszivárgó ssh kulcsok esélyét, miközben jelszavainkat a Kerberoszal védjük. Az ssh kulcsokat csak biztonságos gépekről és csak automatizált feladatok esetén használjuk (amire a Kerberos lényegében nem alkalmas). Emellett javasoljuk azt is, hogy az ssh beállításai

között tiltsuk le a kulcsok átküldését (key forwarding) vagy használjuk az `from=IP/DOMAIN` opciót, amivel az ssh csak a megadott gépekről engedi az `authorized_keys` állomány és a így benne levő kulcsok használatát.

14.4. DES, Blowfish, MD5 és a Crypt

Egyes részeit újraírta és aktualizálta: Bill Swingle.

Minden UNIX rendszer használójához tartozik egy jelszó is a hozzáféréséhez. Teljesen nyilvánvalónak tűnik, hogy ezt a jelszót csak az adott felhasználó és az adott operációs rendszer ismeri. A jelszavakat a titokban tartásukhoz ún. “csapóajtó függvényekkel” titkosítják, amelyeket könnyű titkosítani, ám nehéz visszafejteni. Tehát amit egy perccel ezelőtt még nyilvánosnak tituláltunk, az mostanra már nem is teljesen igaz: *valójában* az operációs rendszer sem ismeri a jelszót. Az operációs rendszer csak a jelszó *titkosított* változatát ismeri. A jelszó “titkosítatlan” formáját csak nyers erő igénybevételel tudjuk megkeresni az összes lehetséges jelszó szénakazlában.

Sajnos, annak idején, amikor a jelszavak titkosítása bekerült a UNIX-ba, egyedül a DES, vagy más néven a Data Encryption Standard (Adattitkosítási szabvány) jött szóba. Ez alapvetően nem jelentett problémát az Egyesült Államok állampolgárai számára, de mivel a DES forráskódját nem lehetett kivinni az Egyesült Államokból, a FreeBSD-nek találnia kellett valami olyasmit, ami mind megfelel az Egyesült Államok törvényeinek, mind pedig kompatibilis marad az összes többi DES-t használó UNIX variánsal.

Ezt úgy oldották meg, hogy felosztották a titkosítással foglalkozó függvénykönyvtárakat, így az Egyesült Államokban élő felhasználók tudtak DES könyvtárakat telepíteni és használni, miközben a többi nemzet felhasználói olyan más titkosítási módszert tudtak választani, amit kinn is lehetett alkalmazni. Ennek tulajdonítható, hogy a FreeBSD alapértelmezés szerint az MD5 segítségével titkosít. Az MD5-öt a DES-nél sokkalta biztonságosabbnak tartják, ezért a DES telepítésének lehetőségét leginkább csak kompatibilitási okokból ajánlották fel.

14.4.1. A titkosítási mechanizmus azonosítása

Jelenleg a könyvtár ismeri a DES, MD5 és Blowfish függvényeit. A FreeBSD a jelszavak titkosításához alapból az MD5-öt használja.

Nagyon könnyen meg tudjuk mondani, hogy a FreeBSD éppen melyik titkosítási módszert alkalmazza. Ennek egyik lehetősége, ha az `/etc/master.passwd` állományt vizsgáljuk meg. Az MD5 függvényével titkosított jelszavak hosszabbak, mint a DES függvényével titkosítottak és a `1` karakterekkel kezdődnek. A `$2a$` karakterekkel kezdődő jelszavakat Blowfish-sel titkosították. A DES kódolású jelszavaknak nincs semmilyen különleges ismertetőjelük, de általánosságban elmondható róluk, hogy rövidebbek az MD5 jelszavaknál és olyan 64 karakteres ábécével kódolják ezeket, amelyek nem tartalmazzák a `$` karaktert, így tehát a viszonylag rövid, nem dollárjellel kezdődő karakterláncok minden bizonnyal DES kódolású jelszavak.

Az új jelszavak kódolásához használt formátumot az `/etc/login.conf` állományban tárolt `passwd_format` bejelentkezési tulajdonság adja meg, amelynek értékei `des`, `md5` vagy `blf` lehetnek. A `login.conf(5)` man oldalon tájékozódhatunk bővebben a bejelentkezési tulajdonságokról.

14.5. Egyszeri jelszavak

A FreeBSD alapértelmezés szerint támogatja az OPIE-t (One-time Passwords In Everything, azaz “Egyszeri jelszavak mindenben”), ami alapból az MD5 függvényét használja.

A jelszavak három fajtáját fogjuk a továbbiakban tárgyalni. Az első a megszokott UNIX stílusú avagy Kerberos jelszó. Ezt a továbbiakban “UNIX jelszónak” nevezzük. A második fajtában az OPIE `opiekey(1)` nevű segédprogramja által generált és a bejelentkezésnél a `opiepasswd(1)` által elfogadott jelszavak tartoznak. Ezeket “egyszeri jelszavaknak” fogjuk nevezni. A jelszavak utolsó típusa az a titkos jelszó, amit az `opiekey` programnak (és néha a `opiepasswd` programnak) adunk meg, ami ebből egyszer használatos jelszavakat állít elő. Ezt innentől “titkos jelszónak” vagy csak egyszerűen “jelszónak” hívjuk.

A titkos jelszónak semmi köze sincs a UNIX jelszavunkhoz. Természetesen megegyezhetnek, de ezt nem ajánljuk. Az OPIE által használt titkos jelszavaknak nem kell a régi UNIX jelszavakhoz hasonlóan legfeljebb 8 karakteresnek lenniük ¹, bármikorát használhatunk. A hat vagy hét szóból álló jelszavak ilyenkor igen gyakoriak. Az OPIE jobbra a UNIX jelszórendszerétől teljesen függetlenül működik.

A jelszavak mellett két másik fajta adat fontos az OPIE számára. Közülük az egyiket “magnak” vagy “kulcsnak” nevezik, ami két betűből és öt számjegyből áll. A másik az “iterációk száma”, ami egy 1 és 100 közötti számot takar. Az OPIE úgy hozza létre az egyszeri jelszavakat, hogy egymás után fűzi a magot és a titkos jelszót, majd az iterációk megadott számának megfelelő mennyiségben kiszámolja rá az MD5 függvény értékét és az eredményt hat rövid angol szóba önti. Ez a hat angol szó lesz a mi egyszeri jelszavunk. A hitelesítéssel foglalkozó rendszer (elsősorban a PAM) figyelemmel kíséri a legutoljára használt egyszeri jelszavunkat, és csak akkor engedi a felhasználót hitelesíteni, ha az általa megadott jelszó kódolt változata megegyezik az előzőleg megadott jelszavával. A csapóajtó függvények használata miatt lehetetlen legenerálni a következő egyszeri jelszót, ha a sikerült megszereznünk az egyiket. Az iterációk száma minden egyes sikeres bejelentkezés után csökken eggyel, amivel a felhasználót és a bejelentkeztető programot szinkronban tartja. Amikor így az iterációk száma eléri az egyet, az OPIE-t újra kell inicializálni.

Az említésre kerülő rendszerek mindegyikéhez tartozik néhány program. Az `opiekey` bekéri az iterációk számát, a magot és a titkos jelszót, majd előállít egy egyszer használatos jelszót vagy azok folytonos listáját. Az `opiepasswd` az OPIE inicializálásért, a jelszavak, az iterációk számának és a mag megváltoztatásáért felelős. Egyaránt elfogad titkos jelmondatot, iterációs számot vagy magot és egy egyszeri jelszót. Az `opieinfo` megvizsgálja a felhasználókra vonatkozó adatbázist (`/etc/opiekeys`) és kiírja az adott felhasználó által használt iterációs számot és magot.

Négyféle különböző műveletről fogunk most itt beszélni. Az elsőben egy biztonságos kapcsolaton keresztül elsőként inicializáljuk az egyszeri jelszavakat, vagy megváltoztatjuk a jelszót vagy a magot az `opiepasswd` segítségével. A második műveletben ugyanarra adjuk ki az `opiepasswd` parancsot egy nem biztonságos kapcsolaton keresztül az `opiekey` paranccsal együtt egy biztonságos kapcsolaton keresztül. A harmadikban az `opiekey` használatával nem biztonságos kapcsolaton keresztül jelentkezünk be. A negyedikben az `opiekey` paranccsal létrehozunk egy adott mennyiségű kulcsot, amelyeket aztán leírhatunk vagy kinyomtathatunk, hogy magunkkal tudjuk vinni olyan helyre, ahonnan nem tudnk biztonságos módon csatlakozni.

14.5.1. Inicializálás biztonságos kapcsolattal

Az OPIE első inicializálásához adjuk ki az `opiepasswd` parancsot:

```
% opiepasswd -c
[grimreaper] ~ $ opiepasswd -f -c
Adding unfurl:
Only use this method from the console; NEVER from remote. If you are using
telnet, xterm, or a dial-in, type ^C now or exit with no password.
Then run opiepasswd without the -c parameter.
Using MD5 to compute responses.
Enter new secret pass phrase:
Again new secret pass phrase:
```

```
ID unfurl OTP key is 499 to4268
MOS MALL GOAT ARM AVID COED
```

A figyelmeztetés fordítása:

Ezt a módszert csak konzolról alkalmazzuk, SOHA ne távoli kapcsolaton keresztül! Ha telnetet, xtermet vagy betárcsázós kapcsolatot használunk, akkor azonnal nyomjunk ^C-t vagy ne adjunk meg jelszót.

Az Enter new secret pass phrase: vagy Enter secret password: kérdések után adjunk meg egy jelmondatot, illetve jelszót. Ne felejtsük el, hogy ez nem bejelentkezéshez használt jelszó lesz, hanem ebből jönnek majd létre az egyszeri kulcsaink. Az "ID" sor adja meg az aktuális példányunk paramétereit: a bejelentkezéshez használt nevünket, az iterációk számát és a magot. Amikor a bejelentkezések során a rendszer emlékszik a paraméterekre és megjeleníti ezeket, nem kell megjegyeznünk. Az utolsó sor adja meg a paramétereinknek és a titkos jelszavunknak megfelelő egyszeri jelszót. Ha most azonnal akarnánk bejelentkezni, akkor ezt az egyszeri jelszót kellene hozzá használnunk.

14.5.2. Inicializálás nem biztonságos kapcsolattal

Ha egy nem biztonságos kapcsolaton keresztül akarjuk inicializálni vagy megváltoztatni a jelszavunkat, akkor szükségünk lesz valahol egy megbízható kapcsolatra, ahol le tudjuk futtatni az `opiekey` parancsot. Ez lehet egy számunkra biztonsági szempontból elfogadható gép parancssora. Emellett ki kell találnunk egy iterációs számot (erre a 100 egy jó választás) és adnunk egy magot vagy használni egy véletlenszerűen generáltat. Az inicializálás színtere felé vezető nem biztonságos kapcsolaton keresztül adjuk ki az `opiepasswd` parancsot:

```
% opiepasswd

Updating unfurl:
You need the response from an OTP generator.
Old secret pass phrase:
    otp-md5 498 to4268 ext
    Response: GAME GAG WELT OUT DOWN CHAT
New secret pass phrase:
    otp-md5 499 to4269
    Response: LINE PAP MILK NELL BUOY TROY

ID mark OTP key is 499 gr4269
LINE PAP MILK NELL BUOY TROY
```

Az alapértelmezett mag elfogadásához nyomjuk le a **Return** billentyűt. Mielőtt megadnánk a hozzáférés jelszavát, menjünk át a biztonságos kapcsolatra és adjuk meg neki ugyanezeket a paramétereket:

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

Most váltsunk vissza a nem biztonságos kapcsolatra és másoljuk be az így generált egyszeri jelszót a megfelelő programba.

14.5.3. Egyetlen egyszeri jelszó létrehozása

Miután sikeresen inicializáltuk az OPIE-t és bejelentkeztünk, a következőket láthatjuk:

```
% telnet example.com
Trying 10.0.0.1...
Connected to example.com
Escape character is '^['.

FreeBSD/i386 (example.com) (tty)

login: felhasználói_név
otp-md5 498 gr4269 ext
Password:
```

Mellékesen megjegyezzük, hogy az OPIE parancssorának van egy (itt nem látható) hasznos képessége: ha **Return** billentyűt nyomunk a jelszó bekérésekor, akkor a program megmutatja a begépett betűket, így láthatjuk pontosan mit is írunk be. Ez nagyon kényelmes lehet olyankor, amikor valahonnan, például egy lapról olvassuk a jelszót.

A bejelentkezéshez ekkor le kell valahogy generálnunk az egyszeri jelszavunkat. Ezt egy megbízható rendszeresen tudjuk megtenni az `opiekey` lefuttatásával. (Ennek vannak DOS-os, Windows-os és Mac OS-es változatai is.) Paraméterként az iterációs számot és a magot kell megadnunk. Ezt akár közvetlenül át is másolhatjuk annak a gépnek a bejelentkezési képernyőjéről, ahova be akarunk jelentkezni.

A megbízható rendszeren tehát:

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

Most már megvan a bejelentkezéshez szükséges egyszeri jelszavunk.

14.5.4. Több egyszeri jelszó létrehozása

Néha olyan helyekre kell mennünk, ahol se egy megbízható gép, sem pedig biztonságos kapcsolat nem található. Ilyen esetekben megadhatjuk az `opiekey` parancsnak, hogy előre gyártson le több egyszer használatos jelszót, amit később aztán ki tudunk nyomtatni. Például:

```
% opiekey -n 5 30 zz99999
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase: <secret password>
26: JOAN BORE FOSS DES NAY QUIT
27: LATE BIAS SLAY FOLK MUCH TRIG
28: SALT TIN ANTI LOON NEAL USE
29: RIO ODIN GO BYE FURY TIC
30: GREW JIVE SAN GIRD BOIL PHI
```

Az `-n 5` öt kulcsot kér egymás után, a `30` pedig megadja az utolsó iterációs számot. Vegyük észre, hogy a kulcsokat a felhasználás sorrendjével *ellentétes* sorrendben írja ki a program. Ha igazán paranoiások vagyunk, akkor írjuk le kézzel a jelszavakat. Ha viszont annyira nem, akkor egyszerűen küldjük át ezeket az `lpr` parancsnak.

Megfigyelhetjük, hogy minden sorban látható az iterációs szám és a hozzá tartozó egyszeri jelszó. Hasznos lehet a felhasználás szerinti felírni a jelszavakat.

14.5.5. A UNIX® jelszavak használatának leszűkítése

Az OPIE képes a bejelentkezéshez használt IP-címek alapján leszűkíteni a UNIX jelszavak használatát. Ehhez az `/etc/opieaccess` használható, amely alpból megtalálható a rendszerünkön. Az `opieaccess(5)` man oldalán találhatjuk meg a rá vonatkozó információkat és az összes vele kapcsolatos biztonsági megfontolást.

Íme egy példa az `opieaccess` állományra:

```
permit 192.168.0.0 255.255.0.0
```

Ezzel a sorral megengedjük a UNIX jelszavak használatát minden olyan felhasználó számára, akinek az IP-je illeszkedik a megadott címre és maszkra (ez viszont álcázással kijátszható).

Ha az `opieaccess` állományból egyetlen szabály sem illeszkedik, akkor alapértelmezés szerint nem engedélyezettek a nem OPIE típusú jelszavak.

14.6. A TCP kapcsolatok burkolása

Írta: Tom Rhodes.

Aki ismeri az `inetd(8)` programot, az már biztosan hallott a TCP kapcsolatok burkolásáról, eredeti nevén a a TCP wrapperekről. Azonban csak kevesek képesek felfogni ezek valódi hasznát. Úgy néz ki, mindenki csak tűzfalakon keresztül akarja megoldani a hálózati kapcsolatot kezelését. Habár a tűzfalakat sok mindenre fel lehet ugyan használni, egyetlen tűzfal nem képes például szövegesen válaszolni a kapcsolatok kezdeményezőinek. Ellenben bármelyik TCP-wrapper szoftver képes erre, sőt még többre is. A következő néhány szakaszban szemügyre vesszük a TCP wrapperek számos lehetőségét, és ahol lehetséges, ott konfigurációs állományokkal is illusztráljuk ezek használatát.

A TCP burkoló szoftverek kiterjesztik az **inetd** képességeit minden alatta dolgozó szerverdémon támogatására. Ezzel a módszerrel meg lehet oldani a naplózást, üzenetek küldését a kapcsolatokhoz, a démonok elérhetőségének korlátozását stb. Noha ezen lehetőségek közül néhány tűzfallal is megvalósítható, ezzel nem csupán egy további védelmi réteget húzunk fel a rendszerünk köré, hanem túllépjük mindazt, amit egy tűzfallal irányítani lehet.

A TCP burkolók használatával hozzáadott funkcionalitás azonban nem helyettesít egy jó tűzfalat. A TCP kapcsolatok burkolását tűzfallal vagy más egyéb biztonsági megoldással együtt tudjuk csak eredményesen használni, viszont a rendszerünk biztonságában egy újabb remek védelmi vonalat képvisel.

Mivel lényegében ez az **inetd** beállításának kibővítése, ezért a szakasz elolvasásához feltételezzük az `inetd` beállításával kapcsolatos tudnivalók ismeretét.

Megjegyzés: Bár az `inetd(8)` által indított programok nem egészen tekinthetők "démonoknak", hagyományosan démonnak hívják ezeket. Ezért rájuk ebben a szakaszban is ezt a kifejezést használjuk.

14.6.1. Kezdeti beállítások

FreeBSD alatt a TCP burkolók használatának egyetlen feltétele csupán annyi, hogy az **inetd** parancsot a `-w` paraméterrel indítsuk az `rc.conf` állományból. Az egyébként az alapbeállítás. Természetesen nem árt, ha helyesen állítjuk be az `/etc/hosts.allow` állományt is, ellenkező esetben a `syslogd(8)` egyébként dobálni fogja erről az üzeneteket.

Megjegyzés: Eltérően a TCP burkolók egyéb implementációitól, a `hosts.deny` állományt itt már nem használjuk. Minden beállítást az `/etc/hosts.allow` állományba kell raknunk.

A legegyszerűbb konfiguráció esetén a démonok kapcsolódását egyszerűen engedélyezhetjük vagy letilthatjuk az `/etc/hosts.allow` állományban szereplő beállításokkal. A FreeBSD alapértelmezett beállításai szerint minden **inetd** által indított démonhoz lehet kapcsolódni. Ennek megváltoztatásával az alapkonfiguráció áttekintése után foglalkozunk.

Az alapkonfiguráció általában `démon : cím : cselekvés` alakú. Itt a démon egy olyan démonra utal, amelyet az **inetd** indított el. A cím egy érvényes hálózati név, IP-cím vagy szögletes zárójelek (`[]`) között megadott IPv6 formátumú cím. A cselekvést tartalmazó mező (action) lehet `allow` vagy `deny` annak megfelelően, hogy engedélyezzük vagy tiltjuk a megadott címről a csatlakozást. Nem szabad elfelejtenünk, hogy az így megadott beállítások közül mindig az elsőként illeszkedő érvényesül, ami arra utal, hogy a konfigurációs állományban szereplő szabályok egymás után növekvő sorrendben értékelődnek ki. Ha valamelyikük illeszkedik, akkor a keresés megáll.

Rengeteg egyéb opció is megadható még, de ezekről csak a későbbi szakaszokban fogunk szólni. Egy egyszerű konfigurációs állomány már ennyi információból is könnyedén összeállítható. Például, ha engedélyezni szeretnénk a POP3 kapcsolatokat a `mail/qpopper` démonon keresztül, akkor a következő sorral kell kiegészítenünk a `hosts.allow` állományt:

```
# Ez a sor kell a POP3 kapcsolatokhoz:
qpopper : ALL : allow
```

Miután hozzáadtuk ezt a sort, az **inetd** szervert újra kell indítanunk. Ezt vagy a `kill(1)` parancssal, vagy pedig az `/etc/rc.d/inetd` szkript `restart` paraméterével tehetjük meg.

14.6.2. Komolyabb beállítások

A TCP kapcsolatok burkolásánál is meg lehet adni további opciókat. Segítségükkel még jobban irányítani tudjuk a kapcsolatok kezelésének módját. Néhány esetben az is hasznos lehet, ha küldünk valamilyen választ az egyes gépeknek vagy démonoknak. Máskor szükségünk lehet a csatlakozások naplózására vagy e-mailen keresztüli jelzésére a rendszergazda felé. Teljesen más helyzetekben csak a helyi hálózatunkról engedjük meg a csatlakozást. Ez mind lehetséges a helyettesítő jelekként ismert beállítási opciók, kiterjesztő karakterek és külső parancsok végrehajtásának használatával. A következő két szakasz az ilyen és ehhez hasonló szituációk megoldására íródott.

14.6.2.1. Külső parancsok

Tegyük fel, hogy olyan helyzetben vagyunk, amikor a kapcsolatot tiltani akarjuk, de közben azért szeretnénk erről értesíteni a kapcsolatot kezdeményező felet is. Hogyan tudjuk ezt megcsinálni? Ezt a `twist` nevű opcióval tehetjük meg. Amikor megpróbál valaki csatlakozni, akkor a `twist` hívódik meg és végrehajt egy megadott parancsot vagy szkriptet. Erre találunk is egy példát a `hosts.allow` állományban:

```
# The rest of the daemons are protected.
ALL : ALL \
    : severity auth.info \
    : twist /bin/echo "You are not welcome to use %d from %h."
```

Ez a példa a következő üzenetet jeleníti meg: "You are not allowed to use a démon neve from hálózati név." (Jelentése: "A démon neve démon nem érheti el a hálózati név helyről!") Ez minden olyan démon esetén megjelenik, amiről nem nyilatkoztunk korábban az állományban. Ezzel nagyon könnyen vissza tudunk küldeni egy választ a kapcsolat kezdeményezője felé, miután a kapcsolatot eldobtuk. Vegyük észre, hogy a visszaküldendő üzenetet " karakterek közé *kell* tennünk, ez alól semmi sem kivétel.

Figyelem DoS támadást lehet előidézni azzal, ha egy támadó vagy támadók egy csoportja csatlakozási kérésekkel kezdi el bombázni a démonainkat.

Ilyen esetekben használhatjuk a `spawn` opciót is. A `spawn` a `twist` opcióhoz hasonlóan implicit módon tiltja a kapcsolódást és arra használható, hogy lefuttassunk vele egy parancsot vagy szkriptet. A `spawn` azonban a `twist` opciótól eltérően nem küld vissza semmilyen választ a kapcsolatot létrehozni kívánó egyénnek. Ehhez példaként vegyük a következő sort a konfigurációs állományban:

```
# We do not allow connections from example.com:
ALL : .example.com \
    : spawn (/bin/echo %a from %h attempted to access %d >> \
    /var/log/connections.log) \
    : deny
```

Ezzel a `*.example.com` címtartományból érkező összes kapcsolódási kísérlet sikertelen lesz, miközben ezzel egyidőben a `/var/log/connections.log` állományba rögzítjük a csatlakozni akaró egyén hálózati nevét, IP-címét és a démon.

A korábban már kifejtett helyettesítő karakterek túl, mint például az `%a`, még léteznek továbbiak is. Róluk a `hosts_access(5)` man oldalon találhatjuk meg a teljes listát.

14.6.2.2. Helyettesítő jelek

Az eddigi példákban folyamatosan csak az `ALL` opciót adtuk meg. Azonban rajta kívül léteznek mások is, amivel a megoldás funkcionalitását még egy kicsivel tovább növelhetjük. Például az `ALL` használható egy démon, egy tartomány vagy egy IP-cím illesztésére. A másik ilyen helyettesítő jel a `PARANOID`, amelyet olyan gépek IP-címének illesztésekor alkalmazhatunk, ami feltételezhetően hamis. Más szóval a `PARANOID` olyan cselekvések megadását teszi lehetővé, amelyek akkor hajtódnak végre, amikor a kapcsolatot létrehozó gép IP-címe eltér a hálózati nevétől. A most következő példa valószínűleg segít fényt deríteni ennek lényegére:

```
# Block possibly spoofed requests to sendmail:
sendmail : PARANOID : deny
```

A példában minden olyan kapcsolatkérést elutasítunk, ami a `sendmail` felé a hálózati névtől eltérő IP-címről irányul.

Figyelem Ha rossz DNS beállításokat használunk, a `PARANOID` megadásával súlyosan mozgásképtelenné tehetjük a kliensünket vagy szerverünket. Ezért legyünk óvatosak vele!

A helyettesítő jelekről és hozzájuk tartozó további lehetőségekről a `hosts_access(5)` man oldalon tájékozódhatunk.

A `hosts.allow` állományból ki kell venni az első sort ahhoz, hogy bármilyen egyéb konfigurációs beállítás működőképes legyen. Ezt említettük a szakasz elején is.

14.7. KerberosIV

Írta: Mark Murray. Eredetileg írta: Mark Dapoz.

A Kerberos egy olyan járulékos rendszer/protokoll, amellyel a felhasználók egy biztonságos szerver szolgáltatásain keresztül tudják hitelesíteni magukat. Ilyen szolgáltatás többek közt a távoli bejelentkezés, távoli másolás, a rendszeren belüli biztonságos másolás és minden olyan egyéb veszélyes feladat, amit számottevően megbízhatóbbá és irányíthatóbbá tettek.

A következő utasítások a FreeBSD-hez mellékelt Kerberos beállításához adnak útmutatást. A teljes leíráshoz azonban érdemes fellapoznunk a menet közben hivatkozott man oldalakat is.

14.7.1. A KerberosIV telepítése

A Kerberos a FreeBSD egyik választható komponense. Legkönnyebben úgy tudjuk feltelepíteni, ha a FreeBSD telepítése során a **sysinstall** programban kiválasztjuk a `krb4` vagy `krb5` terjesztések valamelyikét. Ezzel felrakhatjuk a Kerberos “eBones” (KerberosIV) vagy “Heimdal” (Kerberos5) elnevezésű változatait. A FreeBSD azért tartalmazza ezeket az implementációkat, mert nem az Amerikai Egyesült Államokban vagy Kanadában fejlesztették, így az Egyesült Államok titkosításokkal kapcsolatos kiviteli korlátozások korában minden olyan rendszer adminisztrátora el tudta érni, aki nem ezekben az országokban lakott.

A Kerberos MIT által fejlesztett implementációját egyébként a Portgyűjteményből a `security/krb5` porton keresztül érhetjük el.

14.7.2. A kezdeti adatbázis létrehozása

Ezt a lépést csak a Kerberos szerveren kell elvégezni. Először is győződjünk meg róla, hogy semmilyen korábbi Kerberos adatbázis nem található a gépen. Váltunk az `/etc/kerberosIV` könyvtárra és ellenőrizzük a következő állományok meglétét:

```
# cd /etc/kerberosIV
# ls
README          krb.conf        krb.realms
```

Ha rajtuk kívül további állományok is feltűnnének (mint például a `principal.*` vagy `master_key`), akkor a `kdb_destroy` paranccsal pusztítsuk el a régi Kerberos adatbázist, vagy ha nem fut már a Kerberos, akkor egyszerűen csak töröljük le ezeket.

Ezután lássunk neki a `krb.conf` és `krb.realms` állományok átírásán keresztül a Kerberos egyes övezeteinek (realm) létrehozásához. Itt most az `EXAMPLE.COM` lesz a létrehozandó övezet, a hozzá tartozó szerver pedig a `grunt.example.com`. Így szerkesszük át vagy készítsünk el a neki megfelelő `krb.conf` állományt:

```
# cat krb.conf
EXAMPLE.COM
EXAMPLE.COM grunt.example.com admin server
CS.BERKELEY.EDU okeeffe.berkeley.edu
ATHENA.MIT.EDU kerberos.mit.edu
ATHENA.MIT.EDU kerberos-1.mit.edu
ATHENA.MIT.EDU kerberos-2.mit.edu
ATHENA.MIT.EDU kerberos-3.mit.edu
LCS.MIT.EDU kerberos.lcs.mit.edu
TELECOM.MIT.EDU bitsy.mit.edu
ARC.NASA.GOV trident.arc.nasa.gov
```

A többi övezetnek valójában nem feltétlenül kell itt lennie. Ezek csupán azért szerepelnek itt, hogy bemutassák miként lehet egyetlen géphez hozzárendelni egyszerre több övezetet is. Az egyszerűség kedvéért nyugodtan elhagyhatóak.

Az első sor nevezi meg a rendszer által működtetett övezeteket. Az utána következő sorokban övezeteket és hálózati neveket láthatunk. Itt az első elem egy övezetet nevez meg, a második elem pedig az övezet “kulcselosztó központját” (key distribution center). A hálózati nevet követő `admin server` kulcsszavak arra utalnak, hogy az adott gép adminisztratív szerepet ellátó adatbázist is tartalmaz. Ezeket a fogalmakat részleteiben a Kerberos man oldalain ismerhetjük meg.

Ezután hozzá kell adnunk a `grunt.example.com` nevű gépet az `EXAMPLE.COM` övezethez, valamint az `.example.com` tartományban levő összes géphez létre kell hoznunk egy bejegyzést az `EXAMPLE.COM` övezetben. A `krb.realms` állományt ehhez a következőképpen kellene módosítanunk:

```
# cat krb.realms
grunt.example.com EXAMPLE.COM
.example.com EXAMPLE.COM
.berkeley.edu CS.BERKELEY.EDU
.mit.edu ATHENA.MIT.EDU
.mit.edu ATHENA.MIT.EDU
```

Ismét hozzátesszük, hogy a többi övezetnek nem kötelező itt szerepelnie. Ezek csupán azt demonstrálják, hogy miként kell egy gépet egyszerre több övezethez is beállítani. Az átláthatóság kedvéért minden további nélkül eltávolíthatjuk ezeket.

Itt az első sor az *adott* rendszert elhelyezi egy nevesített övezetbe. A többi sor azt mutatja meg, hogyan kell alapértelmezett módon a meghatározott altartományokba tartozó gépeket egy nevesített övezethez hozzárendelni.

Most már készen állunk az adatbázis létrehozására. Ehhez egyedül a Kerberos szerverét (avagy Kulcselosztó központját) kell elindítanunk. Adjuk ki a `kdb_init` parancsot:

```
# kdb_init
Realm name [default ATHENA.MIT.EDU ]: EXAMPLE.COM
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
```

```
Enter Kerberos master key:
```

Az üzenet fordítása:

Most az adatbázis mesterkulcsát kell megadni. Fontos, hogy NE FELEJTSÜK EL ezt a jelszót.

Most el kell mentenünk a kulcsot, így a helyi gépen futó szerverek fel tudják szedni. Ehhez a `kstash` parancsra van szükségünk:

```
# kstash
```

```
Enter Kerberos master key:
```

```
Current Kerberos master key version is 1.
```

```
Master key entered. BEWARE!
```

Az üzenet fordítása:

```
A Kerberos mesterkulcsának jelenlegi változata: 1.
```

```
VIGYÁZAT, megadták a mesterkulcsot!
```

Ez elmenti a titkosított mesterkulcsot az `/etc/kerberosIV/master_key` állományba.

14.7.3. Az egész beüzemelése

Mindegyik Kerberossal őrzött rendszerrel kapcsolatban két ún. szereplőt (principal) kell még hozzátennünk az adatbázishoz. A nevük `kpasswd` és `rcmd`. Minden rendszerhez létre kell hoznunk ezeket a szereplőket, példányonként (instance) az egyes rendszerek neveivel.

A `kpasswd` és `rcmd` démonok teszik lehetővé a többi rendszer számára, hogy megváltoztathassák a Kerberos jelszavukat, valamint hogy futtathassák az `rcp(1)`, `rlogin(1)` és `rsh(1)` parancsokat.

Vegyük fel ezeket a bejegyzéseket is:

```
# kdb_edit
```

```
Opening database...
```

```
Enter Kerberos master key:
```

```
Current Kerberos master key version is 1.
```

```
Master key entered. BEWARE!
```

```
Previous or default values are in [brackets] ,  
enter return to leave the same, or new value.
```

```
Principal name: passwd
```

```
Instance: grunt
```

```
<Not found>, Create [y] ? y
```

```
Principal: passwd, Instance: grunt, kdc_key_ver: 1
```

```
New Password: <---- írjuk be, hogy "RANDOM"
```

```
Verifying password
```

```
New Password: <---- írjuk be, hogy "RANDOM"
```

```
Random password [y] ? y
```

```

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name: rcmd
Instance: grunt

<Not found>, Create [y] ?

Principal: rcmd, Instance: grunt, kdc_key_ver: 1
New Password:          <---- írjuk be, hogy "RANDOM"
Verifying password

New Password:          <---- írjuk be, hogy "RANDOM"

Random password [y] ?

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:          <---- ha nem adunk meg semmit, akkor kilép

```

14.7.4. A szerver állomány létrehozása

Most pedig kivonatolni kell azokat a példányokat, amelyek szolgáltatást definiálnak a gépen. Erre az `ext_srvtab` parancsot használjuk. Ennek eredményeképpen keletkezik egy állományt, amelyet *biztonságos eszközökkel* át kell másolni vagy át kell mozgatni az egyes Kerberos kliensek `/etc` könyvtárába. Ennek az állománynak egyaránt jelent kell lennie a szerveren és a kliensen is, nélküle a Kerberos működésképtelen.

```

# ext_srvtab grunt
Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!
Generating 'grunt-new-srvtab'....

```

Ez a parancs most létrehozott egy ideiglenes állományt, amit át kell nevezni az `srvtab` névre, hogy megtalálhassák a szerverek. Az eredeti rendszeren a `mv(1)` paranccsal tudjuk a helyére rakni:

```
# mv grunt-new-srvtab srvtab
```

Ha egy kliensnek szánjuk az állományt és a hálózatunkat nem tekinthetjük biztonságosnak, akkor a `kliens-new-srvtab` állományt másoljuk egy mozgatható adathordozóra és megbízható módon jutassuk el. Ne felejtjük el az állományt `srvtab` néven átrakni a kliens `/etc` könyvtárába és az engedélyeit 600-ra állítani:

```
# mv grumble-new-srvtab srvtab
```

```
# chmod 600 srvtab
```

14.7.5. Az adatbázis feltöltése

Ezt követően rögzítenünk kell néhány felhasználót is adatbázisban. Először is hozzunk létre egy bejegyzést a janos nevű felhasználónak. Ezt a kdb_edit parancs kiadásával tesszük meg:

```
# kdb_edit
Opening database...

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered.  BEWARE!
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name:  janos
Instance:

<Not found>, Create [y] ?  y

Principal: janos, Instance: , kdc_key_ver: 1
New Password:          <---- adjunk meg egy biztonságos jelszót
Verifying password

New Password:          <---- itt ismét adjuk meg a jelszót
Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:          <---- ha nem írunk be semmit, akkor kilép
```

14.7.6. Próbáljuk ki

Elsőként a Kerberos démonait kell beindítanunk. Ezzel kapcsolatban megjegyeznénk, hogy ha ehhez megfelelően átírtuk az /etc/rc.conf állományunkat, akkor ez az újraindítással együtt magától lezajlik. Ezt csak a Kerberos szerveren kell megcsinálni. A Kerberos kliensei maguktól összeszedik a működésükhöz szükséges adatokat az /etc/kerberosIV könyvtárból.

```
# kerberos &
Kerberos server starting
Sleep forever on error
Log file is /var/log/kerberos.log
Current Kerberos master key version is 1.

Master key entered. BEWARE!
```

```
Current Kerberos master key version is 1
Local realm: EXAMPLE.COM
# kadmin -n &
KADM Server KADM0.0A initializing
Please do not use 'kill -9' to kill this job, use a
regular kill instead
```

```
Current Kerberos master key version is 1.
```

```
Master key entered. BEWARE!
```

A fenti figyelmeztetés fordítása:

A program leállítására ne a 'kill -9' parancsot, hanem a normális kill parancsot használjuk

Ezután a kinit parancs használatával próbáljunk meg az előbb létrehozott janos azonosítónak kérni egy jegyet:

```
% kinit janos
MIT Project Athena (grunt.example.com)
Kerberos Initialization for "janos"
Password:
```

A klist paranccsal most próbáljuk meg kilistázni a tokeneket és így ellenőrizni, hogy valóban rendelkezünk velük:

```
% klist
Ticket file:      /tmp/tkt245
Principal:        janos@EXAMPLE.COM

    Issued                Expires                Principal
Apr 30 11:23:22  Apr 30 19:23:22  krbtgt.EXAMPLE.COM@EXAMPLE.COM
```

Ezután a passwd(1) használatával próbáljuk meg megváltoztatni a jelszavunkat. Ezzel tudjuk ellenőrizni, hogy a **kpasswd** démon hozzáfér a Kerberos adatbázisához:

```
% passwd
realm EXAMPLE.COM
Old password for janos:
New Password for janos:
Verifying password
New Password for janos:
Password changed.
```

14.7.7. Adminisztrátori jogosultságok felvétele

A Kerberos lehetővé teszi, hogy *mindegyik* olyan felhasználónak, akinek rendszergazdai jogokra lenne szüksége, a su(1) eléréséhez *külön* meg tudjunk adni egy jelszót. Most már tudunk mondani egy olyan azonosítót is, amely jogosult a su(1) használatával root jogokat szerezni. Ezt úgy tudjuk megoldani, ha az adott szereplőhöz társítunk egy root példányt. A kdb_edit használatával készíteni tudunk egy janos.root bejegyzést a Kerberos adatbázisában:

```
# kdb_edit
```

Opening database...

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!

Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: **janos**

Instance: **root**

<Not found>, Create [y] ? y

Principal: janos, Instance: root, kdc_key_ver: 1

New Password: <---- ide csak egy BIZTONSÁGOS jelszót adjuk meg!

Verifying password

New Password: <---- adjuk meg ismét a jelszót

Principal's new key version = 1

Expiration date (enter yyyy-mm-dd) [2000-01-01] ?

Max ticket lifetime (*5 minutes) [255] ? **12** <--- ne állítsuk nagyon hosszúra!

Attributes [0] ?

Edit O.K.

Principal name: <---- ha nem adunk meg semmit, akkor kilép

Ezt követően úgy tudunk megbizonyosodni a működéséről, hogy megpróbálunk neki tokeneket szerezni:

```
# kinit janos.root
```

```
MIT Project Athena (grunt.example.com)
```

```
Kerberos Initialization for "janos.root"
```

```
Password:
```

Most rakjuk bele a felhasználót a root .klogin állományába:

```
# cat /root/.klogin
```

```
janos.root@EXAMPLE.COM
```

Ezután próbáljunk meg kiadni a su(1) parancsát:

```
% su
```

```
Password:
```

Nézzük meg milyen tokenjeink is vannak:

```
# klist
```

```
Ticket file: /tmp/tkt_root_245
```

```
Principal: janos.root@EXAMPLE.COM
```

Issued	Expires	Principal
May 2 20:43:12	May 3 04:43:12	krbtgt.EXAMPLE.COM@EXAMPLE.COM

14.7.8. Más parancsok használata

Az iménti példában létrehoztunk egy *janos* nevű szereplőt, amihez a *root* egy példányát rendeltük. Ez egy olyan felhasználón alapján történt, akinek a neve megegyezik a hozzá tartozó szereplővel, ami a Kerberosban alapértelmezés. Amennyiben a szükséges megjegyzések megtalálhatóak a *root* könyvtárában levő *.klogin* állományban, akkor a *felhasználó.root* formátumú *szereplő.példány* azonosító megengedi a *felhasználó* számára, hogy végrehajtsa a *su(1)* parancsot.

```
# cat /root/.klogin
janos.root@EXAMPLE.COM
```

Ehhez hasonlóan, ha a felhasználó saját könyvtárában megtalálható egy ilyen állomány:

```
% cat ~/.klogin
janos@EXAMPLE.COM
jozsef@EXAMPLE.COM
```

Ezzel a konfigurációval bárki, aki *janos* felhasználóként vagy *jozsef* felhasználóként (a *kinit* parancson keresztül) hitelesítette magát *EXAMPLE.COM* övezetből, ezen a rendszeren (*grunt*) bejelentkezhet a *janos* nevű felhasználóként vagy hozzáférhet az állományaihoz az *rlogin(1)*, *rsh(1)* vagy *rcp(1)* használatával.

Például *janos* most egy másik Kerberost használó rendszerre jelentkezik be:

```
% kinit
MIT Project Athena (grunt.example.com)
Password:
% rlogin grunt
Last login: Mon May  1 21:14:47 from grumble
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California.  All rights reserved.

FreeBSD BUILT-19950429 (GR386) #0: Sat Apr 29 17:50:09 SAT 1995
```

Vagy *jozsef* jelentkezik be ugyanazon a gépen *janos* hozzáféréssel (a *janos* nevű felhasználónak a fentebb bemutatott *.klogin* állomány található a könyvtárában és a Kerberos üzemeltetéséért felelős személy létrehozott egy *jozsef* nevű szereplőt egy null példánnyal):

```
% kinit
% rlogin grunt -l janos
MIT Project Athena (grunt.example.com)
Password:
Last login: Mon May  1 21:16:55 from grumble
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California.  All rights reserved.
FreeBSD BUILT-19950429 (GR386) #0: Sat Apr 29 17:50:09 SAT 1995
```

14.8. Kerberos5

Írta: Tillman Hodgson. Eredetileg írta: Mark Murray.

A FreeBSD 5.1 után következő mindegyik FreeBSD kiadás már csak a **Kerberos5** támogatást tartalmaz. Ezért bennük csak a **Kerberos5** található meg, és a beállítása sok szempontból hasonlít a **KerberosIV** beállításához. A most következő információk csak és kizárólag a FreeBSD 5.0 kiadás után következőkben található **Kerberos5** változatra vonatkoznak. A **KerberosIV** szolgáltatásait a felhasználók csomagként, a `security/krb4` porton keresztül érhetik el.

A **Kerberos** egy hálózati kiegészítő rendszer/protokoll, amivel a felhasználók egy biztonságos szerveren keresztül képesek magukat azonosítani. A távoli bejelentkezések, távoli másolások, a rendszer belüli védett másolások valamint egyéb nagyon kockázatos feladatok, szolgáltatások biztonsága és felügyelete így jelentős mértékben javítható.

A **Kerberos** úgy írható le, mint az személyazonosságok ellenőrzésére feljogosított rendszer. Vagy tekinthetjük egy megbízható külső megfigyelő által végzett hitelesítési rendszernek is. A **Kerberos** csak egyetlen funkciót kínál fel — ez a felhasználók biztonságos hitelesítése a hálózaton. Viszont nem nyújt semmilyen felhatalmazási (mit csinálhatnak a felhasználók) vagy vizsgálati (mit csináltak végül a felhasználók) lehetőséget. Miután egy kliens és a szerver a **Kerberos** használatával azonosították egymást, az egymás közt folyó kommunikációjuk titkosításával képesek megőrizni az átáramló adatok sértetlenségét és lehallgathatatlanságát.

Ennek tükrében a **Kerberos** használata csak más olyan biztonsági módszerekkel együttesen javasolt, amelyek felhatalmazást és vizsgálati szolgáltatásokkal is rendelkeznek.

A most következő utasítások arra igyekeznek útmutatást adni, hogy miként használjuk a FreeBSD-vel együtt terjesztett **Kerberos** verziót. Azonban a teljes leírást csak a témához tartozó man oldalak átolvasásával együtt kapjuk meg.

A **Kerberos** telepítésének bemutatásához az alábbi névtereket fogjuk használni:

- A DNS tartomány (“zóna”) az `example.org` lesz.
- A **Kerberos** övezet az `EXAMPLE.ORG` lesz.

Megjegyzés: Kérjük, hogy még abban az esetben is valódi tartományneveket adjuk meg, amikor a **Kerberos** használatát csak a belső hálózaton tervezzük. Ezzel elkerülhetjük az egyes **Kerberos** övezetek együttműködése során felmerülő DNS problémákat.

14.8.1. A Kerberos története

A **Kerberost** az MIT hozta létre a hálózati biztonsággal kapcsolatos problémák egyik megoldásaként. A **Kerberos** erős titkosítást használ, ezért a kliensek képesek egy nem biztonságos hálózaton is azonosítani magukat a szerver felé (és fordítva).

A **Kerberos** egyaránt utal egy hálózati protokoll névére és azokra programokra, amelyek implementálják (például **Kerberos** telnet). Az 5 a protokoll jelenlegi verziója, amit az RFC 1510 ír le.

A protokollnak számos szabad változata létezik, rengeteg típusú operációs rendszerre. A Massachusettsi Műszaki Intézet (Massachusetts Institute of Technology, MIT), ahol a **Kerberost** eredetileg kifejlesztették, napjainkban is folytatja a saját **Kerberos** csomagjának fejlesztését. Többnyire az Egyesült Államokban használják titkosításra, mivel régebben az amerikai kiviteli korlátozások voltak rá érvényesek. Az MIT **Kerberos** változata portként érhető

el (security/krb5). A Heimdal **Kerberos** egy másik 5 verziójú implementáció, amit a kiviteli korlátozások elkerülése érdekében határozottan az Egyesült Államokon kívül fejlesztettek ki (ezért gyakran megtalálhatjuk a különböző nem kereskedelmi UNIX variánsokban). A Heimdal **Kerberos** terjesztés portként elérhető (security/heimdal) és kisebb mérethben a FreeBSD alaptelepítésének is része.

Mivel ezzel az írással a legtöbb felhasználót kívánjuk segíteni, ezért a következő utasítások a FreeBSD telepítésében mellékelt Heimdal terjesztés használatát feltételezik.

14.8.2. A Heimdal kulcselosztójának telepítése

A kulcselosztó központ (Key Distribution Center, avagy KDC) az a centralizált hitelesítési szolgáltatás, amit a **Kerberos** nyújt — lényegében az a számítógép, amely **Kerberos**-jegyeket bocsájt ki. A KDC “megbízhatónak” tekinthető a **Kerberos** által kialakított övezetben levő többi számítógép számára, ezért védelme kiemelten fontos.

Itt jegyeznénk meg, hogy habár a **Kerberos** szerver futtatása nagyon kevés számítógépes erőforrást igényel, ennek ellenére biztonsági szempontból egy külön számítógépet javasoljunk a kulcselosztó szerepének betöltéséhez.

Mielőtt nekifognánk a KDC konfigurálásának, ellenőrizzük, hogy az `/etc/rc.conf` tartalmazza a KDC működéséhez szükséges beállításokat (az elérési utakat természetesen a saját rendszerünk szerint állítsuk be):

```
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

A következő lépésben vegyük szemügyre a **Kerberos** beállításait tartalmazó `/etc/krb5.conf` állományt:

```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
        kdc = kerberos.example.org
        admin_server = kerberos.example.org
    }
[domain_realm]
    .example.org = EXAMPLE.ORG
```

Vegyük észre, hogy az itt szereplő `/etc/krb5.conf` állomány szerint a kulcselosztónk teljes hálózati neve `kerberos.example.org`. Ha a kulcselosztónknak nem ez a neve, akkor a zónákat leíró állományba vegyünk még fel egy ilyen CNAME (álnév) bejegyzést.

Megjegyzés: Ha egy nagyobb hálózatban vagyunk, ahol a DNS szervert is megfelelően beállították, akkor az iménti példa ennyire leszűkíthető:

```
[libdefaults]
    default_realm = EXAMPLE.ORG
```

Itt már a következő sorokat hozzáadták `example.org` zónát leíró állományhoz:

```
_kerberos._udp      IN  SRV      01 00 88 kerberos.example.org.
_kerberos._tcp      IN  SRV      01 00 88 kerberos.example.org.
_kpasswd._udp       IN  SRV      01 00 464 kerberos.example.org.
_kerberos-adm._tcp  IN  SRV      01 00 749 kerberos.example.org.
_kerberos            IN  TXT       EXAMPLE.ORG
```

Megjegyzés: A kliensek csak akkor lesznek képesek elérni a **Kerberos** szolgáltatásait, ha vagy *kötelező jelleggel* megadunk egy teljesen beállított `/etc/krb5.conf` állományt, vagy egy minimális `/etc/krb5.conf` állományt és egy helyesen beállított DNS szerveret használunk.

Ezután létrehozzuk a **Kerberos** adatbázisát. Ez az adatbázis tartalmazza az összes szereplő kulcsát a mesterkulccsal titkosítva. Erre a jelszóra nem kell feltétlenül emlékeznünk, mivel ez egy állományban tárolódik (`/var/heimdal/m-key`). A mesterkulcsot a `kstash` parancs kiadásával és egy jelszó megadásával tudjuk létrehozni.

Ahogy a mesterkulcs elkészült, a `kadmin` parancs `-l` (mint “lokális”, azaz helyi) opciójával inicializálni tudjuk az adatbázist. Ez az opció arra utasítja a `kadmin` programot, hogy ne a `kadmind` hálózati szolgáltatást használja, hanem közvetlenül az adatbázis állományait módosítsa. Ezzel oldható meg az adatbázis kezdeti létrehozásának problémája. Miután megkaptuk a `kadmin` parancssorát, az övezetünkhöz tartozó adatbázis inicializálásához adjuk ki az `init` parancsot.

Végül, még mindig a `kadmin` parancssorát használva, az `add` paranccsal hozzuk létre az első szereplőnket. Egyelőre érjük be az alapértelmezett értékekkel, a `modify` paranccsal később úgyis meg tudjuk változtatni ezeket. Hozzáteesszük, hogy itt a `?` parancs segítségével bármikor lekérhetjük az opciók ismertetését.

Példa egy adatbázis létrehozására:

```
# kstash
Master key: xxxxxxxx
Verifying password - Master key: xxxxxxxx
```

```
# kadmin -l
kadmin> init EXAMPLE.ORG
Realm max ticket life [unlimited]:
kadmin> add tillman
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
Password: xxxxxxxx
Verifying password - Password: xxxxxxxx
```

Most már ideje elindítani a KDC szolgáltatásait. Ezeket az `/etc/rc.d/kerberos start` és `/etc/rc.d/kadmind start` parancsok kiadásával tudjuk felhozni. Megjegyezzük, hogy most még semmilyen kerberizált démont nem kell elindítanunk. Ellenben igyekezzünk ellenőrizni a KDC működőképességét azzal, hogy KDC parancssorából kérünk egy jegyet a frissen hozzáadott szereplőnknek (felhasználónknak) és kilistázzuk:

```
% kinit tillman
tillman@EXAMPLE.ORG's Password:

% klist
Credentials cache: FILE:/tmp/krb5cc_500
Principal: tillman@EXAMPLE.ORG
```

```
Issued          Expires          Principal
Aug 27 15:37:58 Aug 28 01:37:58  krbtgt/EXAMPLE.ORG@EXAMPLE.ORG
```

Miután végeztünk, nyugodtan törölhetjük a jegyet:

```
% kdestroy
```

14.8.3. Szerverek kerberizálása a Heimdal használatával

Ehhez először is szükségünk lesz a **Kerberos** konfigurációs állományának, az `/etc/krb5.conf` másolatára. Ezt úgy tudjuk megtenni, ha egyszerűen átmásoljuk a kulcselosztóról az egyik kliensre valamilyen megbízható módon (vagy az `scp(1)` programhoz hasonló hálózati segédprogramok, vagy például fizikailag egy floppy lemez használatával).

Ezután szükségünk lesz egy `/etc/krb5.keytab` nevű állományra. Ez az alapvető különbség a kerberizált démonokat felkínáló szerver és egy munkaállomás közt — a szervernek rendelkeznie kell egy `keytab` állománnyal. Ez az állomány tartalmazza a szerver kulcsát, amivel így a kulcselosztóval kölcsönösen azonosítani tudják egymást. Ezt a szerverre biztonságosan kell eljuttatnunk, mivel ennek napvilágra kerülésével a szerver védelme komoly veszélybe kerül. Tehát, ha egy titkosítás nélküli csatornán, például FTP-n keresztül visszük át, akkor kifejezetten rossz ötlet.

A szerverre általában a `kadmin` program használatával érdemes átvinni a `keytab` állományt. Ez azért is hasznos, mert ehhez a `kadmin` segítségével létre kell hoznunk a befogadó szereplőt is (a kulcselosztó a `krb5.keytab` állomány végén).

Vegyük észre, hogy már kaptunk egy jegyet és ezzel a jeggyel jogosultaknak kell lennünk a `kadmind.acl` állomány `kadmin` felület használatára. A hozzáférést vezérlő listák (ACL-ek) tervezésével kapcsolatban olvassuk el Heimdal info oldalán található “Remote administration” című szakaszt (`info heimdal`). Amennyiben nem kívánjuk engedélyezni a `kadmin` távoli elérését, egyszerűen csak csatlakozunk valamilyen biztonságos módon (helyi konzolon, `ssh(1)` vagy egy kerberizált `telnet(1)` használatával) a kulcselosztóhoz, és a `kadmin -l` paranccsal végezzük el helyben az adminisztrációt.

Miután telepítettük az `/etc/krb5.conf` állományt, a **Kerberos** szerverről el tudjuk érni a `kadmin` felületét. Az `add --random-key` paranccsal most már hozzáadhatjuk a szerver befogadó szereplőjét és az `ext` paranccsal ki tudjuk vonni a szerver befogadó szereplőjét a saját `keytab` állományából. Például:

```
# kadmin
kadmin> add --random-key host/myserver.example.org
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
kadmin> ext host/myserver.example.org
kadmin> exit
```

Itt jegyeznénk meg, hogy az `ext` parancs (az “extract” rövidítése) a kivont kulcsot alapértelmezés szerint az `/etc/krb5.keytab` állományba menti ki.

Ha a kulcselosztón nem fut a `kadmind` szolgáltatás (valószínűleg biztonsági okokból) és ezért távolról nem tudjuk elérni a `kadmin` felületét, akkor így tudjuk közvetlenül hozzáadni a befogadó szereplőt (`host/myserver.EXAMPLE.ORG`), majd kivonatolni azt egy ideiglenes állományba (elkerülve az `/etc/krb5.keytab` felülírását):

```
# kadmin
kadmin> ext --keytab=/tmp/example.keytab host/myserver.example.org
kadmin> exit
```

Ezután valamilyen biztonságos eszközzel (például `scp` vagy floppy használatával) át tudjuk másolni `keytab` állományt a szerverre. A kulcselosztón levő `keytab` felülírását elkerülendő, ne feledkezzünk el egy megfelelő név megadásáról sem.

Ezen a ponton már a szerver képes felvenni a kapcsolatot a kulcselosztóval (a `krb5.conf` állomány miatt) és bizonyítani a személyazonosságát (a `krb5.keytab` állomány miatt). Így tehát készen állunk a szolgáltatások kerberizálására. Ebben a példában most a `telnet` szolgáltatást vesszük célba úgy, hogy először az `/etc/inetd.conf` állományba berakjuk az alábbi sort, majd újraindítjuk az `inetd(8)` szolgáltatást az `/etc/rc.d/inetd restart` paranccsal:

```
telnet      stream  tcp      nowait  root    /usr/libexec/telnetd  telnetd -a user
```

Itt az a legfontosabb, hogy az `-a` (mint authentication, azaz hitelesítés) paramétert a “user” beállítással adjuk meg. A `telnetd(8)` man oldalán olvashatunk ennek pontos részleteiről.

14.8.4. Kliensek kerberizálása a Heimdal használatával

A kliensek beállítása szinte majdnem gyerekjáték. A **Kerberos** beállításához egyedül az `/etc/krb5.conf` állományra lesz szükségünk. Valamilyen biztonságos eszközzel másoljuk át a kulcselosztóról a kliensre.

Úgy tudjuk letesztelni klienst, ha megpróbáljuk róla kiadni a `kinit`, `klist` és `kdestroy` parancsokat a fentebb létrehozott szereplő jegyének megszerzéséhez, lekérdezéséhez és megsemmisítéséhez. A **Kerberos** használatával megpróbálkozhatunk csatlakozni valamelyik kerberizált szerverre is, ha viszont ez nem működik még egy jegy megszerzése után sem, akkor a gond többnyire a szerverrel van, nem pedig a klienssel vagy a kulcselosztóval.

Amikor egy `telnet` vagy egy hozzá hasonló alkalmazást tesztelünk, egy csomaglehallgató (mint amilyen például a `tcpdump(1)`) elindításával győződjünk meg róla, hogy a jelszavak ilyenkor titkosítva mennek át. Próbáljuk meg titkosítani a teljes kommunikációt a `telnet -x` paraméterével (hasonlóan az `ssh` parancshoz).

Alapból még számos más kiegészítő **Kerberos** kliensalkalmazás is telepíthető. Ezeken érezhető meg valójában az alapszerverhez tartozó Heimdal változat “minimalitása”: ebben a `telnet` az egyedüli kerberizált szolgáltatás.

A Heimdal port igyekszik pótolni a hiányzó klienseket a kerberizált `ftp`, `rsh`, `rcp`, `rlogin` és néhány kevésbé ismert program telepítésével. Az MIT változat portja szintén tartalmazza a **Kerberos** kliensek teljes kelléktárát.

14.8.5. A felhasználók konfigurációs állományai: a `.k5login` és a `.k5users`

Általában az övezetben található felhasználók mindegyikéhez tartozik egy **Kerberos**-szereplő (mint például a `tillman@EXAMPLE.ORG`), ami a felhasználó helyi hozzáférésére mutat (mint például a `tillman` nevű helyi hozzáférés). A `telnet` és a hozzá hasonló kliensalkalmazások általában nem igényelnek felhasználót vagy szereplőt.

Előfordulhat azonban, hogy valaki olyan szeretné elérni egy helyi felhasználó hozzáférését, aki nem rendelkezik a hozzá tartozó **Kerberos**-szereplővel. Például a `tillman@EXAMPLE.ORG` nevű felhasználó el szeretné érni a helyi számítógépen levő `webdevelopers` hozzáférést. Más szereplők is elérhetik a helyi hozzáféréseket.

A probléma megoldásához a felhasználók könyvtárában található `.k5login` és a `.k5users` állományok használhatóak a `.host` és `.rhosts` állományok kombinációjához hasonlóan. Például a `.k5login` így néz ki:

```
tillman@example.org
jdoe@example.org
```

Ezt a `webdevelopers` nevű helyi felhasználó könyvtárában kell elhelyeznünk, így a felsorolt szereplőt megosztott jelszó használata nélkül képesek elérni a hozzáférést.

Az említett parancsok man oldalának elolvasása ajánlott. Megjegyezzük, hogy a `ksu` man oldal foglalkozik a `.k5users` állománnyal.

14.8.6. Tippek, trükkök a Kerberos használatáról és hibaelhárítás

- Akár a **Kerberos** Heimdal vagy az MIT változatát használjuk, ne felejtsük úgy beállítani a `PATH` környezeti változóban felsorolt elérési utakat, hogy a kliensalkalmazások kerberizált változatai a rendszerben használatos verziók elé kerüljenek.
- Az övezetben minden számítógép órája ugyanúgy jár? Ha nem, akkor a hitelesítés csődöt mondhat. A 29.10 Szakaszból tudhatjuk meg hogyan szinkronizáljunk órákat az NTP segítségével.
- Az MIT és a Heimdal verziók a `kadmin` kivételével remekül megvannak egymással, mivel az általa használt protokollt még nem szabványosították.
- Ha megváltoztatjuk a gépünk hálózati nevét, akkor a ugyanígy a `host / szereplő`ket is meg kell változtatni és frissíteni a `keytab` állományunkat. Ez olyan speciális `keytab` bejegyzésekre is vonatkozik, mint például az Apache `www/mod_auth_kerb` moduljához tartozó `www / szereplő`.
- Az övezetünkben levő összes számítógépnek (mind a két irányba) feloldható DNS névvel kell rendelkeznie (vagy legalább egy `/etc/hosts` állománnyal). Erre a `CNAME` rekord megfelelő, de az `A` és `PTR` rekordoknak mindenképpen rendben kell lenniük. Az ilyenkor keletkező hibaüzenet nem éppen fogja meg a lényegét:
`Kerberos5 refuses authentication because Read req failed: Key table entry not found.`
- A kulcselosztó számára kliensként viselkedő bizonyos operációs rendszerek nem állítják be megfelelően a `ksu` engedélyeit, ezért nem lehet `root` jogokkal futtatni. Ezért a `ksu` parancs nem fog működni, ami alapvetően nem egy rossz ötlet, de idegesítő. Ez nem a kulcselosztó hibája.
- Ha a **Kerberos** MIT változatát használjuk és a meg akarjuk hosszabbítani a szereplőknek kiadott jegyek élettartamát az alapértelmezett tíz óráról, akkor a `kadmin` felületén a `modify_principal` paranccsal tudjuk megváltoztatni mind a kérdéses szereplő, mind pedig a `krbtgt` jegyeinek élettartamának maximumát. Ezt követően a szereplő a `kinit -l` opciójával tud egy nagyobb élettartammal rendelkező jegyet kérni.
-

Megjegyzés: Amikor egy kulcselosztóval kapcsolatos hibát próbálunk felderíteni a csomagok lehallgatásával, és a munkaállomásunkról kiadjuk a `kinit` parancsot, akkor arra lehetünk figyelmesek, hogy a TGT már egyből a `kinit` indításakor átküldésre kerül — még mielőtt egyáltalán megadtuk volna a jelszavunkat! Ezt azzal lehet magyarázni, hogy a **Kerberos** szerver bármilyen hitelesítetlen kérésre elküld egy TGT-t (Jegyadó jegy, azaz Ticket Granting Ticket). Azonban mindegyik ilyen TGT a felhasználó jelszavából származtatott kulccsal titkosított. Ezért amit a felhasználó jelszóként megad, nem megy el a kulcselosztónak, hanem vele a `kinit` a már megkapott TGT-t kódolja ki. Amennyiben a visszakódolás egy érvényes időbélyeggel rendelkező, használható jegyet eredményez, akkor a felhasználó érvényes **Kerberos** hitelesítést szerez. Ez a hitelesítés magában foglal egy kulcsot, amellyel a későbbiekben a **Kerberos** szerverekkel tudjuk felvenni biztonságos módon a kapcsolatot, és rajta kívül egy újabb jegyadó jegyet, amelyet a **Kerberos** szerver a saját kulcsával titkosított. A titkosítás második vonala a felhasználó számára ismeretlen, de segítségével a **Kerberos** szerver képes ellenőrizni az egyes jegyadó jegyek hitelességét.

- Ha a jegyeket hosszabb (például egyhetes) élettartammal akarjuk használni és a jegyeket tároló géphez **OpenSSH** segítségével csatlakozunk, akkor mindenképpen ellenőrizzük, hogy az `sshd_config` állományban a **Kerberos** `TicketCleanup` beállításának értéke `no`, máskülönben a kijelentkezés után automatikusan törölődnek a jegyeink.
- Ne hagyjuk figyelmen kívül azt sem, hogy a befogadó szereplők is rendelkezhetnek nagyobb élettartamú jegyekkel. Ha a felhasználónkhoz tartozó szereplő jegye például egy hét alatt évül el, de a számítógép, amire bejelentkezünk, csupán kilenc óráig tartja életben ezeket, akkor a jegyeket tároló gyorsítótárunkban hamarabb elévül a hozzá tartozó jegy, ami miatt pedig hibák keletkeznek.
- Ha a rossz jelszavak használata ellen beállítjuk a `krb5.dict` állományt (erről a `kadmind` man oldalán találunk egy rövid leírást), akkor nem szabad elfelejteni, hogy ez csak olyan szereplőkre vonatkozik, akiknek a jelszáva is állítottunk be szabályozásokat. A `krb5.dict` állományok felépítése nem bonyolult: minden sorban egyetlen karakterlánc szerepel. Érdemes lehet például létrehozni ezen a néven egy szimbolikus linket a `/usr/share/dict/words` állományra.

14.8.7. Eltérések az MIT porttól

A Heimdal és az MIT változatok közti egyik legnagyobb eltérés a `kadmin` programmal kapcsolatban van, ami eltérő (de egyébként ekivalens) parancskészlettel rendelkezik és más protokollt használ. Ennek komoly következménye, hogy ha az MIT-féle kulcselosztót használjuk, akkor azt a Heimdal `kadmin` felületével nem tudjuk távolról adminisztrálni (és vica versa).

A klienskalkalmazások paraméterezése is eltérhet ugyanazon feladatoknál. Ezért velük kapcsolatban az MIT **Kerberos** honlapja (<http://web.mit.edu/Kerberos/www/>) a mérvadó. Vigyázzunk az elérési utakkal: az MIT port magát alapértelmezés szerint a `/usr/local` könyvtárba telepíti, ezért az általuk kiváltani kívánt “normális” rendszerprogramokat esetleg hamarabb találja meg a rendszer, ha nem jól állítottuk be a `PATH` környezeti változónkat.

Megjegyzés: Ha nem értjük, hogy miért működnek olyan furcsán a `telnetd` és a `klogind` által kezelt bejelentkezések, akkor olvassuk el a FreeBSD `security/krb5` portjával települő MIT változat `/usr/local/share/doc/krb5/README.FreeBSD` állományt (angolul). Az a legfontosabb, hogy a `incorrect permissions on cache file` hiba eltüntetéséhez a `login.krb5` binárist kell használnunk, így a továbbított jogosultságoknak megfelelően át tudja állítani a tulajdonost.

Az `rc.conf` állományt is módosítani kell a következő beállítás kialakításához:

```
kerberos5_server="/usr/local/sbin/krb5kdc"
kadmind5_server="/usr/local/sbin/kadmind"
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

Erre azért van szükség, mert a **Kerberos** MIT változata a `/usr/local` könyvtáron belülre telepíti fel a hozzá tartozó alkalmazásokat.

14.8.8. A Kerberosban talált korlátozások enyhítése

14.8.8.1. A Kerberos a “mindent vagy semmit” megközelítést követi

A hálózaton minden szolgáltatást módosítanunk kell ahhoz, hogy együtt tudjanak működni a **Kerberos**szal (vagy valamilyen más módon védenünk kell ezeket a támadások ellen), különben a felhasználók jogait el lehet lopni vagy újra fel lehet használni. Erre jó példa lehet az összes távoli parancssoros elérés (például az `rsh` valamint a `telnet`) kerberizálása, de a jelszavakat titkosítatlanul küldő POP3 levelező szerver kihagyása.

14.8.8.2. A Kerberos az egyfelhasználós munkaállomások számára készült

Többfelhasználós környezetben a **Kerberos** már nem annyira biztonságos. Ez azért mondható el, mert a jegyeket a mindenki által olvasható `/tmp` könyvtárban tárolja. Ha az adott felhasználó számítógépét egyszerre több emberrel is megosztja (tehát többfelhasználós), akkor a felhasználó jegyeit egy másik felhasználó bármikor lemásolhatja (ellophatja).

Ezt a `-c` opció után megadott állománynévvel vagy (inkább) a `KRB5CCNAME` környezeti változó megfelelő beállításával tudjuk áthidalni, habár ezt ritkán teszik is meg. Ha a felhasználók könyvtárában és a megfelelő engedélyekkel tároljuk ezeket a jegyeket, akkor némileg visszaszoríthatjuk a probléma kockázatát.

14.8.8.3. A kulcselosztó a rendszer legsebezhetőbb pontja

A rendszer kialakításából fakadóan a kulcselosztónak legalább annyira megbízhatónak kell lennie, mint a rajta levő központi jelszóadatbázisnak. A kulcselosztón semmilyen más szolgáltatás nem futhat és fizikailag is biztonságba kell helyezni. A kockázat nagy, mivel a **Kerberos** az összes jelszót ugyanazzal a kulccsal (a “mesterkulccsal”) titkosítja, amelyet a kulcselosztó egy állományban tárol.

Széljegyzet gyanánt hozzátesszük, hogy a mesterkulcs elvesztése nem annyira rossz, mint azt első gondolnánk. A mesterkulcsot csupán a véletlenszám-generátor inicializálásához használják a **Kerberos** adatbázisának titkosításakor. Amíg a kulcselosztóhoz nem tudnak illetéktelenek hozzáférni, addig nem tudnak sokat kezdeni a mesterkulccsal.

Mellesleg ha a kulcselosztó nem elérhető (talán pontosan egy DoS támadás vagy éppen hálózati problémák miatt), akkor a hitelesítés nem végezhető el, mivel így a hozzá szükséges hálózati szolgáltatások sem használhatóak. Ez remek eszköz egy DoS támadáshoz. Ezen több (egy központi és egy vagy több alárendelt) kulcselosztó telepítésével, valamint a másodlagos vagy tartalékként használt hitelesítési eszközök (a PAM erre tökéletes) körütekintő megvalósításával enyhíthetünk.

14.8.8.4. A Kerberos hiányosságai

A **Kerberos** révén a felhasználók, számítógépek és szolgáltatások tudják egymást hitelesíteni. Ellenben semmilyen eszközt nem kínál fel a kulcselosztó hitelességének ellenőrzésére. Így tehát (például) egy eltérített `kinit` képes ellopnia az összes felhasználói nevet és jelszót. Az ilyen incidensek elkerülésére a `security/tripwire` és a hozzá hasonló segédprogramok segítségével lehet megőrizni a rendszer sértelenségét.

14.8.9. Erőforrások és további információk

- A **Kerberos** GYIK (angolul) (<http://www.faqs.org/faqs/Kerberos-faq/general/preamble.html>)

- Egy hitelesítési rendszer kidolgozása: párbeszéd négy színben (angolul) (<http://web.mit.edu/Kerberos/www/dialogue.html>)
- RFC 1510: A **Kerberos** hálózati hitelesítési szolgáltatás (V5) (angolul) (<http://www.ietf.org/rfc/rfc1510.txt?number=1510>)
- Az MIT **Kerberos** honlapja (angolul) (<http://web.mit.edu/Kerberos/www/>)
- A Heimdal **Kerberos** honlapja (angolul) (<http://www.pdc.kth.se/heimdal/>)

14.9. OpenSSL

Írta: Tom Rhodes.

A FreeBSD-hez adott **OpenSSL** az egyik olyan tényező, amit a legtöbb felhasználó figyelmen kívül hagy. Az **OpenSSL** egy titkosítási réteget nyújt a hagyományos kommunikációs csatorna felett, így rengeteg hálózati alkalmazásba és szolgáltatásba bele lehet szőni.

Az **OpenSSL** felhasználható többek közt a levelező kliensek titkosított hitelesítésére, hitelkártyás fizetések weben keresztüli lebonyolítására alkalmas, és még sok minden másra. Sok port, köztük a `www/apache13-ssl` és a `mail/sylpheed-claws` is felajánlja az **OpenSSL** felhasználását.

Megjegyzés: A legtöbb esetben a Portgyűjtemény megpróbálja lefordítani a `security/openssl` portot, hacsak a `WITH_OPENSSL_BASE` változót határozottan a "yes" értékre nem állítjuk.

A FreeBSD-hez mellékelt **OpenSSL** ismeri a Secure Sockets Layer v2/v3 (SSLv2/SSLv3) és Transport Layer Security v1 (TLSv1) hálózatz biztonsági protokollokat, és általános célú titkosítási könyvtárként is alkalmazható.

Megjegyzés: Noha az **OpenSSL** ismeri az IDEA algoritmusát is, az Egyesült Államokban érvényben levő szabadalmak miatt alapértelmezés szerint nem engedélyezett. A használatához el kell olvasni a hozzá tartozó licencet, és ha elfogadjuk a benne foglaltakat, akkor állítsuk be a `MAKE_IDEA` változót a `make.conf` állományban.

Az **OpenSSL**-t leginkább a szoftverek tanúsítványainak elkészítéséhez használják. Ilyen tanúsítványokkal lehet szavatolni, hogy az érte felelős cég vagy egyén valóban megbízható és nem szélhámos. Amennyiben a kérdéses tanúsítványt nem vizsgálta be valamelyik "tanúsítványok hitelesítésével foglalkozó hatóság" (Certificate Authority, vagy CA), akkor erről általában kap egy figyelmeztetést a felhasználó. A tanúsítványokat hitelesítő cégek, mint például a VeriSign (<http://www.verisign.com>), írják alá ezeket a tanúsítványokat és ezzel érvényesítik az egyes cégek vagy egyének megbízhatóságát. Ez ugyan pénzbe kerül, de használatuk egyáltalán nem is kötelező. Azonban az átlagosnál paranoidabb felhasználók számára megnyugvást jelenthet.

14.9.1. Tanúsítványok előállítás

A tanúsítványok létrehozására a következő parancs áll rendelkezésre:

```
# openssl req -new -nodes -out req.pem -keyout cert.pem
Generating a 1024 bit RSA private key
.....++++++
```

```

.....+++++
writing new private key to 'cert.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:országnev (kétbetűs kóddal)
State or Province Name (full name) [Some-State]:állam vagy tartomány teljes neve
Locality Name (eg, city) []:település neve
Organization Name (eg, company) [Internet Widgits Pty Ltd]:szervezet neve
Organizational Unit Name (eg, section) []:szervezeti egység neve
Common Name (eg, YOUR name) []:általános név (hálózati név!)
Email Address []:e-mail cím

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:VALAMILYEN JELSZÓ
An optional company name []:egy másik szervezet neve

```

Az adatok bekérésére előtt megjelenő figyelmeztető üzenet fordítása:

Itt a tanúsítvány igénylésével kapcsolatos információkat kell megadnunk. Itt egy ún. "ismertetőnevet" (Distinguished Name, DN) kell megadnunk. Ezen kívül van még néhány más mező is, de ezeket akár üresen is hagyhatjuk. Néhány mezőnek van alapértelmezett értéke, de ha oda egy pontot írunk, akkor kitöröljük.

A "Common Name" mezőnél ellenőrzési okokból egy hálózati nevet, tehát a szerverünk nevét kell megadnunk. Ha nem így járunk el, akkor lényegében egy használhatatlan tanúsítványt kapunk. További opciók is elérhetőek, mint például a lejárat idő (expire time) megadása, a titkosítási algoritmus megváltoztatása stb. Ezek teljes listája megtalálható az openssl(1) man oldalon.

Az előbbi parancs kiadása után két állománynak kell létrejönnie az aktuális könyvtárban. A tanúsítványkérést, vagyis az req.pem állományt kell eljuttatnunk a tanúsítványok hitelesítésével foglalkozó szervhez, aki majd érvényesíti az imént megadott adatainkat. A második, cert.pem nevű állomány a tanúsítványhoz tartozó privát kulcs, amit semmilyen körülmények között sem szabad kiadnunk. Ha ez mások kezébe kerül, akkor el tudnak játszani bennünket (vagy a szerverünket).

Amikor a hitelesítő szerv aláírása nem feltétlenül szükséges, akkor készíthetünk egy saját magunk által aláírt tanúsítványt is. Ehhez először is generálnunk kell egy RSA-kulcsot:

```
# openssl dsaparam -rand -genkey -out saját_RSA.kulcs 1024
```

Most pedig készítsünk el a hitelesítő szerv kulcsát is:

```
# openssl gendsa -des3 -out hitelesítő.kulcs saját_RSA.kulcs
```

Ezzel a kulccsal most gyártsunk le egy tanúsítványt:

```
# openssl req -new -x509 -days 365 -key hitelesítő.kulcs -out új.tanúsítvány
```

Ekkor két új állomány keletkezik a könyvtárunkban: a hitelesítő szerv aláírása, a *hitelesítő.kulcs* és maga a tanúsítvány, az *új.tanúsítvány* állomány. Ezeket tegyük az */etc* könyvtáron belül egy olyan könyvtárba, amelyet csak a *root* tud olvasni. A *chmod* paranccsal állítsunk be rá 0700-as kódú engedélyeket.

14.9.2. Példa a tanúsítványok használatára

Mire is jók ezek az állományok? Például kitűnően alkalmazhatóak a **Sendmail** levelező szerverhez beérkező kapcsolatokat titkosítására. Így lényegében felszámoljuk minden olyan felhasználó titkosítatlan módon zajló hitelesítését, aki a helyi levelező szerveren keresztül küldi a leveleit.

Megjegyzés: Ez általában nem a legjobb megoldás, mivel egyes levelező kliensek hibát jeleneznek a felhasználónak, ha nem rendelkezik a tanúsítvánnyal. A tanúsítványok telepítésével kapcsolatban olvassuk el a szoftverhez adott leírást.

A helyi *.mc* állományba ezeket a sorokat kell beletenni:

```
dnl SSL Options
define(`confCACERT_PATH', `/etc/certs')dnl
define(`confCACERT', `/etc/certs/új.tanúsítvány')dnl
define(`confSERVER_CERT', `/etc/certs/új.tanúsítvány')dnl
define(`confSERVER_KEY', `/etc/certs/hitelesítő.kulcs')dnl
define(`confTLS_SRV_OPTIONS', `V')dnl
```

Itt a */etc/certs/* az a könyvtár, amit tanúsítványok és kulcsok helyi tárolására használunk. Végezetül még újra kell generálnunk a helyi *.cf* állományokat. Ezt a */etc/mail* könyvtárban a *make install* parancs kiadásával könnyen elvégezhetjük. Miután ez megtörtént, akkor **Sendmail**hoz tartozó démont a *make restart* paraméterével indíthatjuk újra.

Ha minden jól ment, akkor a */var/log/maillog* állományban nem találunk egyetlen hibaüzenetet sem, és a **Sendmail** is megjelenik a futó programok között.

A *telnet(1)* segédprogrammal így próbálhatjuk ki a levelező szerveret:

```
# telnet example.com 25
Trying 192.0.34.166...
Connected to example.com.
Escape character is '^]'.
220 example.com ESMTP Sendmail 8.12.10/8.12.10; Tue, 31 Aug 2004 03:41:22 -0400 (EDT)
ehlo example.com
250-example.com Hello example.com [192.0.34.166], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-STARTTLS
250-DELIVERBY
250 HELP
```

```
quit
221 2.0.0 example.com closing connection
Connection closed by foreign host.
```

Ha itt megjelenik a “STARTTLS” sor, akkor mindent sikerült beállítanunk.

14.10. VPN IPsec felett

Írta: Nik Clayton.

VPN létrehozása FreeBSD átjárók használatával két olyan hálózat között, amelyeket egymástól az internet választ el.

14.10.1. Az IPsec bemutatása

Írta: Hiten M. Pandya.

Ebben a szakaszban az IPsec beállításának folyamatát vázoljuk fel. Az IPsec beállításához elengedhetetlen, hogy tisztában legyünk egy saját rendszer mag fordításának alapjaival (lásd 8 fejezet).

Az *IPsec* egy olyan protokoll, amely az Internet Protocol (IP) rétegére épül. Segítségével két vagy több számítógép képes biztonságos módon tartani egymással a kapcsolatot (innen ered a neve). A FreeBSD IPsec “hálózati protokollkészlete” a KAME (<http://www.kame.net>) implementációjára épül, mely egyaránt támogatja az IPv4 és IPv6 protokollcsaládokat.

Az IPsec két alprotokollból tevődik össze:

- A *hasznos adat biztonságos becsomagolása (Encapsulated Security Payload, ESP)* során egy szimmetrikus kriptográfiai algoritmussal (mint például Blowfish, 3DES) titkosítjuk az IP-csomagok tartalmát, ezáltal megvédjük ezeket az illetéktelenektől.
- A *Hitelesítési fejléc (Authentication Header, AH)* használatával megakadályozzuk, hogy az illetéktelenek meghamisítsák az IP csomagok fejlécét. Ezt úgy érjük el, hogy kiszámolunk egy kriptográfiai ellenőrző összeget és az IP-csomagok fejlécének mezőire egy biztonságos függvénnyel generálunk valamilyen ujjlenyomatot. Az ez után következő kiegészítő fejléc tartalmazza ezt az ujjlenyomatot, amellyel a csomag hitelesíthető.

Az ESP és az AH az alkalmazástól függően használható együtt vagy külön-külön.

Az IPsec akár közvetlenül is használható két számítógép forgalmának titkosítására (ezt *Szállítási módnak (Transport Mode)* nevezik), vagy két alhálózat között építhetünk ki vele “virtuális tunneleket”, ami remekül alkalmas két vállalati hálózat kommunikációjának bebiztosítására (ez a *Tunnel mód (Tunnel Mode)*). Ez utóbbit egyszerűen csak *Virtuális magánhálózatként (Virtual Private Network, VPN)* emlegetik. A FreeBSD IPsec alrendszeréről az `ipsec(4)` man oldalon találhatunk további információkat.

A rendszer mag IPsec támogatásának aktiválásához a következő paramétereket kell beletennünk a konfigurációs állományba:

```
options    IPSEC          # IP biztonság
device     crypto
```

Ha szükségünk van a IPsec nyomkövetésére, a következő beállítást is hozzátehetjük:

```
options    IPSEC_DEBUG    # az IP biztonság nyomkövetése
```

14.10.2. A probléma

Semmilyen szabvány nem fogalmazza meg mi is számít VPN-nek. A virtuális magánhálózatok tucatnyi különböző technológiával valósíthatók meg, de mindegyiknek megvan a maga erőssége és gyengesége. Ebben a szakaszban körvonalazunk egy ilyen helyzetet, valamint a benne felépített VPN megvalósításához alkalmazott stratégiákat.

14.10.3. A forgatókönyv: adott egy otthoni és egy vállalati hálózat, amelyek külön-külön csatlakoznak az internetre, és VPN használatával ezeket egyetlen hálózatként szeretnénk használni

Előfeltételezéseink a következők:

- legalább két hálózatunk van;
- magán belül mind a két hálózat IP-t használ;
- mind a két hálózat egy FreeBSD átjárón keresztül csatlakozik az internethez;
- a hálózatok átjárói legalább egy publikus IP-címmel rendelkeznek;
- a hálózatok belső címei lehetnek publikus vagy privát IP-címek, nem számít. Fontos viszont, hogy ezek ne ütközzenek, vagyis ne használja egyszerre mind a kettő a 192.168.1.x címtartományt.

14.10.4. Az IPsec beállítása FreeBSD alatt

Írta: Tom Rhodes.

Kezdeképpen a Portgyűjteményből telepítenünk kell a `security/ipsec-tools` portot. Ez a programcsomag rengeteg olyan alkalmazást tartalmaz, amely segítségünkre lehet a beállítások elvégzése során.

A következő lépésben létre kell hoznunk két gif(4) típusú pszeudoeszközt, melyeken keresztül a két hálózat között egy tunnel segítségével ki tudjuk építeni a szükséges kapcsolatot. Ehhez `root` felhasználóként futtassuk a következő parancsokat (a *belső* és *külső* megnevezésű paramétereket cseréljük ki a valós belső és külső átjárók címére):

```
# ifconfig gif0 create
# ifconfig gif0 belső1 belső2
# ifconfig gif0 tunnel külső1 külső2
```

Tekintsük például, hogy a vállalati LAN publikus IP-címe 172.16.5.4, valamint a privát IP-címe 10.246.38.1. Az otthoni LAN publikus IP-címe legyen most 192.168.1.12, valamint a belső privát IP-címe pedig 10.0.0.5.

Elsőre ez talán még nem teljesen érthető, ezért az `ifconfig(8)` parancs használatával is nézzük meg a példában szereplő hálózatok konfigurációját:

Az első átjáró:

```
gif0: flags=8051 mtu 1280
tunnel inet 172.16.5.4 --> 192.168.1.12
inet6 fe80::2e0::81ff:fe02:5881%gif0 prefixlen 64 scopeid 0x6
inet 10.246.38.1 --> 10.0.0.5 netmask 0xffffffff00
```

A második átjáró:

```
gif0: flags=8051 mtu 1280
tunnel inet 192.168.1.12 --> 172.16.5.4
inet 10.0.0.5 --> 10.246.38.1 netmask 0xffffffff00
inet6 fe80::250:bfff:fe3a:c1f%gif0 prefixlen 64 scopeid 0x4
```

Miután elvégeztük az iménti beállításokat, a ping(8) paranccsal már mind a két privát IP-tartománynak elérhetőnek kell lennie, ahogy azt az alábbi példa is érzékeltetni kívánja:

```
otthoni-halo# ping 10.0.0.5
PING 10.0.0.5 (10.0.0.5): 56 data bytes
64 bytes from 10.0.0.5: icmp_seq=0 ttl=64 time=42.786 ms
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=19.255 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=20.440 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=21.036 ms
--- 10.0.0.5 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 19.255/25.879/42.786/9.782 ms
```

```
vallalati-halo# ping 10.246.38.1
PING 10.246.38.1 (10.246.38.1): 56 data bytes
64 bytes from 10.246.38.1: icmp_seq=0 ttl=64 time=28.106 ms
64 bytes from 10.246.38.1: icmp_seq=1 ttl=64 time=42.917 ms
64 bytes from 10.246.38.1: icmp_seq=2 ttl=64 time=127.525 ms
64 bytes from 10.246.38.1: icmp_seq=3 ttl=64 time=119.896 ms
64 bytes from 10.246.38.1: icmp_seq=4 ttl=64 time=154.524 ms
--- 10.246.38.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 28.106/94.594/154.524/49.814 ms
```

Az elvárásainknak megfelelően tehát a privát címeken mind a két oldalnak képesnek kell lennie ICMP csomagokat küldenie és fogadnia. A következő lépésben meg kell mondanunk az átjáróknak hogyan irányítsák a csomagokat a két hálózat közti forgalom megfelelő áramlásához. Ezt az alábbi paranccsal elérhetjük el:

```
# vállalati-halo# route add 10.0.0.0 10.0.0.5 255.255.255.0

# vállalati-halo# route add net 10.0.0.0: gateway 10.0.0.5

# otthoni-halo# route add 10.246.38.0 10.246.38.1 255.255.255.0

# otthoni-halo# route add host 10.246.38.0: gateway 10.246.38.1
```

Itt már a belső gépeket az átjárókról és az átjárók mögül egyaránt el tudjuk érni. A következő példa alapján erről könnyedén meg is tudunk győződni:

```
vallalati-halo# ping 10.0.0.8
PING 10.0.0.8 (10.0.0.8): 56 data bytes
64 bytes from 10.0.0.8: icmp_seq=0 ttl=63 time=92.391 ms
```

```

64 bytes from 10.0.0.8: icmp_seq=1 ttl=63 time=21.870 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=63 time=198.022 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=63 time=22.241 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=63 time=174.705 ms
--- 10.0.0.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.870/101.846/198.022/74.001 ms

```

```

otthoni-halo# ping 10.246.38.107
PING 10.246.38.1 (10.246.38.107): 56 data bytes
64 bytes from 10.246.38.107: icmp_seq=0 ttl=64 time=53.491 ms
64 bytes from 10.246.38.107: icmp_seq=1 ttl=64 time=23.395 ms
64 bytes from 10.246.38.107: icmp_seq=2 ttl=64 time=23.865 ms
64 bytes from 10.246.38.107: icmp_seq=3 ttl=64 time=21.145 ms
64 bytes from 10.246.38.107: icmp_seq=4 ttl=64 time=36.708 ms
--- 10.246.38.107 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.145/31.721/53.491/12.179 ms

```

A tunnelek beállítása volt igazából a könnyebb rész, egy biztonságos összeköttetés kialakítása azonban már valamivel komolyabb folyamatot rejt magában. A most következő konfigurációban erre “előre ismert” (vagyis pre-shared, PSK) RSA-kulcsokat fogunk használni. A konkrét IP-címeiktől eltekintve az átjárókon a /usr/local/etc/racoon/racoon.conf állományok hasonlóan fognak kinézni, nagyjából valahogy így:

```

path    pre_shared_key "/usr/local/etc/racoon/psk.txt"; # az ismert kulcsot tartalmazó állomány h
log      debug; # a naplózás részletességének beállítása: ha végeztünk a teszteléssel és a hibakeresé
padding  # ezeket ne nagyon változtassuk meg
{
    maximum_length  20;
    randomize        off;
    strict_check     off;
    exclusive_tail   off;
}

timer   # időzítési beállítások, állítsuk be igény szerint
{
    counter          5;
    interval          20 sec;
    persend           1;
#    natt_keepalive  15 sec;
    phasel            30 sec;
    phase2            15 sec;
}

listen  # cím [port], ahol a racoon majd válaszolni fog
{
    isakmp            172.16.5.4 [500];
    isakmp_natt        172.16.5.4 [4500];
}

remote  192.168.1.12 [500]
{

```

```

exchange_mode    main,aggressive;
doi              ipsec_doi;
situation        identity_only;
my_identifier     address 172.16.5.4;
peers_identifier  address 192.168.1.12;
lifetime         time 8 hour;
passive          off;
proposal_check    obey;
# nat_traversal   off;
generate_policy   off;

                    proposal {
                        encryption_algorithm    blowfish;
                        hash_algorithm          md5;
                        authentication_method    pre_shared_key;
                        lifetime time           30 sec;
                        dh_group                1;
                    }
}

sainfo (address 10.246.38.0/24 any address 10.0.0.0/24 any) # address $hálózat/$hálózati_maszk $
        # (a $típus lehet "any" vagy "esp")
{
        # a $hálózat a két összekapcsolni kívánt belső hálózat legyen
    pfs_group      1;
    lifetime        time    36000 sec;
    encryption_algorithm    blowfish,3des,des;
    authentication_algorithm    hmac_md5,hmac_shal;
    compression_algorithm    deflate;
}

```

A példában szereplő összes opció részletes kifejtése jóval meghaladná ezen leírás kereteit, ezért a bővebb információkkal kapcsolatban inkább a **racoona** beállításaihoz tartozó man oldal elolvasását javasoljuk.

A gépek közti hálózati forgalom titkosításához be kell még állítanunk egy SPD házirendet is, így a FreeBSD és a **racoona** képes kódolni és dekódolni a csomagokat.

Ezt a most következő, a vállalati átjárón találhatóhoz hasonló egyszerű shell szkripttel tudjuk elvégezni. Ezt az állományt a rendszer indításakor fogjuk felhasználni, melyet `/usr/local/etc/racoona/setkey.conf` néven mentünk el:

```

flush;
spdf flush;
# Az otthoni hálózati felé
spdadd 10.246.38.0/24 10.0.0.0/24 any -P out ipsec esp/tunnel/172.16.5.4-192.168.1.12/use;
spdadd 10.0.0.0/24 10.246.38.0/24 any -P in ipsec esp/tunnel/192.168.1.12-172.16.5.4/use;

```

Ahogy ezzel megvagyunk, a **racoona** az egyes átjárókon a következő paranccsal indítható el:

```
# /usr/local/sbin/racoona -F -f /usr/local/etc/racoona/racoona.conf -l /var/log/racoona.log
```

A parancs eredménye ennek megfelelően nagyjából a következő lesz:

```
vallalati-halo# /usr/local/sbin/racoona -F -f /usr/local/etc/racoona/racoona.conf
Foreground mode.
```

```

2006-01-30 01:35:47: INFO: begin Identity Protection mode.
2006-01-30 01:35:48: INFO: received Vendor ID: KAME/racoon
2006-01-30 01:35:55: INFO: received Vendor ID: KAME/racoon
2006-01-30 01:36:04: INFO: ISAKMP-SA established 72.16.5.4[500]-192.168.1.12[500] spi=623b9b3bd24
2006-01-30 01:36:05: INFO: initiate new phase 2 negotiation: 72.16.5.4[0]192.168.1.12[0]
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 92.168.1.12[0]->172.16.5.4[0] spi=284
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]->192.168.1.12[0] spi=47
2006-01-30 01:36:13: INFO: respond new phase 2 negotiation: 172.16.5.4[0]192.168.1.12[0]
2006-01-30 01:36:18: INFO: IPsec-SA established: ESP/Tunnel 192.168.1.12[0]->172.16.5.4[0] spi=12
2006-01-30 01:36:18: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]->192.168.1.12[0] spi=17

```

A tunnel megfelelő működését úgy tudjuk ellenőrizni, ha átváltunk egy másik konzolra és a tcpdump(1) program segítségével figyeljük a hálózati forgalmat. A példában szereplő em0 interfészt természetesen ne felejtjük el kicserélni a megfelelő eszköz nevére.

```
# tcpdump -i em0 host 172.16.5.4 and dst 192.168.1.12
```

Ennek hatására az alábbiakhoz hasonló adatoknak kellene megjelennie a konzolon. Amennyiben nem ez történik, valamilyen hiba történt, ezért meg kell keresnünk azt a visszakapott adatok alapján.

```

01:47:32.021683 IP vállalatihalozat.com > 192.168.1.12.otthonihalozat.com: ESP(spi=0x02acbf9f,seq
01:47:33.022442 IP vállalatihalozat.com > 192.168.1.12.otthonihalozat.com: ESP(spi=0x02acbf9f,seq
01:47:34.024218 IP vállalatihalozat.com > 192.168.1.12.otthonihalozat.com: ESP(spi=0x02acbf9f,seq

```

Itt már mind a két hálózatnak elérhetőnek kell lennie és egyként kell látszódnia. A hálózatokat ezen felül még érdemes külön védeni egy tűzfalal is. Ilyenkor a csomagok két hálózati közti zavartalan oda-vissza vándorlásához további szabályokat kell még felvennünk a tűzfal szabályrendszerébe. A ipfw(8) tűzfal esetén ez a következő sorok hozzáadását jelenti a tűzfal konfigurációs állományához:

```

ipfw add 00201 allow log esp from any to any
ipfw add 00202 allow log ah from any to any
ipfw add 00203 allow log ipencap from any to any
ipfw add 00204 allow log udp from any 500 to any

```

Megjegyzés: A szabályok számozását mindig az adott gép aktuális beállításainak megfelelően kell módosítani.

A pf(4) és ipf(8) felhasználók számára ehhez a következő parancsot javasoljuk:

```

pass in quick proto esp from any to any
pass in quick proto ah from any to any
pass in quick proto ipencap from any to any
pass in quick proto udp from any port = 500 to any port = 500
pass in quick on gif0 from any to any
pass out quick proto esp from any to any
pass out quick proto ah from any to any
pass out quick proto ipencap from any to any
pass out quick proto udp from any port = 500 to any port = 500
pass out quick on gif0 from any to any

```

Végezetül a következő sor hozzáadásával engedélyezzük az /etc/rc.conf állományban a VPN indítását a rendszer indítása során:

```

ipsec_enable="YES"
ipsec_program="/usr/local/sbin/setkey"
ipsec_file="/usr/local/etc/racoon/setkey.conf" # engedélyezzük az spd házirend beállítását a rend.
racoon_enable="yes"

```

14.11. OpenSSH

Írta: Chern Lee.

Az **OpenSSH** olyan hálózati kapcsolódási eszközök összessége, amivel biztonságos módon érhetünk el távoli számítógépeket. Az `rlogin`, `rsh`, `rcp` és a `telnet` direkt kiváltására használható. Emellett SSH-n keresztül TCP/IP kapcsolatok is biztonságosan bújthatóak vagy küldhetőek tovább.

Az **OpenSSH**-t az OpenBSD projekt tartja karban, és az SSH 1.2.12 verziójára épül hibajavításokkal és frissítésekkel egyetemben. Az SSH 1 és 2 protokollokkal egyaránt kompatibilis.

14.11.1. Az OpenSSH használatának előnyei

A hétköznapi esetben, vagyis amikor a `telnet(1)` vagy `rlogin(1)` alkalmazásokat használjuk, az adatok titkosítatlan formában közlekednek a hálózaton. A szerver és a kliens közé bárhova becsatlakozó hálózati kíváncsiskodók így könnyedén el tudják lopni a felhasználói nevünket és jelszavunkat, vagy lényegében bármilyen adatot, ami az adott munkamenetben megfordul. Az **OpenSSH** ennek kivédésére kínál fel különféle hitelesítési és titkosítási eszközöket.

14.11.2. Az sshd engedélyezése

Az **sshd** a FreeBSD telepítésekor jelentkező Standard lehetőségek egyike. Az **sshd** engedélyezését úgy tudjuk kideríteni, ha az `rc.conf` állományban megkeressük a következő sort:

```
sshd_enable="YES"
```

Ez tölti be a rendszer indításakor az `sshd(8)`-t, az **OpenSSH** démonát. Vagy az `/etc/rc.d/sshd rc(8)` szkript segítségével is elindíthatjuk az **OpenSSH**-t:

```
/etc/rc.d/sshd start
```

14.11.3. Az SSH kliens

Az `ssh(1)` segédprogram az `rlogin(1)` programhoz hasonlóan működik.

```

# ssh felhasználó@gép.hu
Host key not found from the list of known hosts. Are you sure you
want to continue connecting (yes/no)? yes Host
'gép.hu' added to the list of known hosts.
felhasználó@gép.hu's password:
*****

```

Az üzenetek fordítása:

Nem találtam meg a gépet az ismert gépek között. Biztosan csatlakozni akarunk hozzá (igen/nem)? **igen** A 'gép.hu' felkerült az ismert gépek közé.
Adja meg a felhasználó@gép.hu jelszavát:

Bejelentkezés után minden ugyanolyan, mintha az rlogin vagy a telnet programokat használtuk volna. Az SSH egy kulcs segítségével próbálja azonosítani a számítógépeket, ezzel ellenőrzi a szerver hitelességét a kliensek csatlakozásakor. A felhasználónak ilyenkor először mindig **yes** választ kell adnia. A későbbi bejelentkezési kísérletek pedig majd mindig az így kapott kulccsal történnek. Ha eltérne a kulcs, akkor az SSH kliens erre figyelmeztetni fog minket. A kulcsok a `~/.ssh/known_hosts` vagy az SSH v2 protokoll esetén a `~/.ssh/known_hosts2` állományba kerülnek elmentésre.

Alapértelmezés szerint az **OpenSSH** szerverek csak SSH v2 kapcsolatokat fogadnak el. Lehetőség szerint a kliens is ezt a változatot fogja használni, de ha nem sikerül, akkor megpróbálkozik a v1-el. A kliensnek a `-1` vagy `-2` opciók segítségével elő is lehet írni, hogy az első vagy a második változatot használja. A kliensben az első változat támogatását csupán a régebbi verziók kompatibilitása miatt tartják karban.

14.11.4. Biztonságos másolás

Az `scp(1)` parancs az `rcp(1)` parancshoz hasonlóan működik: egyik gépről másol a másikra, biztonságosan.

```
# scp felhasználó@gép.hu:/COPYRIGHT COPYRIGHT
felhasználó@gép.hu's password: *****
COPYRIGHT          100% | ***** | 4735
00:00
#
```

Mivel a kulcsot már ismerjük ehhez a távoli géphez (az előbbi példából), ezért az `scp(1)` használatakor már ezzel hitelesítünk.

Az `scp(1)` paraméterei hasonlóak a `cp(1)` parancséhoz: első helyen az állomány vagy állományok neveit adjuk meg, a másodikon pedig a célt. Mivel az állományokat a hálózaton SSH-n keresztül küldik át, ezért az állományok neveit `felhasználó@gép:elérési_út` formában kell megadni.

14.11.5. Beállítások

Az **OpenSSH** démon és kliens rendszerszintű konfigurációs állományai az `/etc/ssh` könyvtárban találhatók.

Az `ssh_config` tartalmazza a kliens beállításait, miközben az `sshd_config` tartalmazza a démonét.

Emellett az `rc.conf` állományban megadható `sshd_program` (ez alaphoz a `/usr/sbin/sshd`) és `sshd_flags` opciókkal további beállítási szinteket nyújtanak.

14.11.6. ssh-keygen

Jelszavak helyett az `ssh-keygen(1)` programmal a felhasználók azonosítására DSA- vagy RSA-kulcsokat tudunk készíteni:

```
% ssh-keygen -t dsa
Generating public/private dsa key pair.
```

```

Enter file in which to save the key (/home/felhasználó/.ssh/id_dsa):
Created directory '/home/felhasználó/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/felhasználó/.ssh/id_dsa.
Your public key has been saved in /home/felhasználó/.ssh/id_dsa.pub.
The key fingerprint is:
bb:48:db:f2:93:57:80:b6:aa:bc:f5:d5:ba:8f:79:17 felhasználó@gép.hu

```

Az `ssh-keygen(1)` ekkor a hitelesítésre létrehoz egy publikus és egy privát kulcsból álló párt. A privát kulcs a `~/.ssh/id_dsa` vagy `~/.ssh/id_rsa` állományba kerül, miközben a publikus kulcs a `~/.ssh/id_dsa.pub` vagy `~/.ssh/id_rsa.pub` lesz attól függően, hogy DSA vagy RSA a kulcs típusa. A módszer működéséhez a publikus DSA- vagy RSA-kulcsot a távoli számítógép `~/.ssh/authorized_keys` állományába kell bemásolni.

Így tehát a távoli számítógépre jelszavak alkalmazása helyett SSH-kulccsal tudunk belépni.

Ha az `ssh-keygen(1)` parancsnak megadunk egy jelmondatot is, akkor a felhasználó a privát kulcsát csak ennek megadásával tudja használni. A hosszú jelmondatok állandó beírogatásától a 14.11.7 Szakasz szakaszban hamarosan bemutatásra került `ssh-agent(1)` igyekszik megkímélni minket.

Figyelem A különböző opciók és állományok eltérhetnek a számítógépünkre telepített **OpenSSH** verziójától függően. Ilyen esetben javasolt felkeresni az `ssh-keygen(1)` man oldalát.

14.11.7. Az `ssh-agent` és az `ssh-add`

Az `ssh-agent(1)` és `ssh-add(1)` segédprogramokkal be tudjuk tölteni az **SSH**-kulcsokat a memóriába, amivel elkerülhetjük a jelmondat állandó begépelését.

A hitelesítést az `ssh-agent(1)` program kezeli a betöltött privát kulcsok alapján. Az `ssh-agent(1)` használatával egy másik programot is elindíthatunk, egy parancsértelmezőtől kezdve egy ablakkezelőig szinte bármit.

Az `ssh-agent(1)` programot úgy tudjuk egy parancsértelmezőben használni, hogy először is elindítjuk vele az adott parancsértelmezőt. Ezután az `ssh-add(1)` lefuttatásával hozzá kell adnunk egy identitást, annak jelmondatának megadásával. Miután ezeket megtettük, a felhasználó bármelyik olyan távoli gépre be tud jelentkezni, ahol a publikus kulcsát ismerik. Például:

```

% ssh-agent csh
% ssh-add
Enter passphrase for /home/felhasználó/.ssh/id_dsa:
Identity added: /home/felhasználó/.ssh/id_dsa (/home/felhasználó/.ssh/id_dsa)
%

```

Az `ssh-agent(1)` programot X11-el úgy tudjuk használni, ha az `~/.xinitrc` állományba tesszük bele. Ezzel az `ssh-agent(1)` az összes X11-ben indított program számára rendelkezésre áll. Példának vegyük ezt az `~/.xinitrc` állományt:

```
exec ssh-agent startxfce4
```

Így az X11 indulásakor mindig elindul az ssh-agent(1), amely pedig elindítja az **XFCE** alkalmazást. Miután átírtuk a saját állományunkat, a rendszer életbeléptetéséhez indítsuk újra az X11-et, az ssh-add(1) futtatásával pedig töltsük be az összes SSH-kulcsunkat.

14.11.8. Tunnelezés SSH-val

Az **OpenSSH**-val létre tudunk hozni egy tunnelt, amellyel egy másik protokoll adatait tudjuk titkosított módon becsomagolni.

Az alábbi parancs arra utasítja az ssh(1) programot, hogy hozzon létre egy tunnelt a **telnet** használatához:

```
% ssh -2 -N -f -L 5023:localhost:23 felhasználó@izé.mizé.hu
%
```

Az ssh parancsnak a következő kapcsolókat adtuk meg:

-2

Az ssh parancs a protokoll második változatát használja. (Ne adjuk meg, ha régi SSH szerverekkel dolgozunk.)

-N

Tunnel létrehozása. Ha nem adjuk meg, akkor az ssh egy hagyományos munkamenet felépítését kezdi meg.

-f

Az ssh a háttérben fusson.

-L

Egy helyi tunnel a *helyiport:távoligép:távoliport* felírásban.

felhasználó@izé.mizé.hu

A távoli SSH szerver.

Az SSH által létrehozott járatok úgy működnek, hogy létrehozunk egy csatlakozást a localhost (a helyi gép) megadott portján. Ezután minden olyan kapcsolatot, ami a helyi gép adott portjára érkezik, SSH-n keresztül átírányítunk a távoli gép portjára.

Ebben a példában a helyi gép 5023 portját átírányítjuk a helyi gép 23 portjára. Mivel a 23 a **telnet** portja, ezért az így definiált SSH járattal egy biztonságos **telnet** munkamenetet hozunk létre.

Ezen a módon tetszőleges nem biztonságos TCP protokollt, például SMTP-t, POP3-at, FTP-t stb. be tudunk csomagolni.

Példa 14-1. Biztonságos tunnel létrehozása SSH-val SMTP-hez

```
% ssh -2 -N -f -L 5025:localhost:25 felhasználó@levelező.szerver.hu
felhasználó@levelező.szerver.hu's password: *****
% telnet localhost 5025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^['.
220 levelező.szerver.hu ESMTP
```

Az ssh-keygen(1) és további felhasználói hozzáférések alkalmazásával ezen a módon ki tudunk alakítani egy minden további problémától és zűrtől mentes SSH tunnelezési környezetet. A jelszavak helyett kulcsokat használunk és minden tunnel külön felhasználóként is futtatható.

14.11.8.1. Gyakorlati példák a tunnelek használatára

14.11.8.1.1. Egy POP3 szerver biztonságos elérése

Tegyük fel, hogy a munkahelyünkön van egy SSH szerver, amire kívülről lehet csatlakozni, illetve vele egy hálózatban van egy POP3 levelező szerver is. A munkahelyünk és az otthonunk között levő hálózati útvonalat részben vagy teljesen nem tartjuk megbízhatónak. Ezért az e-mailjeinket valamilyen biztonságos módon szeretnénk elérni. Ezt úgy tudjuk megvalósítani, ha otthonról csatlakozunk a munkahelyen levő SSH szerverre és ezen keresztül érjük a levelező szervert.

```
% ssh -2 -N -f -L 2110:levél.gép.hu:110 felhasználó@ssh-szerver.gép.hu
felhasználó@ssh-szerver.gép.hu's password: *****
```

Miután a tunnel létrejött és működőképes, állítsuk be a levelező kliensünkben, hogy a POP3 kéréseket a localhost 2110 portjára küldje. Innen pedig biztonságos módon megy tovább a levél.gép.hu címre.

14.11.8.1.2. Egy szigorú tűzfal megkerülése

Egyes hálózati adminisztrátorok túlságosan szigorú szabályokat adnak meg a tűzfalban, és nem csak a bejövő kapcsolatokat szűrik, hanem a kimenőket is. A távoli gépekhez csak a 22 (SSH) és 80 (böngészés) portjaikon tudunk csatlakozni.

Mi viszont szeretnénk más (nem egészen a munkánkkal kapcsolatos) szolgáltatásokat is elérni, például egy Ogg Vorbis szerverről zenét hallgatni. Ehhez a szerverhez viszont csak akkor tudnánk csatlakozni, ha a 22 vagy 80 portokon üzemelne.

Ezt a problémát úgy oldhatjuk meg, ha felépítünk egy SSH kapcsolatot a hálózatunk tűzfalán kívül levő számítógéppel és segítségével átbújunk az Ogg Vorbis szerverhez.

```
% ssh -2 -N -f -L 8888:zene.gép.hu:8000 felhasználó@tűzfalazatlan-rendszer.gép.org
felhasználó@tűzfalazatlan-rendszer.gép.org's password: *****
```

A zenelejátszó kliensüknek adjuk meg a localhost 8888 portját, amely pedig a tűzfal sikeres kijátszásával továbbítódik a zene.gép.hu 8000-res portjára.

14.11.9. Az AllowUsers felhasználói beállítás

Gyakran nem árt korlátozni a felhasználók bejelentkezését. Az AllowUsers erre tökéletesen megfelel. Például, ha csak 192.168.1.32 címről engedjük bejelentkezni a root felhasználót, akkor ehhez valami ilyesmit kell beírunk az /etc/ssh/sshd_config állományba:

```
AllowUsers root@192.168.1.32
```

Ezzel pedig csupán nevének megadásával engedélyezzük az `admin` felhasználó bejelentkezését (bárhonnan):

```
AllowUsers admin
```

Egy sorban több felhasználó is megadható, mint például:

```
AllowUsers root@192.168.1.32 admin
```

Megjegyzés: Ilyenkor ne felejtjük el megadni az összes bejelentkezésre (valamilyen formában) jogosult felhasználót megadni, máskülönben kizárjuk ezeket.

Miután elvégeztük a szükséges változtatásokat az `/etc/ssh/sshd_config` állományban, utasítsuk az `sshd(8)` demont a konfigurációs állományok újraolvasására:

```
# /etc/rc.d/sshd reload
```

14.11.10. Ajánlott olvasnivalók (angolul)

OpenSSH (<http://www.openssh.com/>)

```
ssh(1) scp(1) ssh-keygen(1) ssh-agent(1) ssh-add(1) ssh_config(5)
```

```
sshd(8) sftp-server(8) sshd_config(5)
```

14.12. Az állományrendszerek hozzáféréseit vezérlő listák

Írta: Tom Rhodes.

A FreeBSD 5.0 és későbbi változatai különböző fejlesztéseket hoztak az állományrendszerekben, például a pillanatképek készítése vagy a hozzáférés-vezérlési listák (Access Control List, ACL-ek) támogatása.

A hozzáférés-vezérlési listák a szabványos UNIX-os engedély modellt bővítik ki egy igen kompatibilis (POSIX.1e) módon. Használatával a rendszergazdák egy sokkal kifinomultabb biztonsági modellt tudhatnak a kezük ügyében.

Az UFS állományrendszerek ACL támogatását úgy tudjuk engedélyezni, ha a rendszermagot az

```
options UFS_ACL
```

paraméterrel fordítjuk le. Amennyiben ezt nem fordítottuk bele, akkor az ACL támogatással rendelkező állományrendszerek csatlakoztatása során egy figyelmeztetést kapunk. Ez az opció a `GENERIC` rendszermag része. Az ACL az állományrendszeren engedélyezett kiterjesztett tulajdonságokra támaszkodik. Ezeket a kiterjesztett tulajdonságokat a következő generációs UNIX állományrendszer, az UFS2 már alapból ismeri.

Megjegyzés: UFS1 típusú állományrendszereken sokkal nagyobb a kiterjesztett tulajdonságok kezelésének költsége, mint az UFS2 esetében. Az UFS2 jóval nagyobb teljesítménnyel képes dolgozni a kiterjesztett tulajdonságokkal. Emiatt a hozzáférés-vezérlési listák használatához az UFS2 sokkal inkább ajánlott, mint az UFS1.

Az ACL használatát a csatlakoztatáskor megadott `acls` beállítással engedélyezhetjük, amelyet érdemes felvennünk az `/etc/fstab` állományba. Ha a `tunefs(8)` segédprogrammal az állományrendszer fejlécében levő szuperblokk ACL kapcsolóját átírjuk, akkor ez a beállítás automatikussá tehető. A szuperblokk használata több okból is ajánlatos:

- A csatlakoztatáskor megadott ACL beállítás nem változtatható egy egyszerű újracsatlakoztatással (`mount(8) -u`), csak egy teljes leválasztással (`umount(8)`) és egy friss csatlakoztatással (`mount(8)`). Ennek értelmében az ACL-ek a rendszerindító állományrendszeren a rendszer indulása után nem engedélyezhetők. Ám ez azt is jelenti, hogy egy már használatban levő állományrendszer beállításai sem változtathatók meg.
- Ha a kapcsolót a szuperblokkban állítjuk be, akkor az állományrendszert még akkor is ACL támogatással csatlakoztatja a rendszer, ha azt nem adtuk meg az `fstab` állományban vagy az eszközeink átrendeződtek. Így az állományrendszereket még véletlenül sem tudjuk ACL használata nélkül csatlakoztatni, ami egyébként így komoly biztonsági problémákat okozhatna.

Megjegyzés: Beállíthatjuk úgy is ACL kezelését, hogy egy friss csatlakoztatás nélkül is bekapcsolható legyen, azonban az ilyen állományrendszerek ACL nélküli csatlakoztatását nem ajánljuk senkinek, mivel ha egyszer már engedélyeztük a használatukat, majd kikapcsoljuk ezeket és végül a kiterjesztett tulajdonságok törlése nélkül újra engedélyezzük, akkor nagyon könnyen pórul járhatunk. Ha elkezdtük használni az ACL-eket egy állományrendszeren, akkor ne tiltsuk le ezeket, mert az így keletkező állományvédelem nem feltétlenül lesz kompatibilis a felhasználók által beállítottakkal, és az ACL újraengedélyezése a változásaik előtti korábbi ACL engedélyeket fogja visszaállítani az állományokra, aminek hatása kiszámíthatatlan.

A hozzáférés-vezérlési listákat használó állományrendszerek esetén egy + (plusz) jellel ábrázolják a kiterjesztett engedélyeket. Például:

```
drwx----- 2 robert robert 512 Dec 27 11:54 private
drwxrwx---+ 2 robert robert 512 Dec 23 10:57 könyvtár1
drwxrwx---+ 2 robert robert 512 Dec 22 10:20 könyvtár2
drwxrwx---+ 2 robert robert 512 Dec 27 11:57 könyvtár3
drwxr-xr-x 2 robert robert 512 Nov 10 11:54 public_html
```

Láthatjuk, hogy a `könyvtár1`, `könyvtár2` és `könyvtár3` könyvtárakhoz tartoznak ACL típusú engedélyek, míg a `public_html` könyvtárhoz nem.

14.12.1. Az ACL-ek használata

Az állományrendszerben található ACL engedélyeket a `getfacl(1)` segédprogrammal nézhetjük meg. Például a *próba* állomány ACL engedélyeit a következő paranccsal tudjuk megnézni:

```
% getfacl próba
#file:próba
#owner:1001
#group:1001
user::rw-
group::r--
other::r--
```

Egy állomány ACL engedélyeit a `setfacl(1)` segédprogrammal tudjuk megváltoztatni. Figyeljük meg:

```
% setfacl -k próba
```

A `-k` opció törli az összes ACL alapú engedélyt egy állományról vagy állományrendszeréről. Ennél viszont sokkal hasznosabb a `-b` opció használata, mivel az meghagyja az ACL működéséhez szükséges alapvető mezőket.

```
% setfacl -m u:trhodes:rw,group:web:r--,o:---- próba
```

Ebben a fenti parancsban a `-m` opciót pedig arra használtuk, hogy módosítsuk az alapértelmezett ACL bejegyzéseket. Mivel az ezt megelőző parancsban teljesen töröltük még az előredefiniált bejegyzéseket is, ez a parancs a megadott paraméterekkel kiegészítve ezeket vissza fogja állítani. Ügyeljünk arra, hogy ha olyan felhasználót vagy csoportot adunk meg, ami nem létezik a rendszerben, akkor a szabvány kimenetre egy `Invalid argument` hibaüzenetet kapunk.

14.13. A külső programok biztonsági problémáinak figyelése

Írta: Tom Rhodes.

Az utóbbi években a biztonsági kérdésekkel foglalkozó világban számos fejlesztésre került sor a sebezhetőségi figyelmeztetések feldolgozásában. Manapság tulajdonképpen bármilyen operációs rendszer fokozott veszélynek teszi ki magát a külső programok telepítésével és használatával.

A sebezhetőségekről beszámoló értesítések a biztonság egyik alapköve, azonban a FreeBSD projekt nem tud ilyen jelentéseket kiadni a FreeBSD alaprendszerén kívül minden egyes külső alkalmazáshoz. Azonban lehetőségünk van enyhíteni a külső csomagok sebezhetőségén és figyelmeztetni a rendszergazdákat az ismert biztonsági problémákra. A FreeBSD-nek van egy **Portaudit** nevű segédprogramja, amit kizárólag erre a célra hoztak létre.

A `ports-mgmt/portaudit` port egy adatbázist használ, ahol a FreeBSD biztonsági csapata és a portok fejlesztői tartják karban az ismert biztonsági problémákat.

A **Portaudit** használatának megkezdéséhez telepítsük a Portgyűjteményből:

```
# cd /usr/ports/ports-mgmt/portaudit && make install clean
```

A telepítési folyamat során a `periodic(8)` konfigurációs állományai is frissítődnek, így a **Portaudit** is lefut a napi biztonsági ellenőrzések folyamán. Gondoskodjunk róla, hogy a `root` felhasználónak levélben elküldött a napi biztonsági értesítéseket rendszeresen elolvassuk. Nincs szükségünk további beállításokra.

A telepítés után a rendszergazda a következő paranccsal tudja frissíteni a saját adatbázispéldányát és megnézni a pillanatnyilag telepített csomagok ismert sebezhetőségeit:

```
# portaudit -Fda
```

Megjegyzés: Ez az adatbázis a `periodic(8)` minden egy futásakor magától frissül, ezért ez a parancs lényegében elhagyható. Egyedül a soronkövetkező példákhoz kell kiadni.

A Portgyűjteményből telepített külső alkalmazások megbízhatóságának ellenőrzését az alábbi parancs kiadásával bármikor elvégezhetjük:

```
# portaudit -a
```

A **Portaudit** ennek hatására valahogy így fogja megjeleníteni a sebezhető csomagokat:

```
Affected package: cups-base-1.1.22.0_1
Type of problem: cups-base -- HPGL buffer overflow vulnerability.
Reference: <http://www.FreeBSD.org/ports/portaudit/40a3bca2-6809-11d9-a9e7-0001020eed82.html>
```

1 problem(s) in your installed packages found.

You are advised to update or deinstall the affected package(s) immediately.

Fordítása:

```
Érintett csomag: cups-base-1.1.22.0_1
A probléma jellege: cups-base -- HPGL puffer túlcsordulási sebezhetőség.
Link: <http://www.FreeBSD.org/ports/portaudit/40a3bca2-6809-11d9-a9e7-0001020eed82.html>
```

A telepített csomagokkal kapcsolatban 1 problémát találtam.

Javasoljuk, hogy az érintett csomagokat azonnal frissítse vagy távolítsa el.

Ha a böngészőnk az itt megadott címre irányítjuk, akkor megismerhetjük a kérdéses sebezhetőség pontosabb részleteit. Ezen az oldalon megtalálhatjuk a hiba által érintett verziókat a FreeBSD portok verziója szerint, illetve más olyan honlapokat, ahol biztonsági figyelmeztetéseket találhatunk.

Röviden összefoglalva, a **Portaudit** egy komoly segédeszköz és hitetlenül hasznos kiegészítője a **Portupgrade** portnak.

14.14. A FreeBSD biztonsági figyelmeztetései

Írta: Tom Rhodes.

A FreeBSD több más kereskedelmi minőségű operációs rendszerhez hasonlóan “Biztonsági figyelmeztetéseket” (Security Advisory) ad ki. Ezek a figyelmeztetések általában megjelennek a biztonsággal foglalkozó levelezési listákon és a hivatkozott hibák kijavítása után a megfelelő kiadások hibajegyzékében is. Ebben a szakaszban megismerjük és értelmezzük ezeket a figyelmeztetéseket, valamint megtudhatjuk, milyen lépéseket kell megtennünk a rendszerünk kijavításához.

14.14.1. Hogyan épül fel egy figyelmeztetés?

A FreeBSD biztonsági figyelmeztetései az alább látható formában jelennek meg, amit mi most a freebsd-security-notifications (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications>) levelezési listáról kölcsönöztünk.

```
=====
FreeBSD-SA-XX:XX.UTIL                                     Security Advisory
                                                         The FreeBSD Project

Topic:             denial of service due to some problem❶

Category:          core❷
Module:            sys❸
Announced:        2003-09-23❹
Credits:           Person@EMAIL-ADDRESS❺
```

Affects: All releases of FreeBSD^❶
 FreeBSD 4-STABLE prior to the correction date

Corrected: 2003-09-23 16:42:59 UTC (RELENG_4, 4.9-PRERELEASE)
 2003-09-23 20:08:42 UTC (RELENG_5_1, 5.1-RELEASE-p6)
 2003-09-23 20:07:06 UTC (RELENG_5_0, 5.0-RELEASE-p15)
 2003-09-23 16:44:58 UTC (RELENG_4_8, 4.8-RELEASE-p8)
 2003-09-23 16:47:34 UTC (RELENG_4_7, 4.7-RELEASE-p18)
 2003-09-23 16:49:46 UTC (RELENG_4_6, 4.6-RELEASE-p21)
 2003-09-23 16:51:24 UTC (RELENG_4_5, 4.5-RELEASE-p33)
 2003-09-23 16:52:45 UTC (RELENG_4_4, 4.4-RELEASE-p43)
 2003-09-23 16:54:39 UTC (RELENG_4_3, 4.3-RELEASE-p39)^❷

CVE Name: CVE-XXXX-XXXX^❸

For general information regarding FreeBSD Security Advisories, including descriptions of the fields above, security branches, and the following sections, please visit <http://www.FreeBSD.org/security/>.

I. Background^❹

II. Problem Description⁽¹⁰⁾

III. Impact⁽¹¹⁾

IV. Workaround⁽¹²⁾

V. Solution⁽¹³⁾

VI. Correction details⁽¹⁴⁾

VII. References⁽¹⁵⁾

- ❶ A `Topic` mezőben olvashatjuk pontosan mi is maga a probléma. Alapvetően bemutatja az érintett biztonsági figyelmeztetést és megemlíti a sebezhető segédprogramot.
- ❷ A `Category` mező hivatkozik a rendszer azon részére, amelyre a hiba kihatással lehet. Értéke lehet `core`, `contrib` vagy `ports`. A `core` kategória azt jelzi, hogy a sebezhetőség a FreeBSD legfontosabb komponenseit érinti. A `contrib` kategória a FreeBSD projekt számára felajánlott szoftverek, mint például a **sendmail** sebezhetőségére utal. Végezetül a `ports` kategória jelzi, hogy a sebezhetőség valamelyik, a Portgyűjteményben szereplő szoftverre érvényes.
- ❸ A `Module` mező a sebezhető komponens helyét nevezi meg, például `sys`. Ebben a példában azt láthatjuk, hogy a `sys` modul a hibás. Ezért a sebezhetőség egy rendszermagban használt komponenst érint.
- ❹ Az `Announced` mező a biztonsági figyelmeztetés kiadásának vagy széleskörű kihirdetésének dátumát rögzíti. Ez azt jelenti, hogy a biztonsági csapat meggyőződött a probléma létezéséről és a hibát orvosoló javítás már felkerült a FreeBSD forráskódjába.

- ⑤ A `Credits` mező azokat az egyéneket vagy szervezeteket említi meg, akik észlelték a sebezhetőséget és jelentették.
- ⑥ Az `Affects` mezőben megadják, hogy a FreeBSD melyik kiadásaira van hatással a sebezhetőség. Ha a rendszermag esetén lefuttatjuk az `ident` parancsot az érintett állományokra, akkor megtudhatjuk a pontos revíziójukat. A portoknál a verziószám a port neve után szerepel a `/var/db/pkg` könyvtárban. Ha a rendszerünket nem frissítettük CVS-ről és fordítottuk újra, akkor nagy a valószínűsége, hogy a sebezhetőség minket is érint.
- ⑦ A `Corrected` mező tartalmazza a a kijavítás dátumát, idejét, időzónáját és az ezt tartalmazó kiadást.
- ⑧ Az ismert sebezhetőségek adatbázisában (Common Vulnerabilities Database, CVD) használt azonosítási információk alapján végzett keresések számára fenntartott.
- ⑨ A `Background` mező adja meg részleteiben a sebezhető programmal kapcsolatos tudnivalókat. Az esetek többségében itt írják le, hogy miért jött létre az adott eszköz a FreeBSD-ben, mire használják és hogyan keletkezett.
- (10) A `Problem Description` mező a biztonsági rést részletezi. Ebben a részben szerepelhet a hibás kódrészlet vagy akár még az is, hogy miként kell vele előidézni a hibát.
- (11) Az `Impact` mező a probléma lehetséges hatásait írja körül a rendszerben. Ez például lehet egy DoS támadás, speciális engedélyek ellopása vagy akár a rendszeradminisztrátori jogok megszerzése.
- (12) A `Workaround` mező igyekszik elfogadható megoldást nyújtani a rendszerük frissítésére képtelen rendszergazdák számára. Ennek oka lehet az idő rövideje, a hálózati elérhetőség vagy más okokból fakadó elcsúszás. Ennek ellenére a biztonsági kérdéseket sosem szabad félvállról venni, ezért a sebezhető rendszereket vagy ki kell javítani vagy valamilyen módon meg kell kerülni a biztonsági rés kialakulását.
- (13) A `Solution` mező utasításokkal segít a rendszer kijavítását. Ez egy lépésről lépésre tesztelt és ellenőrzött módszer, amellyel a rendszerünket megfelelően ki tudjuk javítani és biztonságossá tenni.
- (14) A `Correction Details` mező mutatja a CVS-ág vagy kiadás nevét, amelyben a pontokat aláhúzásra cserélték. Ezenkívül még az egyes ágakban az érintett állományok revízióját is mutatja.
- (15) A `References` mező általában a témával kapcsolatos további forrásokat kínálja fel URL, könyv, levelezési lista vagy hírcsoport formájában.

14.15. A futó programok nyilvántartása

Írta: Tom Rhodes.

A futó programok nyilvántartása olyan biztonsági módszer, ahol a rendszergazda figyelemmel kíséri a rendszer használatban levő erőforrásait, a felhasználók közti megoszlását, gondoskodik a rendszer felügyeletéről és valamennyire nyomon követi a felhasználók parancsait.

Ennek a módszernek egyaránt megvannak a maga előnyei és hátrányai. Az egyik előnye, hogy a használatával a behatolás egészen a betörés pontjáig visszakövethető. Hátránya viszont, hogy a futó programok nyilvántartása rengeteg mennyiségű naplót generál és ehhez sok lemezterületre lesz szükségünk. Ebben a szakaszban végigjárjuk a programok nyilvántartásának alapjait.

14.15.1. A futó programok nyilvántartásának engedélyezése és használata

A futó programok nyilvántartását először engedélyeznünk kell. Ehhez a következő parancsokat kell kiadnunk:

```
# touch /var/account/acct

# accton /var/account/acct

# echo 'accounting_enable="YES"' >> /etc/rc.conf
```

Miután aktiváltuk, a nyilvántartást elkezdi számbavenni a processzor kihasználtságát, a parancsokat stb. A nyilvántartás emberek számára nem olvasható formátumban készül, ezért csak az `sa(8)` segédprogrammal tudjuk megnézni. Ha nem adunk meg neki semmilyen opciót, akkor az `sa` kilistázza a felhasználónkénti hívásokat, az összes eltelt időt percben, a teljes processzor- és felhasználói időt percben, az I/O műveletek átlagos számát stb.

A kiadott parancsokról a `lastcomm(1)` programmal tudunk tájékozódni. A `lastcomm` segítségével ki tudjuk írni a felhasználók adott terminálon kiadott parancsait is, mint például:

```
# lastcomm ls
trhodes tty1
```

Ezzel megjelenik a `trhodes` nevű felhasználó `tty1` terminálon kiadott összes ismert `ls` parancsa.

Számos hasznos beállítást és hozzájuk tartozó leírást találhatunk még a `lastcomm(1)`, `acct(5)` és `sa(8)` man oldalakon.

Megjegyzések

1. FreeBSD alatt a bejelentkezéshez használt szabványos jelszavak akár 128 karakteresek is lehetnek.

Fejezet 15. A jail alrendszer

Írta: Matteo Riondato.

15.1. Áttekintés

Ez a fejezet a FreeBSD-ben található jail alrendszert, valamint annak használatát mutatja be közelebbről. Az jail, melyet gyakran csak úgy emlegetnek, mint a *chroot környezetek* továbbfejlesztését, a rendszergazdák számára ajánlott, nagyon sokoldalú eszköz, de a haladó felhasználók is hasznosnak találhatják.

A fejezet elolvasása során megismerjük:

- mi is az a jail, milyen célra használható a FreeBSD-ben;
- hogyan hozzunk létre, indítsunk el és állítsunk le jaileket;
- a létrehozott jailek karbantartásainak alapjait, a jailek belülről és kívülről egyaránt.

A jail alrendszerrel még több hasznos információt a következő helyekről tudhatunk meg:

- A jail(8) man oldal. Ez tartalmazza a `jail` segédprogram teljes referenciáját — ez az a karbantartásra használható eszköz, amellyel el tudjuk indítani, le tudjuk állítani és vezérelni tudjuk a jaileket a FreeBSD-ben.
- A levelezési listák és azok archívumai. A FreeBSD general questions levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>) archívuma és a FreeBSD lista szerver (<http://lists.FreeBSD.org/mailman/listinfo>)-en található többi levelezési lista rengeteg olvasnivalót tartogat a jailekkel kapcsolatban. Mindig érdemes keresni ezekben az archívumokban, vagy beküldeni a kérdésünket a `freebsd-questions` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>) levelezési listára.

15.2. A jail alrendszerhez kapcsolódó fogalmak

A fejezet további részében a következő fogalmakat fogjuk használni, hogy a FreeBSD jailekhez tartozó egyes részeit és azok belső működését, valamint kapcsolatukat a rendszer többi részével még inkább érthetővé tegyük:

`chroot(8)` (parancs)

Egy segédprogram, amely a FreeBSD `chroot(2)` rendszerhívásán keresztül egy program és annak leszármazottjainak futtatásához megváltoztatja a rendszer gyökérkönyvtárát (`change root`).

`chroot(2)` (környezet)

A “chroot” módban futó programok környezete. Olyan erőforrásokat foglal magában, mint mondjuk az állományrendszer látható része, az elérhető felhasználói és csoport azonosítók, hálózati csatlók és egyéb folyamatok közti kommunikációs mechanizmusok stb.

`jail(8)` (parancs)

Az a rendszerkarbantartó segédprogram, amely lehetővé teszi program elindítását elzárt környezetben.

befogadó (rendszer, program, felhasználó stb.)

Az elzárt környezetet irányító rendszer. A befogadó rendszer hozzá tud férni az összes elérhető hardveres erőforráshoz, képes az elzárt környezeten kívül és belül futó programokat vezérelni. Az egyik legfontosabb különbség a befogadó és az elzárt rendszer között, hogy azok a korlátozások, amelyek az elzárt környezetben rendszeradminisztrátori jogokkal futó programokra vonatkoznak, nem feltétlenül érvényesek a befogadó rendszerben futóakra.

befogadott (rendszer, program, felhasználó stb.)

Olyan program, felhasználó vagy más egyéb egyed, amely csak egy jailen keresztül, korlátozottan tud hozzáférni az erőforrásokhoz.

15.3. Bevezetés

Mivel a rendszeradminisztráció egy nehéz és zavarba ejtő feladat, rengeteg komoly eszköz jött létre a rendszergazdák életének megkönnyítésére. Ezek az eszközök többnyire a rendszerek telepítését, beállítását és karbantartását igyekeznek valamilyen módon jobbá tenni. A rendszergazdák egyik feladata úgy gondoskodni a biztonságról, hogy közben a rendszer képes legyen ellátni eredeti feladatát.

A FreeBSD rendszerek biztonságosságának növelését hivatott egyik ilyen eszköz a *jails*. Először a FreeBSD 4.X verziójában bukkant fel, de jelentős fejlődésen ment keresztül a FreeBSD 5.X verziókban, aminek köszönhetően sokkal erőteljesebb és rugalmasabb alrendszerre vált. A fejlesztése természetesen most is folytatódik tovább, állandóan fejlődik a használhatósága, teljesítménye, megbízhatósága és biztonságossága.

15.3.1. Mi is az a jail?

A BSD-szerű operációs rendszerekben már a 4.2BSD óta megtalálható volt a chroot(2). A chroot(8) segédprogrammal meg tudjuk megváltoztatni adott programok számára a gyökérkönyvtárat, és ezzel egy biztonságos környezetet teremteni, távol a rendszer többi részétől. A chroot-tal kialakított környezetben elinduló programok nem tudnak hozzáférni a rajta kívül található állományokhoz és erőforrásokhoz. Ennek okán, ha egy ilyen környezetben futó szolgáltatást megtámadnak, az önmagában még nem teszi lehetővé a támadó számára, hogy elérhesse az egész rendszert. A chroot(8) remekül használható olyan egyszerűbb feladatok megoldására, amelyek nem igényelnek túlságosan sok rugalmasságot vagy bonyolult és fejlett támogatást. A chroot ötletének felmerülése óta azonban számos kiskaput találtak már az általa létrehozott környezetekben, és habár ezek mindegyikét javították a FreeBSD újabb változataiban, teljesen egyértelművé vált, hogy a chroot(2) nem biztosít járható utat a szolgáltatások biztonságossá tételéhez. Erre a feladatra egy új alrendszert kellett kiépíteni.

Ez az egyik oka annak, amiért az *jaileket* kifejlesztették.

A jailek által képviselt elzárás ötlete több szempontból is a hagyományos chroot(2) környezet elvén alapszik. Egy hagyományos chroot(2) környezetben futó programok korlátozása csupán abban merül ki, hogy az állományrendszer melyik részét láthatják. A rendszer többi erőforrása (mint mondjuk a felhasználók, futó programok vagy a hálózati alrendszer) azonban továbbra is megosztva marad a chroot környezetben és a befogadó rendszerben futó programok között. A jailek által alkalmazott megoldás kibővíti ezt a modellt, és nem csak az állományrendszerre vonatkozó hozzáférést virtualizálja, hanem több más dolog mellett kiterjeszti ezt a felhasználókra és a FreeBSD hálózati alrendszerére is. Az elzárt környezetek beállításaihoz elérhető finomhangolási lehetőségekről bővebben a 15.5 Szakaszban esik szó.

A jaileket az alább négy elem írja le:

- A könyvtárszerkezet egy részfája — attól a résztől indulva, ahonnan a jail kezdődik. A jailen belül futó programok nem léphetnek ki ebből a részfából. Az eredeti chroot(2) kialakításában merengő biztonsági hibák lehetőségei nem veszélyeztetik a többi FreeBSD jailt.
- A rendszer neve — a név, amelyet a jailen belül használunk. Mivel a jaileket elsősorban hálózati szolgáltatások kordában tartására használjuk, a jailekhez tartozó beszédes rendszernevek sokat tudnak segíteni a rendszergazdák munkájában.
- Egy IP-cím — a jailhez tartozik és nem változtatható meg a működése során. Egy jail IP-címe általában egy már létező hálózati csatoló másik címe, de ez nem szükségszerűen igaz minden esetben.
- Egy parancs — annak a programnak az elérési útja, amelyet elzártan kívánunk futtatni. Az elzárt környezet gyökerétől mérve relatívan adjuk meg, és az adott környezet típusától függően eltérő lehet.

Ezekről eltekintve a jailek rendelkezhetnek saját felhasználókkal és lehetnek saját `root` felhasználóik is.

Természetesen a `root` hatásköre csak az elzárt környezetre korlátozódik, és a befogadó rendszer szemszögéből az elzárt `root` nem mindenható. Ráadásul az elzárt `root` felhasználó nem hajthat végre semmilyen kritikus műveletet a saját jail(8) környezetén kívül. A `root` további képességeiről és korlátozásairól lentiekben bővebben is említést teszünk a 15.5 Szakaszban.

15.4. A jailek létrehozása és vezérlése

Egyes rendszergazdák a jaileket a következő két típusba sorolják: “teljes” jail, mely egy valódi FreeBSD rendszerre emlékeztet, és a “szolgáltatás” jail, mely egyetlen, feltehetően kiemelt jogokkal futó alkalmazás vagy szolgáltatás számára van előkészítve. Ez a besorolás csupán fogalmi szintű, a jail felépítésének módját nem befolyásolja. A jail(8) man oldal részletesen ismerteti a jailek létrehozását:

```
# setenv D /itt/lesz/a/jail
# mkdir -p $D ❶
# cd /usr/src
# make buildworld ❷
# make installworld DESTDIR=$D ❸
# make distribution DESTDIR=$D ❹
# mount -t devfs devfs $D/dev ❺
```

- ❶ Érdemes először a jail helyét megválasztani. Itt fog fizikailag helyet foglalni a befogadó rendszer állományrendszerén belül a jail. Jó választás lehet erre a `/usr/jail/jailnév`, ahol a `jailnév` a jailt azonosító rendszernév. A `/usr/` állományrendszeren általában elegendő hely jut a jail állományrendszerének, ami egy “teljes” jail esetén lényegében a FreeBSD alaprendszer alapértelmezett telepítésében megtalálható összes állomány másolatát tartalmazza.
- ❷ Ha korábban már a `make world` vagy a `make buildworld` parancs segítségével újrafordítottuk az alaprendszert, akkor ezt a lépést ki is hagyhatjuk és telepítsük az új alaprendszert közvetlenül az új jailbe.
- ❸ Ez a parancs fogja felmásolni a jail fizikai helyének választott könyvtár-részfába a működéshez szükséges programokat, függvénykönyvtárakat, man oldalakat és így tovább.
- ❹ A `make` paramétereként megadott `distribution` cél gondoskodik az összes szükséges konfigurációs állomány felmásolásáról. Magyarán szólva, átmásolja az összes telepíthető állományt a `/usr/src/etc/` könyvtárból a `jail/etc` alkönyvtárba, vagyis a `$D/etc/` könyvtárba.

- ⑤ A jaileken belül a devfs(8) csatlakoztatása nem kötelező. Másrészt azonban majdnem mindegyik alkalmazás, a feladatától függően, legalább egy eszközhöz hozzá akar férni. Nagyon fontos, hogy a kezünkbe vegyük a eszközök hozzáféréseinek irányítását a jaileken belül, mivel a helytelen beállítások révén a támadók csúnya dolgokat tudnak majd művelni. A devfs(8) működését a devfs(8) és devfs.conf(5) man oldalakon is ismertetett szabályrendszerek irányítják.

Ahogy a jailt telepítettük, a jail(8) segédprogrammal tudjuk elindítani. A jail(8) négy kötelező paramétert vár, melyekre a 15.3.1 Szakaszban ki is térünk. Más paramétereket is megadhatunk, például azt, hogy az elzárt program egy adott felhasználó jogaival fusson. A *command* paraméter használata a jail típusától függ: egy *virtuális rendszer* esetében a */etc/rc* jó választásnak bizonyulhat, mivel ennek segítségével egy valódi FreeBSD rendszerindítási folyamatát játszhatjuk le. Amennyiben elzárt *szolgáltatásról* van szó, az adott szolgáltatástól vagy alkalmazástól függ.

A jaileket gyakran már a rendszerindítás során elindítják, amit a FreeBSD *rc* mechanizmusa nagyban meg is könnyít.

1. A rendszer indítása során aktiválendő jailek listáját vegyük hozzá a *rc.conf(5)* állományhoz:

```
jail_enable="YES"      # Ide NO-t írjunk, ha ki akarjuk kapcsolni
jail_list="www"        # Szökőzzel elválasztva soroljuk fel a jaileket
```

Megjegyzés: A *jail_list* értékeként felsorolt jailek nevei csak betűket és számjegyeket tartalmazhatnak.

2. A *jail_list*-ben szereplő összes jailt meg kell adnunk az ezeket leíró *rc.conf(5)*-beli beállításokat:

```
jail_www_rootdir="/usr/jail/www"      # a jail gyökérkönyvtára
jail_www_hostname="www.example.org"   # a jail neve
jail_www_ip="192.168.0.10"            # a jail IP-címe
jail_www_devfs_enable="YES"           # legyen-e devfs a jailen belül
jail_www_devfs_ruleset="www_ruleset"  # az alkalmazott devfs szabályrendszer
```

Az *rc.conf(5)* állományban szereplő jailek esetén a */etc/rc* szkript fut le, tehát feltételezi, hogy az így megadott jail egy teljes virtuális rendszer. A szolgáltatások jailbe foglalásához meg kell változtatnunk a jail alapértelmezett parancsát is. Ezt a *jail_jailnév_exec_start* opció megfelelő beállításával tudjuk megtenni.

Megjegyzés: Az összes itt elérhető opciót a *rc.conf(5)* man oldalon találhatjuk meg.

Ha léteznek a megfelelő bejegyzések az *rc.conf* állományban, akkor az */etc/rc.d/jail* szkript is használható arra, hogy a jaileket kézzel indítsuk el vagy állítsuk le:

```
# /etc/rc.d/jail start www
# /etc/rc.d/jail stop  www
```

A jail(8) leállítására jelen pillanatban még nem érhető el szabályos módszer. Ez azért van, mert a szabályos rendszerleállítást elvégző parancsok nem használhatóak a jailen belül. Emiatt a jaileket a legtisztábban úgy tudjuk leállítani, ha kiadjuk az alábbi parancsot magában a jailben vagy pedig a *jexec(8)* segédprogrammal a jailen kívülről:

```
# sh /etc/rc.shutdown
```

Erről a témáról többet a jail(8) man oldalon olvashatunk.

15.5. Finomhangolás és karbantartás

Számos opció állítható be a jaileknél, és sokféle módon vegyíthetjük a befogadó FreeBSD rendszerünket a jailekkel, ami által magasabb szintű alkalmazásokat hozhatunk létre. Ebben a részben bemutatunk:

- Néhány olyan beállítást, amellyel finomhangolhatjuk a telepített jailek által megvalósított biztonsági megszorítások viselkedését.
- A jailek kezelésére alkalmas néhány olyan magasabb szintű alkalmazást, amelyek elérhetőek a FreeBSD Portgyűjteményén keresztül, és általános jail alapú megoldások kialakításához használhatóak.

15.5.1. A FreeBSD-ben található finomhangoló eszközök

A jailek beállításainak finomhangolását túlnyomórészt sysctl(8) változókkal végezhetjük el. A sysctl-en belül egy speciális részében találhatunk erre alkalmas beállításokat: ez a a FreeBSD rendszermag opciói között megtalálható `security.jail.*`. Itt közöljük a jailekre vonatkozó fontosabb sysctl változók listáját, az alapértelmezett értékeikkel együtt. A nevek minden bizonnyal sokat elárulnak, de ha többet szeretnénk tudni róluk, lapozzuk fel a jail(8) és sysctl(8) man oldalakat.

- `security.jail.set_hostname_allowed: 1`
- `security.jail.socket_unixiproute_only: 1`
- `security.jail.sysvipc_allowed: 0`
- `security.jail.enforce_statfs: 2`
- `security.jail.allow_raw_sockets: 0`
- `security.jail.chflags_allowed: 0`
- `security.jail.jailed: 0`

Ezekkel a változókkal a *befogadó rendszer* rendszergazdája tud hozzátenni vagy elvenni a `root` felhasználó alapértelmezett határaihoz. Vegyük azonban észre, hogy egyes korlátozások azonban semmiképpen sem szüntethetők meg. A `root` nem csatlakoztathat és választhat le állományrendszereket a jail(8) környezetben. Az elzárt `root` nem tölthet be és törölhet devfs(8) szabályrendszereket, tűzfal szabályokat sem, ill. nem végezhet semmilyen olyan bármilyen más karbantartási feladatot, amely a rendszermag adataiban módosítást vonna maga után, például nem állíthatja a rendszermag `securelevel` (biztonsági szintjének) értékét.

A FreeBSD alaprendszere tartalmazza azokat a segédeszközöket, amelyekkel a rendszerben aktív jailek információt tudjuk megjeleníteni, vagy csatlakozni tudunk hozzájuk. A `jls(8)` és `jexec(8)` parancsok részei az alap FreeBSD rendszernek, segítségükkel elvégezhetőek az alábbi egyszerű feladatokat:

- Ki tudjuk írni az aktív jailek és hozzájuk tartozó azonosítókat (JID-eket), IP-címeket, neveket és útvonalakat.
- A befogadó rendszerből hozzá tudunk csatlakozni egy futó jailhez, és parancsokat tudunk futtatni a jailen belül vagy karbantartási feladatokat tudunk elvégezni magán a jailen belül. Ez különösen hasznosnak bizonyulhat, amikor a `root` felhasználó szabályosan le akarja állítani a jailt. A `jexec(8)` segédprogrammal el tudunk indítani egy parancsértelmezőt a jailen belül, amiből aztán irányíthatjuk. Példa:

```
# jexec 1 tcsh
```

15.5.2. Magasszintű karbantartó eszközök a FreeBSD Portgyűjteményében

A sok külső karbantartó eszköz közül az egyik legteljesebb és leghasznosabb a `sysutils/jailutils`. Sok kisebb alkalmazást tartalmaz, melyek kibővítik a jail(8) irányíthatóságát. Bővebb információkért kérjük, látogassa meg a hozzá tartozó honlapot.

15.6. A jailek alkalmazása

15.6.1. Szolgáltatások jailbe foglalása

Írta: Daniel Gerzo.

Ez a rész eredetileg Simon L. B. Nielsen <http://simon.nitro.dk/service-jails.html> oldalon található írásán, valamint Ken Tom (<locals@gmail.com>) átdolgozott cikkén alapul. Itt megismerhetjük, hogyan állítsunk be a FreeBSD rendszerünkben egy biztonsági réteget a jail(8) felhasználásával. Továbbá feltételezzük, hogy ez a rendszer legalább RELENG_6_0 verziójú és a fejezetben korábban tárgyaltakat az olvasó teljes mértékben megértette.

15.6.1.1. A kialakítás

A jailek egyik legnagyobb gondja a frissítés folyamatának lebonyolítása. Azért jelent ez egyre inkább gondot, mert minden egyes jailt újra fel kell építenünk a frissítése során. Ez többnyire nem okoz gondot egyetlen jail használata során, mivel maga a frissítési folyamat meglehetősen egyszerű, azonban igen időigényessé és fárasztóvá tud válni több jail esetében.

Figyelem Ez a példa a FreeBSD képességeinek haladó szintű ismeretét követeli meg. Amennyiben az itt bemutatott lépések túlságosan is bonyolultnak tűnnének, érdemes olyan egyszerűbb rendszerek után nézni, mint mondjuk a `sysutils/ezjail`, amely egy egyszerűbb módszert kínál fel a FreeBSD-ben használt jailek karbantartására, és nem is annyira bonyolult, mint ez a példa.

A bemutatandó példa célja, hogy feloldja az ilyen jellegű problémákat, és ezért igyekszik a jailek között mindent megosztani, ami csak lehetséges. Mindezt biztonságosan éri el — írásvédett `mount_nullfs(8)` állományrendszer használatával, aminek köszönhetően a frissítés maga egyszerűbbé, az egyes szolgáltatások különzárása pedig vonzóbbá válik. Ráadásul egyúttal egy nagyon egyszerű módszert mutat az új jailek hozzáadására és a régi törlésére ugyanúgy, mint a frissítésükre.

Megjegyzés: Például ilyen szolgáltatásokat kívánunk szabályozni: egy HTTP szerver, egy DNS szerver, egy SMTP szerver és így tovább.

Az itt szereplő beállítás céljai:

- Készítsünk egy egyszerűen és könnyen átlátható jailkezelési rendszert. Ebből tehát következik, hogy *ne* kelljen lefuttatni a teljes rendszer telepítését minden egyes jailre.
- Könnyítsük meg az új jailek hozzáadását és a régiek eltávolítását.
- Könnyítsük meg a már létező jailek frissítését és cseréjét.
- Tegyük lehetővé saját FreeBSD ágak futtatását.
- Legyünk különösen körültekintőek a biztonság tekintetében, és igyekezzünk minél jobban csökkenteni veszély kockázatát.
- Takarékoskodjunk a tárhellyel és az állományrendszerrel, amennyire csak lehet.

Ahogy azt már korábban is említettük, ez a kialakítás nagyban építkezik egyetlen fő sablonra, amely írásvédetten kerül csatlakoztatásra (**nullfs**en keresztül) az egyes jailekben, valamint jailenként egy-egy írható-olvasható eszközre. Ez az eszköz lehet egy külön fizikai lemez, egy partíció vagy egy vnode alapú md(4) eszköz. Ebben a példában írható-olvasható **nullfs** csatlakozásokat használunk.

Az állományrendszer kiosztása a most következő listában szerepel:

- Minden jailt a `/home/j` könyvtárban csatlakoztatunk.
- A `/home/j/mroot` lesz az összes jail sablonja és mindegyikük számára írásvédett.
- Minden jailnek létrehozunk egy üres alkönyvtárat a `/home/j` könyvtárban.
- Minden jailnek lesz egy `/s` alkönyvtára, amelyet a rendszer írható-olvasható részére irányítunk.
- Minden jailnek lesz egy saját írható-olvasható része, amely a `/home/j/skel` könyvtáron alapszik.
- Mindegyik elzárt terület (a jailek írható-olvasható része) a `/home/js` könyvtárban jön létre.

Megjegyzés: Ez a kiosztás feltételezi, hogy a jaileket a `/home` partíción hozzuk létre. Ez természetesen bármi másra megváltoztatható, de akkor figyelniünk kell erre minden egyes parancs kiadása előtt.

15.6.1.2. A sablon létrehozása

Ez a rész leírja a fő sablon létrehozásához szükséges lépéseket. Ez a jailek számára írásvédett lesz.

Érdemes mindig frissíteni a FreeBSD rendszerünket a legújabb -RELEASE ágra. Ehhez olvassuk el az ide tartozó fejezetet (http://www.FreeBSD.org/doc/hu_HU.ISO8859-2/books/handbook/makeworld.html) a kézikönyvből. Abban az esetben, ha a frissítés nem lenne megoldható, egy `make buildworld` parancsot mindenképpen le kell tudnunk futtatni. Ezenfelül a `sysutils/cpdup` csomagra is szükségünk van. Használni fogjuk a `portsnap(8)` segédprogramot is a FreeBSD Portgyűjtemény letöltéséhez. Akik nem ismernék, a kézikönyv erről szóló fejezetében (http://www.FreeBSD.org/doc/hu_HU.ISO8859-2/books/handbook/portsnap.html) olvashatnak róla.

1. Először is, készítsük el az írásvédett állományrendszer könyvtárszerkezetét, amely majd tartalmazni fogja a jailek által használt FreeBSD-s programokat. Ezután lépünk be a FreeBSD forrásfájának könyvtárába és telepítsük fel az írásvédett állományrendszert a sablonba:

```
# mkdir /home/j /home/j/mroot
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot
```

2. Ezt követően készítsük elő a jailek számára a FreeBSD Portgyűjteményt és FreeBSD forrásfát, melyek kellenek a **mergemaster** használatához:

```
# cd /home/j/mroot
# mkdir usr/ports
# portsnap -p /home/j/mroot/usr/ports fetch extract
# cpdup /usr/src /home/j/mroot/usr/src
```

3. Hozzuk létre a rendszer írásvédett részének vázát:

```
# mkdir /home/j/skel /home/j/skel/home /home/j/skel/usr-X11R6 /home/j/skel/distfiles
# mv etc /home/j/skel
# mv usr/local /home/j/skel/usr-local
# mv tmp /home/j/skel
# mv var /home/j/skel
# mv root /home/j/skel
```

4. Használjuk a **mergemastert** a hiányzó konfigurációs állományok telepítésére. Szabaduljunk meg a **mergemaster** által készített felesleges könyvtáraktól:

```
# mergemaster -t /home/j/skel/var/tmp/temproot -D /home/j/skel -i
# cd /home/j/skel
# rm -R bin boot lib libexec mnt proc rescue sbin sys usr dev
```

5. Most pedig szimbolikusan linkeljük az írható-olvasható állományrendszert az írásvédett állományrendszerre. Ellenőrizzük, hogy a szimbolikus linkek a megfelelő `s/` könyvtárakban jöttek létre. Valós vagy rossz helyen létrehozott könyvtárak használata esetén a telepítés nem fog sikerülni.

```
# cd /home/j/mroot
# mkdir s
# ln -s s/etc etc
# ln -s s/home home
# ln -s s/root root
# ln -s ../s/usr-local usr/local
# ln -s ../s/usr-X11R6 usr/X11R6
# ln -s ../../s/distfiles usr/ports/distfiles
# ln -s s/tmp tmp
# ln -s s/var var
```

6. Utolsó lépésként hozzunk létre egy `/home/j/skel/etc/make.conf` állományt az alábbi tartalommal:

```
WRKDIRPREFIX?= /s/portbuild
```

A `WRKDIRPREFIX` beállításával lehetővé válik a FreeBSD portok jaileken belüli fordítása. Ne felejtsük el, hogy a portokat tartalmazó könyvtár az írásvédett rendszer része! Az átállított `WRKDIRPREFIX` azonban megengedi, hogy a fordítások az egyes jailek írható-olvasható részeiben történjenek.

15.6.1.3. A jailek létrehozása

Most, miután teljesen elkészült a FreeBSD jailek sablonja, be is tudjuk állítani és hozzá is tudjuk venni ezeket az `/etc/rc.conf` állományhoz. Ebben a példában 3 jail létrehozását láthatjuk: “NS”, “MAIL” és “WWW”.

1. Írjuk bele a következő sorokat az `/etc/fstab` állományba, aminek köszönhetően az egyes jailek számára elérhetővé válik az írásvédett sablon és a hozzájuk tartozó írható-olvasható területek:

```
/home/j/mroot /home/j/ns nullfs ro 0 0
```

```

/home/j/mroot    /home/j/mail    nullfs  ro  0  0
/home/j/mroot    /home/j/www     nullfs  ro  0  0
/home/js/ns      /home/j/ns/s    nullfs  rw  0  0
/home/js/mail    /home/j/mail/s  nullfs  rw  0  0
/home/js/www     /home/j/www/s   nullfs  rw  0  0

```

Megjegyzés: Az első helyen nullával jelölt partíciókat a `fsck(8)` nem fogja ellenőrizni a rendszer indulása során, a második helyen nullával jelölt partíciókat pedig nem fogja menteni a `dump(8)`. Mi egyáltalán nem akarjuk, hogy az **fsck** ellenőrizze vagy a **dump** lementse a jailjeinkhez tartozó írásvédett **nullfs**-partícióinkat. Ezért szerepel végig "0 0" a fentebb szereplő `fstab`-bejegyzések utolsó két oszlopában.

2. Állítsuk be a jaileket az `/etc/rc.conf`-ban:

```

jail_enable="YES"
jail_set_hostname_allow="NO"
jail_list="ns mail www"
jail_ns_hostname="ns.example.org"
jail_ns_ip="192.168.3.17"
jail_ns_rootdir="/usr/home/j/ns"
jail_ns_devfs_enable="YES"
jail_mail_hostname="mail.example.org"
jail_mail_ip="192.168.3.18"
jail_mail_rootdir="/usr/home/j/mail"
jail_mail_devfs_enable="YES"
jail_www_hostname="www.example.org"
jail_www_ip="62.123.43.14"
jail_www_rootdir="/usr/home/j/www"
jail_www_devfs_enable="YES"

```

FigyelemAzért állítottuk a `jail_név_rootdir` változó értékét a `/usr/home` könyvtárra a `/home` könyvtár helyett, mert a FreeBSD alaptelepítésében a `/home` könyvtár fizikailag a `/usr/home` könyvtárral egyezik meg. A `jail_név_rootdir` változó értékeként megadott könyvtár *nem* tartalmazhat szimbolikus linket, máskülönben a jailek nem lesznek hajlandóak létrejönni. Ennek megállapításában a `realpath(1)` segédprogram lehet segítségünkre. A korlátozás részleteiről a FreeBSD-SA-07:01.jail biztonsági figyelmeztetésben olvashatunk.

3. Hozzuk létre az egyes jailek írásvédett állományrendszereihez szükséges csatlakozási pontokat:

```
# mkdir /home/j/ns /home/j/mail /home/j/www
```

4. Telepítsük az írható-olvasható sablont az egyes jailekbe. Figyeljük meg a `sysutils/cpdup` használatát, amellyel az egyes könyvtárak pontos másolatait hozhatjuk létre:

```

# mkdir /home/js
# cpdup /home/j/skel /home/js/ns
# cpdup /home/j/skel /home/js/mail
# cpdup /home/j/skel /home/js/www

```

5. Ebben a fázisban a jailek már elkészültek és készen állnak a futásra. Először csatlakoztassuk az egyes jailekhez szükséges állományrendszereket, majd indítsuk el ezeket a `/etc/rc.d/jail` szkripttel:

```
# mount -a
```

```
# /etc/rc.d/jail start
```

A jailek most már futnak. Az elindulásuk ellenőrzéséhez használjuk a `jls(8)` parancsot. Valami ilyesmit láthatunk a kiadása után:

```
# jls
  JID  IP Address      Hostname                Path
    3   192.168.3.17   ns.example.org          /home/j/ns
    2   192.168.3.18   mail.example.org        /home/j/mail
    1   62.123.43.14    www.example.org         /home/j/www
```

Itt már be tudunk jelentkezni az egyes jailekbe, új felhasználókat tudunk készíteni vagy démonokat tudunk beállítani. A `JID` oszlop mutatja az egyes jailek azonosítási számát. A 3-as `JID` számú jailben az alábbi parancs használatával karbantartási feladatokat elvégezni:

```
# jexec 3 tcsh
```

15.6.1.4. Frissítés

Időről időre adódhat, hogy frissítenünk kell a rendszert a FreeBSD egy újabb változatára, vagy egy biztonsági hiba javítása miatt, vagy pedig a már meglevő jailek számára hasznos újítások bevezetése miatt. Ez a kialakítás megkönnyíti a korábban létrehozott jailjeink frissítését. Továbbá igyekszik minimalizálni a kiesésüket is, mivel a jaileket csak a legutolsó pillanatban fogjuk leállítani. Sőt, még az is lehetővé válik, hogy visszaállítsuk a korábbi verziót, ha véletlenül valami rosszul sülné el menetközben.

1. Első lépésként frissítsük magát a befogadó rendszert a megszokott módon. Ezután hozzunk létre egy új írásvédett sablont a `/home/j/mroot2` könyvtárban.

```
# mkdir /home/j/mroot2
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot2
# cd /home/j/mroot2
# cpdup /usr/src usr/src
# mkdir s
```

A `installworld` lefuttatása létrehoz néhány felesleges könyvtárat, melyeket takarítsunk is el:

```
# chflags -R 0 var
# rm -R etc var root usr/local tmp
```

2. Hozzuk újra létre az írható-olvasható szimbolikus linkjeinket a fő állományrendszerre:

```
# ln -s s/etc etc
# ln -s s/root root
# ln -s s/home home
# ln -s ../s/usr-local usr/local
# ln -s ../s/usr-X11R6 usr/X11R6
# ln -s s/tmp tmp
# ln -s s/var var
```

3. Most érkezett el az idő, hogy leállítsuk a jaileket:

```
# /etc/rc.d/jail stop
```

4. Válasszuk le az eredeti állományrendszereket:

```
# umount /home/j/ns/s
# umount /home/j/ns
# umount /home/j/mail/s
# umount /home/j/mail
# umount /home/j/www/s
# umount /home/j/www
```

Megjegyzés: Az írható-olvasható állományrendszerek hozzá vannak kapcsolva az írásvédett állományrendszerhez (/s), ezért azokat először le kell választani.

5. Mozgassuk el az útból a régi írásvédett állományrendszerünket és váltsuk fel az újjal. Így biztonsági mentésként és a régi írásvédett rendszer archívumaként továbbra is rendelkezésre áll, ha valami baj történne. Az itt használt elnevezés az újonnan létrehozott írásvédett állományrendszer dátumából ered. Mozgassuk át az eredeti FreeBSD Portgyűjteményt az új állományrendszerre, hogy megtakarítsunk némi tárhelyet és állományleíró:

```
# cd /home/j
# mv mroot mroot.20060601
# mv mroot2 mroot
# mv mroot.20060601/usr/ports mroot/usr
```

6. Most már készen áll az új írásvédett sablon, így már csak az állományrendszerek újracsatlakoztatása és a jailek újraindítása maradt:

```
# mount -a
# /etc/rc.d/jail start
```

A jls(8) használatával ellenőrizzük, hogy a jailek rendesen elindultak. Ne felejtjük el jailenként lefuttatni a mergemastert sem. A konfigurációs állományokat és az rc.d szkripteket is frissítenünk kell majd.

Fejezet 16. Kötelező hozzáférés-vezérlés (MAC)

Írta: Tom Rhodes.

16.1. Áttekintés

A FreeBSD 5.X változata új biztonsági bővítéseket vett át a TrustedBSD projektből a POSIX.1e nyomán. A két legjelentősebb új biztonsági mechanizmus az állományrendszerekben megtalálható hozzáférés-vezérlési listák (Access Control List, ACL) és a kötelező hozzáférés-vezérlés (Mandatory Access Control, MAC). A kötelező hozzáférés-vezérlés segítségével olyan új hozzáférés-vezérlési modulok tölthetők be, amelyek új biztonsági házirendeket implementálnak. Némelyek közülük védelmet nyújtanak a rendszer egy szűk részének, amivel így egy adott szolgáltatást bátyáznak alá. Mások minden részletre kiterjedő címkézett biztonságot szolgáltatnak alanyokon és objektumokon keresztül. A meghatározás "kötelező" része onnan fakad, hogy a szabályok betartatását a rendszergazdák és a rendszer végzik, és nem bízzák a felhasználókra, ahogy azt a System V típusú rendszerekben a szabványos állományokra és IPC-re érvényes engedélyeken keresztül a tetszés szerinti hozzáférés-vezérlés (Discretionary Access Control, DAC) teszi.

Ebben a fejezetben a kötelező hozzáférés-vezérlést övező keretrendszerre (MAC Framework) és a különböző biztonsági házirendeket megvalósító, beilleszthető modulokra fogunk összpontosítani.

A fejezet elolvasása során megismerjük:

- hogy a FreeBSD jelen pillanatban milyen modulokat tartalmaz a MAC rendszeren belül és milyen mechanizmusok tartoznak hozzájuk;
- hogy a MAC biztonsági házirendjeit képező modulok miket valósítanak meg, valamint mi a különbség a címkézett és címkézetlen házirendek között;
- hogyan kell hatékonyan beállítani és használni rendszerünkben a MAC rendszert;
- hogyan állítsuk be a MAC rendszerben található különféle biztonsági házirendeket képező modulokat;
- hogyan hozzunk létre a MAC rendszer segítségével egy biztonságosabb környezetet, amire példákat is mutatunk;
- hogyan teszteljük le a MAC rendszer beállításait és bizonyosodjunk meg működésének helyességéről.

A fejezet elolvasásához ajánlott:

- a UNIX és a FreeBSD alapjainak ismerete (3 fejezet)
- a rendszermag beállításának és lefordításának ismerete (8 fejezet)
- tisztában lenni az alapvető biztonsági kérdésekkel és azok hatásával a FreeBSD-n belül (14 fejezet)

Figyelem Az itt ismertetésre kerülő információk helytelen alkalmazása a rendszer hozzáférhetőségének teljes elvesztését, a felhasználók bosszantását vagy az X11 által felkínált lehetőségek kirekesztését eredményezheti. Ami viszont ennél is fontosabb, hogy a MAC rendszerre nem úgy kell tekinteni, mint amitől a rendszerünk tökéletesen biztonságossá válik. A MAC segítségével csupán a meglévő biztonsági házirendeket gyarapítjuk. A szilárd biztonsági rutin és a rendszeres ellenőrzések elvégzése nélkül a rendszerünk valójában sosem lesz teljesen biztonságos.

Hozzá kell tennünk, hogy a fejezetben bemutatott példák tényleg csak példák. Senkinek sem tanácsoljuk, hogy az itt említett beállításokat egy éles rendszerre is kiterjessze. A különböző biztonsági modulok felépítése

rengeteg gondolkodást és próbálgatást igényel. Aki nem érti meg az egész működését, könnyen azon kaphatja magát, hogy újra végig kell mennie a rendszeren és egyenként be kell állítania minden könyvtárat és állományt.

16.1.1. Amivel itt nem foglalkozunk

Ebben a fejezetben a MAC rendszerrel kapcsolatban rengeteg biztonsági kérdéssel foglalkozni fogunk. Az új MAC biztonsági modulok kifejlesztését azonban már nem érintjük. Számos olyan biztonsági modul található a MAC rendszerben, amelyek rendelkeznek az új modulok kialakításához és teszteléséhez szükséges jellemzőkkel. Ilyenek többek közt a `mac_test(4)`, `mac_stub(4)` és a `mac_none(4)`. Ezekről a biztonsági modulokról és az általuk szolgáltatott mechanizmusokról a man oldalaik tudnak bővebb tájékoztatást adni.

16.2. A fejezet fontosabb fogalmai

A fejezet tartalmának kifejtéséhez szükségünk lesz néhány fontosabb alapfogalom tisztázására. Segítségükkel vélhetően sikerül elosztatni a téma feldolgozása során felmerülő félreértéseket, illetve elkerülni az új fogalmak és információk váratlan felbukkanását.

- *alany*: Alanynak tekintünk a rendszerben minden olyan aktív egyedet, amely információt áramoltat az *objektumok*, tehát a felhasználók, a processzorok, a rendszerben futó programok stb. között. A FreeBSD-ben majdnem minden esetben a felhasználók egy szálon keresztül vezérlik a futó programokat.
- *címke*: A címke egy olyan biztonsági tulajdonság, ami vonatkozhat állományokra, könyvtárakra vagy a rendszer más elemeire. Egy címke tekinthető a bizalmasságot jelző pecsétnek is: ha egy állományra címkét teszünk, akkor benne megadjuk a rá vonatkozó biztonsági jellemzőket, és csak a hozzá hasonló biztonsági beállításokkal rendelkező állományok, felhasználók, erőforrások stb. érhetik el. A címkék jelentését és értelmezését a házirendek beállítása határozza meg: míg egyes házirendek a címkéket egy objektum sértetlenségének vagy titkosságának tekintik, addig mások a hozzáféréssel kapcsolatos szabályokat rögzítik bennük.
- *egycímkés*: Egycímkés esetről akkor beszélünk, amikor az adat áramlásának szabályozására az egész állományrendszer egyetlen címkét alkalmaz. Ha ezt beállítjuk egy állományrendszerrel, de nem adjuk meg vele együtt a `multilabel` opciót, akkor az összes állományra ugyanaz a címke érvényes.
- *erős vízjel*: Az erős vízjel házirendje szerint a biztonsági szint akkor növelhető, ha magasabb szintű információkhoz akarunk hozzájutni. A legtöbb esetben a folyamatok befejeződése után visszaállítódik az eredeti szint. A FreeBSD MAC rendszere pillanatnyilag ehhez nem tartalmaz házirendet, de a teljesség kedvéért megadtuk ennek a definícióját is.
- *gyenge vízjel*: A gyenge vízjel házirendje szerint a biztonsági szint csökkenthető az alacsonyabb szintű információk elérése érdekében. A legtöbb esetben a folyamatok befejeződése után visszaállítódik az eredeti szint. A FreeBSD-ben ezt a házirendet egyedül a `mac_lomac(4)` alkalmazza.
- *házirend*: Szabályok olyan gyűjteménye, amely megadja, hogy miként kell a célokat teljesíteni. Egy *házirend* általában az egyes elemek kezelését rögzíti. Ebben a fejezetben a *házirend* kifejezés alatt a *biztonsági házirendet* értjük, tehát olyan szabályok gyűjteményét, amelyek az adatok és az információ áramlását határozzák meg, továbbá megadják, hogy közülük ki mihez férhet hozzá.

- *kényesség*: Általában az MLS tárgyalásakor kerül elő. Az kényesség szintjével az adatok fontosságát vagy titkosságát szokták jelölni. A kényességi szint növekedésével növekszik az adat titkosságának vagy bizalmasságának szintje.
- *objektum*: Objektum vagy rendszerobjektum minden olyan egyed, amelyen információ folyik keresztül az *alanyok* irányításával. Ezek lehetnek többek közt könyvtárak, állományok, mezők, képernyők, billentyűzetek, a memória, mágneses tárolóeszközök, nyomtatók vagy bármilyen más adattároló/hordozó eszköz. Az objektumok alapvetően adattárolók vagy a rendszer erőforrásai. Egy *objektum* elérésén gyakorlatilag az adatok elérését értjük.
- *rekesz*: Egy rekeszbe soroljuk az elrekeszteni vagy elkülöníteni kívánt programok és adatok összességét, ahol a felhasználók explicit módon képesek hozzáférni a rendszer bizonyos komponenseihez. Emellett a rekesz utalhat egy tetszőleges csoportosításra is, például munkacsoportra, osztályra, projektre vagy témára. A rekeszek használata elengedhetetlen a biztonsági házirendek kialakításához.
- *sértetlenség*: A sértetlenség, mint kulcsfogalom, az adatok megbízhatóságának szintje. Minél sértetlenebb az adat, annál inkább tekinthetjük megbízhatónak.
- *szint*: Egy biztonsági tulajdonság megnövelt vagy lecsökkentett beállítása. A szint növekedésével együtt a biztonság mértéke is növekszik.
- *többcímkezés*: A `multilabel` vagyis többcímkezés jellemző az állományrendszerek esetén fordulhat elő, és a `tunefs(8)` segédprogrammal állítható be egyfelhasználós módban vagy a rendszer indítása során az `fstab(5)` állományon keresztül, esetleg egy új állományrendszer létrehozásakor. Ezzel a beállítással a rendszergazda különféle MAC címkeket rendelhet különböző objektumokhoz. Ez a beállítás természetesen csak olyan biztonsági modulok esetén él, amelyek tudnak címkézni.

16.3. A MAC ismertetése

Az imént definiált új fogalmak tükrében most nézzük meg, hogy a MAC rendszer alkalmazásával miként javíthatunk rendszerünk biztonságán. A MAC rendszerhez készített különböző biztonsági modulok alkalmasak a hálózat és az állományrendszerek védelmére, valamint segítségükkel megakadályozhatjuk, hogy a felhasználók elérhessenek bizonyos portokat és socketeket stb. A házirendeket formázó modulokat talán együttesen tudjuk a leghatékonyabban alkalmazni, és ha egyszerre több modul betöltésével egy többretegű védelmi rendszert alakítunk ki. Ez nem ugyanaz, mint a rendszer megerősítése, ahol a rendszer összetevőit jellemző módon csak bizonyos célok tekintetében edzzük meg. A módszer egyedüli hátulütője a többszörös állományrendszeri címkékkel, a felhasználónként beállítandó hálózati eléréssel stb. járó adminisztrációs költségek.

Ezek a hátrányok azonban eltörpülnek a létrehozott rendszer tartósságával szemben. Például, ha képesek vagyunk megmondani, hogy az adott konfigurációban milyen házirendek alkalmazására van szükség, akkor ezzel az adminisztrációs költségek visszaszoríthatóak. A szükségtelen házirendek eltávolításával még növelhetjük is a rendszer összeteljesítményét, valamint az így felkínált rugalmasságot. Egy jó kialakításban figyelembe kell venni az összes biztonsági előírást, és hatékonyan megvalósítani ezeket a rendszer által felajánlott különféle biztonsági modulokkal.

Ezért tehát a MAC lehetőségeit kihasználó rendszerekben legalább annyit meg kell tudni oldani, hogy a felhasználók ne változtathassák kedvükre a biztonsági tulajdonságokat. Az összes felhasználói segédprogramnak, programnak és szkriptnek a kiválasztott biztonsági modulokban szereplő hozzáférési szabályokkal kifizített kereten belül kell mozognia. A MAC totális irányítása pedig a rendszergazda kezében van.

A rendszergazda így egyedül csak a megfelelő biztonsági modulok gondos összeválogatásáért felelős. Bizonyos környezetekben szükséges lehet a hálózaton keresztüli hozzáférések korlátozása is. Ilyen esetekben a `mac_portacl(4)`, `mac_ifoff(4)` vagy a `mac_biba(4)` moduloktól érdemes elindulnunk. Más esetekben az

állományrendszerek objektumainak bizalmasságát kell csupán megőriznünk. Erre a célra a `mac_bsdextended(4)` és `mac_mls(4)` modulok a legalkalmasabbak.

A házirendekhez kapcsolódó döntések a hálózati beállítások alapján is meghozhatóak. Elképzelhető, hogy csak bizonyos felhasználók férhetnek hozzá az `ssh(1)` szolgáltatásain keresztül a hálózathoz vagy az internethez. A `mac_portacl(4)` pontosan ilyen helyzetekben tud a segítségünkre sietni. Mit tegyünk viszont az állományrendszerek esetén? Vágjunk el adott felhasználókat vagy csoportokat bizonyos könyvtáraktól? Vagy korlátozzuk a felhasználók vagy segédprogramok hozzáférését adott állományokhoz bizonyos objektumok bizalmassá tételével?

Az állományrendszerek esetében az objektumokat néhány felhasználó elérheti, mások pedig nem. Például egy nagyobb fejlesztőcsapat kisebb csoportokra bontható. Az A projektben résztvevő fejlesztők nem férhetnek hozzá a B projektben dolgozó fejlesztők munkájához. Ellenben szükségük lehet a C projekten munkálkodó fejlesztők által létrehozott objektumokra. Ez egy igen érdekes helyzet. A MAC rendszer által felkínált különböző biztonsági modulokra építkezve azonban könnyedén csoportokba tudjuk szervezni a felhasználókat, és a megfelelő területekhez az információ kiszivárgása nélkül hozzá tudjuk őket engedni.

Ennek következtében minden egyes biztonsági modul a maga módján gondoskodik az egész rendszer biztonságáról. A céljainknak megfelelő modulokat egy jól átgondolt biztonsági házirend alapján válasszuk ki. Sok esetben az egész házirendet át kell tekinteni és újra kell alkalmazni a rendszerben. A MAC által felajánlott különböző biztonsági modulok megértése segít a rendszergazdáknak megválasztani az adott helyzetben legjobban alkalmazható házirendeket.

A FreeBSD rendszermagja alpból nem tartalmazza a MAC rendszert. Ezért a fejezetben szereplő példák vagy az itt leírtak kipróbálásához az alábbi beállítást kell hozzátennünk a rendszermag beállításait tartalmazó állományhoz:

```
options MAC
```

Majd fordítsuk és telepítsük újra a rendszermagot.

Figyelem Miközben a MAC rendszerhez készült különböző modulok a saját man oldalaik szerint igénylik a beépítésüket, vigyázzunk velük, mert ezzel a rendszerüket pillanatok alatt ki tudjuk zárni a hálózathoz és így tovább. A MAC alapú védelem felépítése leginkább egy tűzfal összeállításához hasonlítható, ahol ugyanígy számolni kell azzal, hogy egy óvatlan paranccsal kizárhatjuk magunkat a rendszerből. Valamilyen módon mindig próbáljunk gondoskodni a rendszer előző állapotának visszaállíthatóságáról, és a MAC távoli adminisztrációját mindig nagyfokú körültekintéssel végezzük.

16.4. Bővebben a MAC címkéiről

A MAC-címke egy olyan biztonsági tulajdonság, amelyet a rendszerben található alanyokhoz és objektumokhoz rendelhetünk.

Egy címke beállításához a felhasználónak pontosan ismernie kell, hogy ilyenkor mi történik. Az objektumokhoz tartozó tulajdonságok a betöltött moduloktól függenek, és az egyes modulok eltérő módon értelmezik ezeket a tulajdonságokat. Ha a precíz megértésük hiányában helytelenül állítjuk be ezeket, vagy nem vagyunk képesek tisztázni a velük járó következményeket, akkor az a rendszerünk kiszámíthatatlan és valószínűleg kedvezőtlen viselkedését eredményezi.

A házirendek az objektumhoz rendelt biztonsági címkéket a hozzáféréssel kapcsolatos döntések meghozásában használják fel. Bizonyos házirendek esetében már maga a címke elegendő információt tartalmaz a döntés megformálásához. Máshol viszont a címkék egy nagyobb szabályrendszer részeként dolgozódnak fel stb.

Például, ha egy állományra beállítjuk a `biba/low` címkét, akkor az arra fog utalni, hogy a címkét a Biba nevű biztonsági modul kezeli és értéke “low”.

Az a néhány modul, amely a FreeBSD-ben támogatja a címkézés lehetőségét, három speciális címkét definiál előre. Ezek rendre a “low” (alacsony), “high” (magas) és “equal” (egyező) címkék. Habár az egyes modulok esetén eltérő módon képesek vezérelni a hozzáférést, azt mindig biztosra vehetjük, hogy a “low” a legalacsonyabb érték, az “equal” címke hatására az adott alanyt vagy objektumot érintetlenül hagyják, és a “high” értékű címke a Biba és MLS modulok esetében a legmagasabb beállítást jelenti.

Az egycímkés állományrendszerek használata során az egyes objektumonkhoz csak egyetlen címkét rendelhetünk hozzá. Ezzel az egész rendszerben csak egyfajta engedélyt alkalmazunk, ami sok esetben pontosan elegendő. Létezik néhány különleges eset, amikor az állományrendszerben levő alanyokhoz vagy objektumokhoz egyszerre több címkét is hozzá kell rendelnünk. Ilyenkor a `multilabel` opciót kell átadnunk a `tunefs(8)` segédprogramnak.

A Biba és az MLS esetében előfordulhat, hogy egy numerikus címkével fogjuk jelölni a hierarchikus irányítás pontos szintjét. A numerikus szintek használatával tudjuk az információt különböző csoportokba szétosztani vagy elrendezni, például úgy, hogy csak az adott szintű vagy a felette álló csoportok számára engedélyezzük a hozzáférést.

Az esetek többségében a rendszergazdának csak egyetlen címkét kell beállítania az egész állományrendszerre.

Hé, álljunk csak meg! Akkor ez viszont pont olyan, mint a DAC! Én azt hittem, hogy a MAC szigorúan a rendszergazda kezébe adja az irányítást. Ez az állítás továbbra is fennáll, mivel bizonyos értelemben a `root` lesz az, aki beállítja a házirendeket, tehát ő mondja meg, hogy a felhasználók milyen kategóriákba vagy hozzáférési szintekbe sorolódnak. Sajnos, sok biztonsági modul még magát a `root` felhasználót is korlátozza. Az objektumok feletti irányítás ilyenkor a csoportra száll, de a `root` bármikor visszavonhatja vagy módosíthatja a beállításokat. Ezzel a hierarchikus/engedély alapú modellel a Biba és az MLS nevű házirendek foglalkoznak.

16.4.1. A címkék beállítása

A címkézéshez kapcsolódó összes beállítást gyakorlatilag az alapvető rendszerprogramokkal végezhetjük el. Ezek a parancsok az objektumok és az alanyok szabályozásához, valamint a konfiguráció módosításához és ellenőrzéséhez adnak egy egyszerű kezelőfelületet.

Az összes konfigurációs beállítást a `setfmac(8)` és `setpmac(8)` segédprogramokkal végezhetjük el. A `setfmac` segítségével a rendszerszintű objektumokhoz tudunk hozzárendelni a MAC-címkéket, míg a `setpmac` parancssal a rendszerben levő alanyokhoz tudunk címkéket rendelni. Vegyük például ezt:

```
# setfmac biba/high próba
```

Amennyiben az iménti parancs hibátlanul lefutott, visszakapjuk a parancssort. Ezek a parancsok csak olyankor maradnak nyugodtan, amikor semmilyen hiba nem történt. Működésük hasonló a `chmod(1)` és `chown(8)` parancsokéhoz. Bizonyos esetekben `Permission denied` (A hozzáférés nem engedélyezett) hibát kapunk, ami általában akkor bukkan fel, ha egy korlátozott objektummal kapcsolatban próbálunk meg címkét beállítani vagy módosítani¹. A rendszergazda a következő parancssal tudja feloldani az ilyen helyzeteket:

```
# setfmac biba/high próba
Permission denied
# setpmac biba/low setfmac biba/high próba
# getfmac próba
próba: biba/high
```

Ahogy az itt tetten is érhető, a `setpmac` használható a modul beállításainak felülbírálására úgy, hogy a meghívott programban egy másik címkét állít be. A `getpmac` segédprogram általában a **sendmail**hez hasonló háttérben futó

programok esetében alkalmazható: ilyenkor a konkrét parancs helyett a futó program azonosítóját kell megadnunk, de működése ugyanaz. Ha a felhasználók a hatókörükön túl levő állományokat próbálnak meg módosítani, akkor a betöltött modulok szabályainak megfelelően a `mac_set_link` függvény `Operation not permitted` (A művelet nem engedélyezett) hibát fog adni.

16.4.1.1. Gyakori címketípusok

A `mac_biba(4)`, `mac_mls(4)` és `mac_lomac(4)` moduloknál használhatunk címkéket. Értékük lehet “high”, “equal” vagy “low”, melyek rövid magyarázata a következő:

- A `low` címke az objektumra vagy alanyra érvényes leggyengébb beállítást jelenti. Az ilyen címkéjű objektumok vagy alanyok nem érhetik el a “high” címkéjűeket.
- Az `equal` címke használható minden olyan objektum vagy alany esetében, amelyeket ki akarunk vonni az adott házirend hatálya alól.
- A `high` címke adja az objektumhoz vagy alanyhoz tartozó legerősebb beállítást.

Az egyes moduloktól függően ezek az értékek az információ áramoltatásának különböző irányait írhatják le. A megfelelő man oldalak elolvasásával még jobban megismerhetjük az egyes címketípusok beállításának jellegzetességeit.

16.4.1.1.1. A címkék beállításáról részletesebben

A numerikus osztályozó címkék *összehasonlítás*: `rekesz+rekesz` alakban használatosak, tehát a

`biba/10:2+3+6(5:2+3-20:2+3+4+5+6)`

kifejezés így értelmezhető:

“A Biba házirend címkéje”/“10 osztály” : “2, 3 és 6 rekeszek”: (“5 osztály...”)

Ebben a példában az első osztály tekinthető “valódi osztálynak”, amely a “valódi rekeszeket” jelenti, a második osztály egy alacsonyabb besorolás, míg az utolsó egy magasabb szintű. A legtöbb konfigurációban nem lesz szükségünk ennyire összetett beállításokra, noha képesek vagyunk felírni ezeket.

Ha ezt kivetítjük a rendszer objektumaira, akkor a rendszerben levő alanyokat illetően csupán az aktuális osztály/rekeszek számítanak, mivel a rendszerben és hálózati csatolófelületeken elérhető hozzáférés-vezérlési jogokat tükrözi.

Az alany-objektum párokban megadott osztályzatok és rekeszek használhatóak fel egy olyan kapcsolat kiépítésére, amit “dominanciának” nevezünk. Ilyenkor egy alany ural egy objektumot, vagy egy objektum ural egy alanyt, vagy egyikük sem uralja a másikat, esetleg mind a kettő uralja egymást. A “kettős dominancia” esete akkor forog fenn, amikor a két címke megegyezik. A Biba információáramoltatási sajátosságaiból adódóan jogunk van rekeszeket létrehozni, “tudunk kell”, hogy ezek projekteknek feleltethetőek meg, de az objektumok is rendelkezhetnek rekeszekkel. A felhasználók ilyenkor csak úgy tudnak elérni egyes objektumokat, ha az `su` vagy a `setpmac` használatával leszűkítik a jogaikat egy olyan rekeszre, ahol már nem érvényesülnek rájuk korlátozások.

16.4.1.2. A felhasználók és címkék kapcsolata

Maguknak a felhasználóknak is szükségük van címkékre, mivel csak ezek segítségével tudnak az állományaik és programjaik megfelelő módon együttműködni a rendszerben érvényes biztonsági házirenddel. Ezt a `login.conf`

állományban megadható bejelentkezési osztályokkal állíthatjuk be. Minden címkéket használó modulban a felhasználóknak is van címkéjük.

Lentebb látható egy ilyen minta bejegyzés, amely minden modulhoz tartalmaz beállítást:

```
default:\
:copyright=/etc/COPYRIGHT:\
:welcome=/etc/motd:\
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
:path=~:/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:\
:manpath=/usr/share/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=partition/13,mls/5,biba/10(5-15),lomac/10[2]:
```

Itt a `label` opciót használtuk a felhasználói osztályhoz tartozó alapértelmezett címkék beállításához, amit majd a MAC betartat. A felhasználók nem módosíthatják ezt az értéket, ezért ez a felhasználók számára nem tekinthető tetszőlegesen elhagyható beállításnak. Egy valós konfigurációban azonban a rendszergazda valószínűleg nem akarja majd egyszerre az összes modult használni. Javasoljuk, hogy mielőtt egy ilyen jellegű konfigurációt adnánk meg, olvassuk el az egész fejezetet.

Megjegyzés: A felhasználók ezt a címkét meg tudják változtatni az első bejelentkezés után, de csak a házirend keretein belül. A fenti példában úgy állítjuk be a Biba házirendet, hogy a futó programok sértetlenségi foka legalább 5, legfeljebb 15 lehet, de az alapértéke 10. Tehát a programok egészen addig 10-es szinten futnak, amíg a programok a Biba bejelentkezéskor megadott tartományában meg nem változtatják ezt a címkét, feltehetően a `setpmac` parancs hatására.

Mindig, amikor megváltoztatjuk a `login.conf` beállításait, a `cap_mkdb` paranccsal újra kell generálni a bejelentkezési osztályokhoz tartozó adatbázist, amire a későbbi példákban vagy részekben igyekezzünk is mindig felhívni a figyelmet.

Nem árt hozzátennünk, hogy sok rendszerben kifejezetten sok felhasználót kell kezelnünk, amihez több különböző bejelentkezési osztályra is szükségünk lehet. Mivel később már csak egyre jobban bonyolódni fog a felhasználók kezelése, ezért soha ne felejtünk el komolyan előre tervezni.

A FreeBSD következő változataiban meg fognak jelenni más módszerek is a felhasználók és címkék közti kapcsolatok kezelésére. A FreeBSD 5.3 előtt azonban ez még semmiképpen sem várható.

16.4.1.3. A hálózati csatolófelületek és a címkék kapcsolata

A hálózati csatlakozások esetében is állíthatunk be címkéket, melyek a hálózaton keresztül folyó adatok áramlását határozzák meg. Minden esetben ugyanúgy működnek, mint ahogy a házirendek az objektumokra. Például a `biba` esetében a magas beállításokkal rendelkező felhasználók nem férhetnek hozzá az alacsonyabb címkéjű hálózati csatolófelületekhez.

Ha MAC-címkéket akarunk rendelni egy hálózati felülethez, akkor az `ifconfig` parancsnak adjuk meg a `maclabel` paramétert. Például a

```
# ifconfig bge0 maclabel biba/equal
```

parancs beállítja a `biba/equal` MAC-címkét a `bge(4)` felületre. A `biba/high(low-high)` alakú címkéket átadásukhoz idézőjelek közé kell tenni, különben hibát kapunk.

Minden címkézést támogató modulhoz tartoznak futási időben állítható paraméterek, amelyekkel akár le is tudjuk tiltani a MAC-címkéket a hálózati csatolófelületeken. Ugyanezt jelenti egyébként, ha `equal` értéket adunk meg a címkének. Ezt behatóbban úgy ismerhetjük meg, ha kielemezzük a `sysctl` parancs kimenetét, a megfelelő modul man oldalát vagy a fejezetben további részében található, erre vonatkozó információkat.

16.4.2. Egy címke vagy több címke?

Alapértelmezés szerint a rendszer a `singlelabel` beállítást használja. Ez vajon mit tartogat a rendszergazda számára? Számos olyan eltérést, aminek megvannak a saját előnyei és hátrányai a rendszer védelmi modelljének rugalmassága szempontjából.

A `singlelabel` beállítás minden alany vagy objektum esetében csupán egyetlen címke, például a `biba/high` használatát engedi. Kevesebb adminisztrációs költséggel jár, azonban csökkenteni a címkézést támogató modulok testreszabhatóságát. Ezért sok rendszergazda inkább a `multilabel` beállítást választja a biztonsági házirend kialakítása során.

A `multilabel` beállítás lehetővé teszi, hogy mindegyik alanyhoz és objektumhoz a szabványos `singlelabel` beállítás lehetőségeivel szemben egymástól függetlenül külön-külön rendelhessünk címkéket a partíciókon. Az egy- és többcímkes opciónak csak olyan modulok esetében van értelme, amelyek támogatják a címkézést, mint például a Biba, Lomac, MLS és a SEBSD házirendek.

Sokszor egyáltalán nincs is szükségünk a `multilabel` használatára. Tekintsük például a következő helyzetet és biztonsági modellt:

- Adott egy FreeBSD webszerver, ahol a MAC rendszert több biztonsági házirenddel alkalmazzuk.
- A gépen egyedül csak a `biba/high` címkére van szükségünk mindenhez a rendszerben. Itt egyszerűen csak nem adjuk meg az állományrendszernek a `multilabel` beállítást, mivel az egycímkes rendszer mindig rendelkezésünkre áll.
- Mivel azonban erre a gépre telepíteni akarunk egy webszervert is, ilyenkor a `biba/low` címke használatával igyekszünk korlátozni a szerver feldolgozási képességeit. A Biba házirendről és annak működéséről csak a későbbiekben fogunk írni, ezért ha az előbbi megjegyzést még nem teljesen értjük, akkor egyszerűen csak olvassunk tovább és térjünk vissza ide. A szerver futása alatt, vagy legalább is idejének nagy részében egy külön partíciót használhatna, amire a `biba/low` címkét állítanánk be. Természetesen ez a példa korántsem teljes, hiszen hiányoznak belőle az adatokra érvényes korlátozások, a konfigurációs és felhasználói beállítások. Ez csupán az iménti gondolatmenet gyors illusztrációja.

Amennyiben címkézést nem támogató modulokat alkalmazunk, a `multilabel` beállításra szinte sosem lesz szükségünk. Ilyenek például a `seeotheruids`, `portacl` és `partition` házirendek.

A `multilabel` opció használata és így speciális, többcímkes védelmi modell létrehozása képes elbonyolítani a rendszer karbantartását, mert ilyenkor az állományrendszerben mindennek lennie kell címkéjének: könyvtáraknak, állományok és még az eszközeirőknak is.

A most következő paranccsal beállítjuk az állományrendszerre a `multilabel` opciót. Ez csak egyfelhasználós módban tehető meg:

```
# tuneefs -l enable /
```

A lapozópartíció esetében erre nincs szükség.

Megjegyzés: Előfordulhat, hogy néhány felhasználónak nem sikerül a `multilabel` opciót beállítania a rendszerindító partícióra. Ha ez történne, akkor olvassuk el a fejezet 16.17 Szakaszát.

16.5. A védelem megtervezése

Mindig hasznos időt szánni a tervezésre, amikor nekilátunk egy új technológia alkalmazásához. A tervezés közben a rendszergazdának “egyben kell látnia a képet”, lehetőleg az alábbiak figyelembevételével:

- Elvárások a modell felé
- A modell célkitűzései

Továbbá a MAC használata esetén:

- Miként osztályozzuk a célrendszeren rendelkezésre álló információt és erőforrásokat
- Milyen információt vagy erőforrást kell korlátoznunk és milyen típusú korlátozást alkalmazunk rájuk
- A MAC melyik moduljain keresztül tudjuk elérni céljainkat

Habár mindig módunkban áll megváltoztatni és újra konfigurálni a rendszerben található erőforrásokat és biztonsági beállításokat, sokszor azért igen kényelmetlen utánanézni a rendszerben és állítgatni az állományok, illetve felhasználói hozzáférések paramétereit. A beállításainkat valamint azok konfigurációját *először* külön próbáljuk ki, mielőtt a MAC alapú megvalósításunkat egy éles rendszeren kezdjük el használni. Ennek elhagyása szinte biztosan kudarcra ítéel minket.

A különböző környezetek igényei és elvárásai eltérnek. Egy alaposan és minden részletében átgondolt védelmi profil megalapozása csökkenti a rendszer üzembehelyezése után elvégzendő módosítások számát. Mint olyanokra, a következő szakaszokban kitérünk a rendszergazdák számára elérhető modulokra, bemutatjuk a használatukat és beállításukat és egyes esetekben betekintést is adunk olyan helyzetekbe, ahol a legjobban kiaknázhatóak a képességeik. Például egy webszerver esetén hasznos lehet a `mac_biba(4)` és `mac_bsextended(4)` házirendek alkalmazása. Más esetekben, például egy kevés felhasználóval működő számítógépen, a `mac_partition(4)` modul lehet jó választás.

16.6. A modulok beállítása

A MAC rendszerben megtalálható összes modul a korábban leírtak szerint beépíthető a rendszermagba vagy menet közben is betölthető modulként. A használni kívánt modulokat a `/boot/loader.conf` állományba javasolt felvenni, így azok be tudnak tölteni a rendszer indítása folyamán.

A soron következő szakaszokban a különböző MAC-modulokat dolgozzuk fel és foglaljuk össze a lehetőségeiket. Továbbá a fejezet szeretne szólni ezek alkalmazásáról speciális helyzetekben is. Egyes modulokkal címkézni is tudunk, aminek révén a hozzáféréseket címkékkel szabályozzuk, például úgy, hogy megmondjuk “mit szabad és mit nem”. A címkék beállításait tartalmazó állomány vezérli az állományok elérését, a hálózati kommunikációt és még sok minden mást. Az előző szakaszban már megismerhettük, hogy a `multilabel` opció segítségével hogyan állíthatjuk be az állományonkénti vagy partíciónkénti hozzáférés-vezérlést.

Az egycímkés konfigurációban az egész rendszerben csupán egyetlen címke használatára nyílik mód, ezért is hívják a `tunefs` beállítását `multilabel`nek.

16.7. A seeotheruids MAC-modul

A modul neve: `mac_seeotheruids.ko`

A rendszermag konfigurációs beállítása: `options MAC_SEEOTHERUIDS`

Rendszerindítási beállítás: `mac_seeotheruids_load="YES"`

A `mac_seeotheruids(4)` modul a `security.bsd.see_other_uids` és `security.bsd.see_other_gids` `sysctl`-változókat utánozza és terjeszti ki. A használatához semmilyen címkét nem kell beállítani és transzparens módon képes együttműködni a többi modullal.

A modult betöltése után az alábbi `sysctl`-változókkal tudjuk vezérelni:

- A `security.mac.seeotheruids.enabled` engedélyezi a modult és az alapértelmezett beállításokat használja. Alapértelmezés szerint egyik felhasználó sem láthatja a többiek futó programjait és csatlakozásait.
- A `security.mac.seeotheruids.specificgid_enabled` egy adott csoportot mentesít a házirend szabályozásai alól. Tehát ki akarunk vonni egy csoportot a házirend alkalmazásából, akkor állítsuk be a `security.mac.seeotheruids.specificgid=xxx` `sysctl`-változót, ahol az `xxx` a mentesíteni kívánt csoport numerikus azonosítója.
- A `security.mac.seeotheruids.primarygroup_enabled` segítségével adott elsődleges csoportokat vonhatunk ki a házirend hatálya alól. Ezt a változót nem használhatjuk a `security.mac.seeotheruids.specificgid_enabled` változóval együtt.

16.8. A bsdextended MAC-modul

A modul neve: `mac_bsdextended.ko`

A rendszermag konfigurációs beállítása: `options MAC_BSDEXTENDED`

Rendszerindítási beállítás: `mac_bsdextended_load="YES"`

A `mac_bsdextended(4)` modul segítségével egy állományrendszer szintjén működő tűzfalat tudunk kialakítani. Ez a modul a szabványos állományrendszeri engedély alapú modelljét bővíti ki, lehetővé téve, hogy a rendszergazda tűzfalszerű szabályokkal nyújtson védelmet a könyvtárszerkezetben található állományoknak, segédprogramoknak és

könyvtáraknak. Amikor egy állományrendszerbeli objektumhoz próbálunk meg hozzáférni, a modul illeszti ezt egy szabályrendszerre, amiben vagy talál egy hozzá tartozó szabályt vagy kifut belőle. Ez a viselkedés a `security.mac.bsextended.firstmatch_enabled` `sysctl(8)` paraméter segítségével változtatható meg. Hasonlóan a FreeBSD-ben található többi tűzfalmodulhoz, az állományok elérését definiáló szabályok a rendszerindítás során egy `rc.conf(5)` változóból olvasódnak be.

A szabályokat a `ugidfw(8)` segédprogrammal adhatjuk meg, amelynek a formai szabályai hasonlóak az `ipfw(8)` programéhoz. A `libugidfw(3)` függvénykönyvtár felhasználásával azonban további segédprogramok is írhatóak hozzá.

A modul használata során igyekezzünk minél jobban odafigyelni, mert helytelen alkalmazásával el tudjuk vágni magunkat az állományrendszer bizonyos részeitől.

16.8.1. Példák

Miután sikerült betölteni a `mac_bsextended(4)` modult, a következő paranccsal tudjuk lekérdezni a jelenleg érvényes szabályokat:

```
# ugidfw list
0 slots, 0 rules
```

Ahogy az várható is volt, pillanatnyilag még egyetlen szabályt sem adtunk meg. Ennek értelmében tehát mindent el tudunk érni. A következő paranccsal tudunk olyan szabályt létrehozni, ahol a `root` kivételével elutasítjuk az összes felhasználó hozzáférését:

```
# ugidfw add subject not uid root new object not uid root mode n
```

Ez egyébként egy nagyon buta ötlet, mivel így a felhasználók még a legegyszerűbb parancsokat, mint például az `ls-t`, sem tudják rájuk kiadni. Ennél sokkal humánusabb lesz, ha:

```
# ugidfw set 2 subject uid felhasználó1 object uid felhasználó2 mode n
# ugidfw set 3 subject uid felhasználó1 object gid felhasználó2 mode n
```

Ilyenkor a `felhasználó1` nevű felhasználótól megvonjuk a `felhasználó2` felhasználói könyvtárának összes hozzáférését, beleértve a listázhatóságot is.

A `felhasználó1` helyett megadhatjuk a `not uid felhasználó2` opciót is. Ebben az esetben egy felhasználó helyett az összes felhasználóra ugyanaz a korlátozás fog érvényesülni.

Megjegyzés: A `root` felhasználóra ezek a beállítások nem vonatkoznak.

Ezzel felvázoltuk, miként lehet a `mac_bsextended(4)` modult felhasználni az állományrendszerek megerősítésére. Részletesebb információért járuljunk a `mac_bsextended(4)` és `ugidfw(8)` man oldalakhoz.

16.9. Az ifoff MAC-modul

A modul neve: `mac_ifoff.ko`

A rendszermag konfigurációs beállítása: `options MAC_IFOFF`

Rendszerindítási beállítás: `mac_loff_load="YES"`

A `mac_loff(4)` modul kizárólag abból a célból készült, hogy segítségével menet közben le tudjuk tiltani bizonyos hálózati csatolófelületek beállítását a rendszerindítás közben. Sem címkékre, sem pedig a többi MAC-modulra nincs szükségünk a használatához.

A vezérlést nagyrészt az alábbi `sysctl`-változókkal tudjuk megoldani.

- A `security.mac.loff.lo_enabled` engedélyezi vagy letiltja a `(lo(4))` helyi loopback felületen az összes forgalmat.
- A `security.mac.loff.bpfrecv_enabled` engedélyezi vagy letiltja a Berkeley csomagszűrő (BPF, Berkeley Packet Filter) felületén az összes forgalmat.
- A `security.mac.loff.other_enabled` engedélyezi vagy letiltja az összes többi csatolófelületen az összes forgalmat.

A `mac_loff(4)` modult általában olyan környezetek monitorozásakor szokták használni, ahol a rendszer indítása során még nem szabad hálózati forgalomnak keletkeznie. Vagy például a `security/aide` porttal együtt használva automatikusan el tudjuk zárni a rendszerünket, ha a védett könyvtárakban új állományok keletkeznek vagy megváltoznak a régié.

16.10. A portacl MAC-modul

A modul neve: `mac_portacl.ko`

A rendszermag konfigurációs beállítása: `MAC_PORTACL`

Rendszerindítási beállítás: `mac_portacl_load="YES"`

A `mac_portacl(4)` modul a helyi TCP és UDP portok kiosztásának korlátozását teszi lehetővé különféle `sysctl`-változókon keresztül. A `mac_portacl(4)` segítségével lényegében a nem-root felhasználók is használhatnak privilegizált, tehát 1024 alatti portokat.

Miután betöltöttük, a modul az összes csatlakozásra alkalmazza a MAC-házirendet. Ezután az alábbi változókkal hangolhatjuk a viselkedését:

- A `security.mac.portacl.enabled` totálisan engedélyezi vagy letiltja a házirend használatát.
- A `security.mac.portacl.port_high` megadja azt a legmagasabb portot, amelyre még kiterjed a `mac_portacl(4)` védelme.
- Ha a `security.mac.portacl.suser_exempt` változónak nem nulla értéket adunk meg, akkor azzal a root felhasználót kivonjuk a szabályozások alól.
- A `security.mac.portacl.rules` az érvényes `mac_portacl` házirendet adja meg, lásd lentebb.

A `security.mac.portacl.rules` változó által megadott aktuális `mac_portacl` házirend formátuma a következő: `szabály[, szabály, ...]`, ahol ezen a módon tetszőleges számú szabályt adhatunk meg. Az egyes szabályok pedig így írhatóak fel: `azonosítótípus: azonosító: protokoll: port`. Az `azonosítótípus` értéke `uid` vagy `gid` lehet, amivel megadjuk, hogy az `azonosító` paraméter felhasználóra vagy csoportra hivatkozik. A `protokoll` paraméter adja meg, hogy a szabályt TCP vagy UDP típusú kapcsolatra értjük, és ennek megfelelően az értéke is `tcp` vagy `udp` lehet. A sort végül a `port` paraméter zárja, ahol annak a portnak számát adjuk meg, amelyhez az adott felhasználót vagy csoportot akarjuk kötni.

Megjegyzés: Mivel a szabályokat közvetlenül maga a rendszermag dolgozza fel, ezért a felhasználók illetve csoportok azonosítója, valamint a port értéke kizárólag numerikus érték lehet. Tehát a szabályokban név szerint nem hivatkozhatunk felhasználókra, csoportokra vagy szolgáltatásokra.

A UNIX-szerű rendszereken alapértelmezés szerint az 1024 alatti portokat csak privilegizált programok kaphatják meg és használhatják, tehát a `root` felhasználó neve alatt kell futniuk. A `mac_portacl(4)` azonban a nem privilegizált programok számára is lehetővé teszi, hogy elfoglalhassanak 1024 alatti portokat, amihez viszont először le kell tiltani ezt a szabvány UNIX-os korlátozást. Ezt úgy érhetjük el, ha a `net.inet.ip.portrange.reservedlow` és `net.inet.ip.portrange.reservedhigh` változókat egyaránt nullára állítjuk.

A `mac_portacl(4)` működésének részleteiről a példákon keresztül vagy a megfelelő man oldalakból tudhatunk meg többet.

16.10.1. Példák

A következő példák az iméntieket igyekeznek jobban megvilágítani:

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0 net.inet.ip.portrange.reservedhigh=0
```

Elsőként beállítjuk, hogy a `mac_portacl(4)` vegye át a szabványos privilegizált portok vezérlését és letiltjuk a normál UNIX-os korlátozásokat.

```
# sysctl security.mac.portacl.suser_exempt=1
```

A `root` felhasználót azonban nem akarjuk kitenni a házirendnek, ezért a `security.mac.portacl.suser_exempt` változónak egy nem nulla értéket adunk meg. A `mac_portacl(4)` modul most pontosan ugyanúgy működik, mint a UNIX-szerű rendszerek alapértelmezés szerint.

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

A 80-as azonosítóval rendelkező felhasználó (aki általában a `www`) számára engedélyezzük a 80-as port használatát. Így a `www` felhasználó anélkül képes webszervert futtatni, hogy szüksége lenne a `root` jogosultságaira.

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

Az 1001-es azonosítóval rendelkező felhasználónak megengedjük, hogy elfoglalhassa a 110-es (“pop3”) és 995-ös (“pop3s”) portokat. Ennek köszönhetően az adott felhasználó el tud indítani egy szervert, amihez a 110-es és 995-ös portokon lehet kapcsolódni.

16.11. A partition MAC-modul

A modul neve: `mac_partition.ko`

A rendszermag konfigurációs beállítása: `options MAC_PARTITION`

Rendszerindítási beállítás: `mac_partition_load="YES"`

A `mac_partition(4)` házirend a futó programokat címkéjük szerint adott “partíciókra” osztja szét. Ezt leginkább egy speciális `jail(8)` megoldásként tudjuk elképzelni, noha teljesen felesleges összehasonlítani a kettőt.

Ez egy olyan modul, amelyet a `loader.conf(5)` állományba kell felvenni, hogy a rendszerindítása közben be tudjon tölteni.

Ezt a házirendet többségében a `setpmac(8)` segédprogrammal tudjuk állítgatni, ahogy az majd lentebb látható lesz. A következő `sysctl`-változó tartozik még a modulhoz:

- A `security.mac.partition.enabled` engedélyezi a futó programok MAC rendszeren keresztüli felosztását.

A házirend engedélyezésével a felhasználók csak a saját programjaikat láthatják, illetve mindazokat, amelyek az övékével egy partícióba tartoznak, de a rajta kívül levő programokkal már nem dolgozhatnak. Például, ha egy felhasználó az `insecure` (“nem biztonságos”) osztály tagja, akkor ne engedjük, hogy hozzáférhessen a `top` vagy bármilyen más olyan parancshoz, amely további futó programokat hoz létre.

A `setpmac` használatával tudunk címkéket készíteni a partíciókhoz és programokat rendelni hozzájuk:

```
# setpmac partition/13 top
```

Így a `top` parancsot hozzáadjuk az `insecure` osztályban levő felhasználókhoz rendelt címkéhez. Vegyük észre, hogy az `insecure` osztályba tartozó felhasználók által elindított összes program a `partition/13` címkét fogja használni.

16.11.1. Példák

A következő parancs megmutatja a partíciók címkéit és a futó programok listáját:

```
# ps Zax
```

Ezzel paranccsal pedig megnézhetjük egy másik felhasználó programjainak címkéit és a felhasználó által futtatott programokat:

```
# ps -ZU trhodes
```

Megjegyzés: A felhasználók látják a `root` címkéjével futó programokat is, hacsak be nem töltjük a `mac_seeotheruids(4)` házirendet.

Ezt a megoldást úgy tudnánk igazán ravaszul felhasználni, ha például az `/etc/rc.conf` állományban letiltanánk az összes szolgáltatást és egy olyan szkripttel indítanánk el ezeket, amely futtatásuk előtt beállítja hozzájuk a megfelelő címkét.

Megjegyzés: A most következő házirendek a három alapértelmezett címkeérték helyett egész számokat használnak. Ezekről, valamint a rájuk vonatkozó korlátozásokról a megfelelő modulok man oldalain ismerhetünk meg többet.

16.12. A többszintű biztonsági MAC-modul

A modul neve: `mac_mls.ko`

A rendszermag konfigurációs beállítása: `options MAC_MLS`

Rendszerindítási beállítás: `mac_mls_load="YES"`

A `mac_mls(4)` (MLS, Multi-Level Security) házirend az információ szigorú áramoltatásával vezérli a rendszerben található alanyok és objektumok közti elérést.

A MLS megoldását alkalmazó környezetekben a rekeszek mellett minden alanyra és objektumra be kell még állítanunk egy adott szintű "engedélyt" is. Mivel az engedélyek avagy az érzékenység szintje akár a hatezret is meghaladhatja, egy rendszergazda számára valódi rémálommá válhat az egyes alanyok és objektumok precíz beállítása. Szerencsére a házirend erre a célra tartalmaz három előre definiált "instant" címkét.

Ezek az `mls/low`, `mls/equal` és `mls/high`. Mivel a man oldal elég részletesen kifejti ezeket, ezért itt csak érintőlegesen foglalkozunk velük:

- Az `mls/low` címke egy olyan alacsony szintű beállítást képvisel, amely lehetővé teszi, hogy az összes többi objektum uralja. Tehát bárminek is adjuk az `mls/low` címkét, alacsony szintű engedéllyel fog rendelkezni és nem lesz képes elérni a magasabb szinten levő információt. Ráadásul a címke a magasabb szintű objektumok számára se fogja engedni, hogy információt közöljön vagy adjon át az alacsonyabb szintek felé.
- Az `mls/equal` címke olyan objektumok esetében ajánlott, amelyeket ki akarunk hagyni a házirend szabályozásaiból.
- Az `mls/high` címke az elérhető legmagasabb szintű engedélyt ábrázolja. Az ilyen címkével ellátott objektumok a rendszer összes többi objektuma felett uralommal rendelkeznek, habár az alacsonyabb szintű objektumok felé nem képesek információt közvetíteni.

Az MLS:

- Egy hierarchikus védelmi szinteket épít fel nem hierarchikus kategóriákkal.
- Szabályai rögzítettek: a felsőbb szintek olvasása és az alsóbb szintek írása egyaránt tiltott (az alanyok csak a saját vagy az alatta levő szinteken levő objektumokat képesek olvasni, de a felette állókat már nem. Ehhez hasonlóan az alanyok a velük egyező vagy a felsőbb szinteket tudják írni, de az alattuk levőket már nem).
- Megőrzi a titkokat (megakadályozza az adatok alkalmatlan közzétételét).
- Megadja mindazt az alapot, ami szükséges ahhoz, hogy az adatokat több kényességi szinten, párhuzamosan is kezelni tudjuk (anélkül, hogy titkos és bizalmas információkat szivárogtatnánk ki).

A speciális szolgáltatások és felületek beállításához az alábbi `sysctl`-változók használhatóak:

- A `security.mac.mls.enabled` engedélyezi vagy tiltja le az MLS házirend alkalmazását.
- A `security.mac.mls.ptys_equal` hatására látja el `mls/equal` címkével az összes `pty(4)` eszközt létrehozásuk során.
- A `security.mac.mls.revocation_enabled` használható az alacsonyabb szintre minősített objektumok hozzáféréseinek megvonására.
- A `security.mac.mls.max_compartments` segítségével adható meg az objektumok által használt rekeszek szintjének maximális száma. Lényegében a rekeszek rendszerben engedélyezett maximuma.

Az MLS címkéit a `setfmac(8)` paranccsal tudjuk módosítani. Egy ehhez hasonló paranccsal tudunk egy objektumhoz címkét rendelni:

```
# setfmac mls/5 próba
```

A *próba* állomány MLS-címkéjét az alábbi paranccsal kérhetjük le:

```
# getfmac próba
```

Ezzel össze is foglaltuk az MLS házirend lehetőségeit. Az eddigiket úgy is megoldhatjuk, hogy létrehozunk egy központi házirendet az */etc* könyvtárban, amelyben megadjuk az MLS házirendhez tartozó információkat, majd átadjuk a *setfmac* parancsnak. Erre a módszerre majd a házirendek bemutatása után kerül sor.

16.12.1. A kényesség megállapítása

A többszintű biztonsági házirend használatával a rendszergazda a kényes információk áramlásának irányát tudja befolyásolni. A megoldás “felfele nem lehet olvasni, lefele nem lehet írni” jellege folytán alapból mindent a legalacsonyabb szintre helyez. Így tehát kezdetben minden elérhető, és a rendszergazdának lassanként ebből az állapotból elindulva kell behangolnia az erre alapozó védelmi rendszert az információ bizalmasságának megfelelően.

A fentebb említett három alapvető címke mellett a rendszergazdának valószínűleg szüksége lesz a felhasználók csoportosítására és a csoportok közti információáramlás szabályozására. A információ bizalmasságának szintjeit minden bizonnyal könnyebb szavakkal beazonosítani, például *Confidential* (bizalmas), *Secret* (titkos) vagy *Top Secret* (szigorúan bizalmas). Bizonyos helyzetekben elég csak a futó projekteknek megfelelően kialakítani csoportokat. Az osztályozás konkrét módszerétől függetlenül azonban mindig elmondható, hogy előzetes tervezés nélkül sose állítsunk össze ilyen fajsúlyú házirendet.

Ezt a biztonsági modult például webes üzletek esetén érdemes használnunk, ahol egy állományszerver tárolja a cég fontos adatait és pénzügyi információit. Viszont egy két vagy három felhasználóval üzemelő munkaállomás esetében szinte teljesen felesleges gondolkodni rajta.

16.13. A Biba MAC-modul

A modul neve: *mac_biba.ko*

A rendszermag konfigurációs beállítása: *options MAC_BIBA*

Rendszerindítási beállítás: *mac_biba_load="YES"*

A *mac_biba(4)* modul a MAC Biba elnevezésű házirendjét tölti be. Ez leginkább az MLS házirendhez hasonlít, azzal a kivétellel, hogy az információ áramoltatására vonatkozó szabályok némileg visszafelé működnek. Tehát míg az MLS házirend a kényes információ áramlását felfelé nem engedi, addig ez a lefelé irányuló áramlást állítja meg. Emiatt ez a szakasz tulajdonképpen mind a két házirendre érvényesül.

A Biba alkalmazása során minden alany és objektum egy “sértetlenséget” jelképező címkét visel. Ezek a címkék hierarchikus osztályokból, nem pedig hierarchikus összetevőkből származnak. Egy objektum vagy alany sértetlensége a besorolásával növekszik.

A modul a *biba/low*, *biba/equal* és *biba/high* címkéket ismeri, vagyis bővebben:

- A *biba/low* címke tekinthető az alanyok és objektumok legkisebb sértetlenségének. Ha beállítjuk egy objektumra vagy alanyra, akkor ezzel megakadályozzuk, hogy nagyobb sértetlenségű objektumokat vagy alanyokat tudjanak írni. Ettől függetlenül azonban még képesek olvasni ezeket.
- A *biba/equal* címke használata kizárólag olyan objektumok esetében javasolt, amelyeket ki akarunk vonni a házirend alól.

- A `biba/high` címke megengedi az alacsonyabb szinteken levő objektumokat írását, de az olvasását viszont már nem. Ezt a címkét olyan objektumra érdemes ragasztani, amelyek hatással vannak az egész rendszer sértetlenségére.

A Biba:

- Hierarchikus sértetlenségi szinteket épít fel nem hierarchikus sértetlenségi kategóriákkal kiegészítve.
- Szabályai rögzítettek: az felsőbb szintek írása és az alsóbb szintek olvasása egyaránt tilos (pontosan az MLS ellentéte). Egy alany csak a saját vagy az alatta álló szinteken szereplő objektumokat tudja írni. Ehhez hasonló módon egy alany csak a saját vagy az afeletti szinten található objektumokat képes olvasni.
- Az adatok sértetlenségét biztosítja (megakadályozza az alkalmatlan módosításukat)
- Sértetlenségi szinteket határoz meg (szemben az MLS kényességi szintjeivel).

Az alábbi `sysctl`-változókkal vezérlhetjük a Biba házirend működését:

- A `security.mac.biba.enabled` használható a célrendszeren a Biba házirend engedélyezésére vagy letiltására.
- A `security.mac.biba.ptys_equal` segítségével kapcsolhatjuk ki a Biba házirend alkalmazását a `pty(4)` eszközökön.
- A `security.mac.biba.revocation_enabled` hatására visszavonódik az objektumok hozzáférése, ha az rájuk vonatkozó címke megváltozik.

A rendszer objektumain a Biba házirendet a `setfmac` és `getfmac` paranccsal állíthatjuk be:

```
# setfmac biba/low próba
# getfmac próba
próba: biba/low
```

16.13.1. A sértetlenség megállapítása

A sértetlenség a kényességtől eltérően azt igyekszik szavatolni, hogy az információt illetéktelenek nem módosítják. Ez egyaránt vonatkozik az alanyok, objektumok és a kettő között átadott adatokra. Gondoskodik róla, hogy a felhasználók csak olyan információkat változtathassanak meg, sőt csak olyat érthessenek el, amire ténylegesen szükségük van.

A `mac_biba(4)` biztonsági modul megengedi a rendszergazda számára, hogy megmondja milyen állományokat és programokat láthat vagy hívhat meg a felhasználó vagy felhasználók egy csoportja, miközben biztosítja, hogy az állományok és a programok nincsenek kitéve semmilyen fenyegetésnek, és a rendszer az adott felhasználóban vagy felhasználói csoportban megbízik.

A kezdeti tervezési fázis során a rendszergazdának fel kell készülnie arra, hogy a felhasználókat osztályokra, szintekre és területekre kell osztania. A felhasználók nem csak adatokhoz, hanem programokhoz és segédprogramokhoz sem lesznek képesek hozzáférni, mind az indításuk előtt és után. A modul aktiválás után a rendszer alaphoz rögtön a legmagasabb címkét kapja meg, és teljesen a rendszergazdára hárul, hogy a felhasználókhoz beállítsa a különféle osztályokat és szinteket. A fentebb leírt engedélyszintek helyett akár témák alapján is tervezhetünk. Például kizárólag csak a fejlesztők számára engedjük meg a forráskód módosítását, a forráskód lefordítását és a többi fejlesztőeszköz használatát. Eközben a többi felhasználót felosztjuk további csoportokba, például tesztelőkre és tervezőkre, vagy meghagyjuk ezeket átlagos felhasználóknak, akik csak olvasási joggal rendelkeznek.

A megvalósított biztonsági modell természetéből fakadóan egy kevésbé sértetlenebb alany nem írhatja a sokkal sértetlenebb alanyokat, a sokkal sértetlenebb alanyok pedig nem érhetik el vagy olvashatják a kevésbé sértetlen objektumokat. A lehető legkisebb osztályú címke beállításával gyakorlatilag elérhetetlenné teszük az alanyok számára. A modult valószínűleg egy korlátozott webservert, fejlesztői- és tesztgépek vagy forráskód tárolására szánt környezetben érdemes bevetni. Annál esélytelenebb a használata viszont egy munkaállomás, útválasztó vagy hálózati tűzfal esetében.

16.14. A LOMAC MAC-modul

A modul neve: `mac_lomac.ko`

A rendszermag konfigurációs beállítása: `options MAC_LOMAC`

Rendszerindítás beállítás: `mac_lomac_load="YES"`

Eltérően a MAC Biba házirendjétől, a `mac_lomac(4)` egyedül csak azután engedi elérni az kevésbé sértetlenebb objektumokat, miután csökkentjük a sértetlenség szintjét és ezzel betartjuk a sértetlenségre vonatkozó szabályokat.

A gyenge vízjeles sértetlenségi házirend MAC alapú változatát nem szabad összetéveszteni a korábbi `lomac(4)` implementációval, amely majdnem ugyanúgy működik, mint a Biba, azzal az a kivétellel, hogy a lebegő címkékkel támogatjuk az alanyok lefokozását egy kisegítő osztály rekeszén keresztül. Ez a másodlagos rekesz `[kisegítő_osztály]` alakú. Tehát amikor egy kisegítő osztállyal adjuk meg a `lomac` házirendet, valahogy így néz ki: `lomac/10[2]`, ahol a kettes (2) szám ez a kisegítésre használt osztály.

A MAC LOMAC házirendje az összes rendszerszintű objektum esetében jelenlevő sértetlenségi címkézéssel alapszik, megengedve az alanyok számára, hogy az kevésbé sértetlen objektumokat olvasni tudják, majd a címke leminősítésével az alany meg tudja akadályozni a sokkal sértetlenebbnek ítélt objektumok jövőbeni írását. Ez az a fentebb tárgyalt `[kisegítő_osztály]` opció, ezért ez a modul a Bibánál több kompatibilitást és kevesebb kezdeti beállítást igényel.

16.14.1. Példák

Hasonlóan a Biba és MLS házirendeknél megszokottakhoz, a `setfmac` és `setpmac` segédprogramok használhatóak a címkék hozzárendeléséhez:

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes lomac/high[low]
```

Itt a kisegítő osztály a `low`. Ezt csak a LOMAC MAC-házirendnél adhatjuk meg.

16.15. A Nagios elzárása a MAC rendszerrel

A most következő bemutatóban a MAC moduljainak és a megfelelően beállított házirendek használatával fogunk kialakítani egy biztonságos környezetet. Ne feledjük azonban, hogy ez csupán egy ártatlan próba és nem pedig a mindenki biztonsági aggályait kielégítő legvégső megoldás. Ha egy házirendet vakon építünk fel és nem értjük meg a működését, az soha nem válik hasznunkra, és egy éles helyzetben katasztrofális hatással járhat.

A folyamat megkezdése előtt be kell állítanunk a `multilabel` opciót mindegyik állományrendszerre, a fejezet elején leírtaknak megfelelően. Ha ezt a lépést kihagyjuk, akkor hibákat kapunk. Továbbá még az előkészület részeként ne felejtünk el gondoskodni a `net-mngt/nagios-plugins`, `net-mngt/nagios` és `www/apache13` portok telepítéséről, beállításáról és megfelelő működéséről sem.

16.15.1. A nem megbízható felhasználók osztályának létrehozása

Az eljárást kezdjük az alábbi (insecure) felhasználói osztály hozzáadásával az `/etc/login.conf` állományban:

```
insecure:\
:copyright=/etc/COPYRIGHT:\
:welcome=/etc/motd:\
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
:path=~:/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
:manpath=/usr/share/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=biba/10(10-10):
```

Valamint egészítsük ki az alapértelmezett (default) felhasználói osztályt a következő sorral:

```
:label=biba/high:
```

Ahogy ezzel elkészültünk, az hozzá tartozó adatbázis újbóli legyártásához a következő parancsot kell kiadnunk:

```
# cap_mkdb /etc/login.conf
```

16.15.2. A rendszerindítással kapcsolatos beállítások

Még ne indítsuk újra a számítógépet, csupán a szükséges modulok betöltéséhez bővítsük ki a `/boot/loader.conf` állományt az alábbi sorokkal:

```
mac_biba_load="YES"
mac_seeotheruids_load="YES"
```

16.15.3. A felhasználók beállítása

Soroljuk be a root felhasználót a default osztályba:

```
# pw usermod root -L default
```

Az összes root felhasználón kívüli hozzáférésnek vagy rendszerfelhasználónak most kellene fog egy bejelentkezési osztály. A bejelentkezési osztályra egyébként is szükség lesz, mert ennek hiányában a felhasználók még az olyan egyszerű parancsokat sem tudják kiadni, mint például a vi(1). A következő sh szkript nekünk erre pontosan megfelel:

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' \
    /etc/passwd`; do pw usermod $x -L default; done;
```

Helyezzük át a nagios és www felhasználókat az insecure osztályba:

```
# pw usermod nagios -L insecure
```

```
# pw usermod www -L insecure
```

16.15.4. A contexts állomány létrehozása

Most csinálnunk kell egy contexts állományt. Ebben példában az /etc/policy.contexts állományt használjuk.

```
# Ez a rendszer alapértelmezett BIBA házirendje.
```

```
# Rendszer:
/var/run                biba/equal
/var/run/*              biba/equal

/dev                    biba/equal
/dev/*                  biba/equal

/var                    biba/equal
/var/spool              biba/equal
/var/spool/*            biba/equal

/var/log                biba/equal
/var/log/*              biba/equal

/tmp                    biba/equal
/tmp/*                  biba/equal
/var/tmp                biba/equal
/var/tmp/*              biba/equal

/var/spool/mqueue       biba/equal
/var/spool/clientmqueue biba/equal

# Nagios:
/usr/local/etc/nagios
/usr/local/etc/nagios/* biba/10

/var/spool/nagios        biba/10
```

```

/var/spool/nagios/*          biba/10

# Apache:
/usr/local/etc/apache       biba/10
/usr/local/etc/apache/*     biba/10

```

Ezzel a házirenddel az információ áramlását szabályozzuk. Ebben a konkrét konfigurációban a felhasználók, a `root` és társai, nem férhetnek hozzá a **Nagios**hoz. A **Nagios** beállításait tároló állományok és a neve alatt futó programok így teljesen különválnak vagyis elzáródnak a rendszer többi részétől.

Ez az iménti állomány a következő parancs hatására kerül be a rendszerünkbe:

```

# setfsmac -ef /etc/policy.contexts /
# setfsmac -ef /etc/policy.contexts /

```

Megjegyzés: A fenti állományrendszer felépítése a környezettől függően eltérhet, habár ezt minden egyes állományrendszeren le kell futtatni.

Az `/etc/mac.conf` állományt törzsét a következőképpen kell még átírnunk:

```

default_labels file ?biba
default_labels ifnet ?biba
default_labels process ?biba
default_labels socket ?biba

```

16.15.5. A hálózat engedélyezése

Tegyük hozzá a következő sort az `/boot/loader.conf` állományhoz:

```
security.mac.biba.trust_all_interfaces=1
```

Ezt az alábbi beállítást pedig szúrjuk be az `rc.conf` állományba a hálózati kártya konfigurációjához. Amennyiben az internetet DHCP segítségével érjük el, ezt a beállítást manuálisan kell megtenni minden rendszerindítás alkalmával:

```
maclabel biba/equal
```

16.15.6. A konfiguráció kipróbálása

Gondoskodjunk róla, hogy a webservert és a **Nagios** nem fog elindulni a rendszer indításakor, majd indítsuk újra a gépet. Ezenkívül még ellenőrizzük, hogy a `root` ne tudjon hozzáférni a **Nagios** beállításait tartalmazó könyvtárhoz. Ha a `root` képes kiadni egy `ls(1)` parancsot a `/var/spool/nagios` könyvtárra, akkor valamit elronthattunk. Normális esetben egy `permission denied` üzenetet kell kapnunk.

Ha minden jónak tűnik, akkor a **Nagios**, **Apache** és **Sendmail** most már elindítható a biztonsági házirend szabályozásai szerint. Ezt a következő parancsokkal tehetjük meg:

```

# cd /etc/mail && make stop && \
setpmac biba/equal make start && setpmac biba/10\10-10\ apachectl start && \
setpmac biba/10\10-10\ /usr/local/etc/rc.d/nagios.sh forcestart

```

Kétszer is ellenőrizzük, hogy minden a megfelelő módon viselkedik-e. Ha valamilyen furcsaságot tapasztalunk, akkor nézzük át a naplókat vagy a hibaüzeneteket. A `sysctl(8)` használatával tiltsuk le a `mac_biba(4)` biztonsági modult és próbáljunk meg mindent a szokott módon újraindítani.

Megjegyzés: A `root` felhasználó különösebb aggodalom nélkül képes megváltoztatni a biztonsági rend betartatását és átírni a konfigurációs állományokat. Egy frissen indított parancsértelmező számára ezzel a paranccsal tudjuk csökkenteni a biztonsági besorolást:

```
# setpmac biba/10 csh
```

Ennek kivédésére a felhasználókat a `login.conf(5)` beállításával le kell korlátozni. Ha a `setpmac(8)` megpróbál a rekesz határain túl futtatni egy parancsot, akkor hibát ad vissza és a parancs nem fut le. Ebben az esetben a `root` felhasználót tegyük a `biba/high(high-high)` értékek közé.

16.16. A felhasználók korlátozása

Ebben a példában egy viszonylag kicsi, nagyjából mindössze ötven felhasználós, adattárolásra használatos rendszert veszünk alapul. A felhasználók rendelkezhetnek bizonyos bejelentkezési tulajdonságokkal, és nem csak adatokat tudnak tárolni, hanem az erőforrásokhoz is hozzá tudnak férni.

Itt most a `mac_bsextended(4)` és a `mac_seeotheruids(4)` modulokat vetjük be együttesen, és nem csak a rendszer objektumainak elérését tudjuk megakadályozni, hanem az egyes felhasználók futó programjait is elrejtjük.

A műveletet kezdjük azzal, hogy a `/boot/loader.conf` állományt kibővítjük a következő módon:

```
mac_seeotheruids_load="YES"
```

A `mac_bsextended(4)` biztonsági modul az alábbi `rc.conf`-változóval hozható működésbe:

```
ugidfw_enable="YES"
```

A hozzá tartozó alapértelmezett szabálykészlet az `/etc/rc.bsextended` állományban tárolódik, amely pedig a rendszer indítása során töltődik be. Ezeket némileg módosítanunk kell majd. Mivel a példában szereplő számítógép csak a felhasználók kiszolgálását hivatott ellátni, az utolsó kettő kivételével mindent hagyhatunk megjegyzésben. Így kikényszerítjük felhasználók által birtokolt rendszerobjektumok alapértelmezés szerinti betöltését.

Vegyük fel a szükséges felhasználókat a számítógépre és indítsuk újra. Tesztelési célból próbáljunk meg különböző felhasználókként bejelentkezni két konzolon. Futassuk le a `ps aux` parancsot, és így meg tudjuk figyelni, hogy mennyire látjuk a többi felhasználót. Amikor megpróbáljuk kiadni a `ls(1)` parancsot a többiek felhasználói könyvtáira, akkor hibát kell kapnunk.

Ne próbálgassunk a `root` felhasználóval, hacsak a megfelelő `sysctl` változóknak be nem állítottuk az ő hozzáféréseinek blokkolását is.

Megjegyzés: Amikor felveszük egy felhasználót a rendszerbe, a hozzá tartozó `mac_bsextended(4)` szabály nem fog szerepelni a szabályrendszerben. A szabályrendszer gyors frissítését úgy tudjuk megoldani, ha a `kldunload(8)` használatával egyszerűen eltávolítjuk a biztonsági modult a memóriából és újratöltjük a `kldload(8)` paranccsal.

16.17. A hibák elhárítása a MAC rendszerben

A fejlesztés fázisában bizonyos normál konfigurációval rendelkező felhasználók gondokat jeleztek. Ezeket foglaljuk most itt össze:

16.17.1. A `multilabel` beállítás nem adható meg a / állományrendszerre

A `multilabel` beállítás nem marad meg a rendszerindító (/) partíciómon!

A tapasztalatok szerint körülbelül minden ötvenedik felhasználó szembesül ezzel a problémával, és mi is találkozunk vele a kezdeti konfigurációk kialakítása során. Ennek az úgynevezett “hibának” a behatóbb tanulmányozása során arra jutottunk, hogy ez többnyire vagy a hibás dokumentálásból vagy a dokumentáció félreértelmezéséből ered. Független attól, hogy ez mitől is következett be, a következő lépések megtételével orvosolhatjuk:

1. Nyissuk meg az `/etc/fstab` állományt és adjuk meg a rendszerindító partíciónak az `ro`, vagyis az írásvédett (read-only) beállítást.
2. Indítsuk újra a gépet egyfelhasználós módban.
3. A `tunefs -l enable` parancsot futtassuk le a / állományrendszeren.
4. Indítsuk újra a rendszert normál módban.
5. Adjuk ki a `mount -urw /` parancsot, majd az `/etc/fstab` állományban írjuk át a `ro` beállítást az `rw` értékre és megint indítsuk újra a rendszert.
6. Alaposan nézzük át a `mount` parancs kimenetét és győződjünk meg róla, hogy a `multilabel` opció valóban beállítódott a rendszerindító állományrendszerre.

16.17.2. A MAC után nem lehet indítani az X11 szervert

Nem indul az X, miután MAC-kel kialakítottunk egy biztonságos környezetet!

Ezt vagy a MAC `partition` házirendje okozza, vagy az egyik címkéket használó házirend helytelen beállítása. A következő módon deríthetjük ki az okát:

1. Figyelmesen olvassuk el a hibaüzenetet: ha a felhasználó az `insecure` osztály tagja, akkor a `partition` házirend lesz a bűnös. Próbáljuk meg a felhasználót visszatenni a `default` osztályba és a `cap_mkdb` paranccsal újragenerálni az adatbázist. Ha ez nem segít a problémán, akkor haladjunk tovább.
2. Alaposan ellenőrizzük a címkékhez tartozó házirendeket. Vizsgáljuk meg, hogy a kérdéses felhasználó esetében a házirendet és az X11 alkalmazást, valamint a `/dev` eszközöket tényleg jól állítottuk be.
3. Ha az iméntiek egyik sem oldja meg gondunkat, küldjük el a hibaüzenetet és a környezetünk rövid leírását a a TrustedBSD (<http://www.TrustedBSD.org>) honlapjáról elérhető TrustedBSD levelezési lista vagy a FreeBSD general questions levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>) címére.

16.17.3. Hiba: `_secure_path(3) cannot stat .login_conf`

Amikor a rendszerben megpróbálok a `root` felhasználóról átváltani egy másik felhasználóra, a `_secure_path: unable to state .login_conf` hibaüzenet jelenik meg.

Ez az üzenet általában akkor látható, amikor a felhasználó nagyobb értékű címkével rendelkezik annál, mint akivé válni akar. Például vegyük a `joska` nevű felhasználót a rendszerben, aki az alap `biba/low` címkével rendelkezik. A `root` felhasználó, akinek `biba/high` címkéje van, nem láthatja `joska` felhasználói könyvtárát. Ez attól függetlenül megtörténik, hogy a `root` a `su` paranccsal váltott át a `joska` nevű felhasználóra vagy sem. Egy ilyen helyzetben a Biba sértetlenségi modellje nem fogja engedni a `root` felhasználóra számára, hogy láthassa a kevésbé sértetlen objektumokat.

16.17.4. A `root` felhasználó nem megy!

A rendszer normál vagy egyfelhasználós módban sem ismeri fel a `root` felhasználót. A `whoami` parancs 0 (nullát) ad vissza és a `su` parancs pedig annyit mond: `who are you?` (ki vagy?). Mi történhetett?

Ez csak olyankor történhet, ha a címkézési házirendet nem engedélyezzük, vagy a `sysctl(8)` használatával, vagy pedig a modul eltávolításával. Ha a házirendet letiltjuk vagy ideiglenesen letiltódik, akkor a bejelentkezési tulajdonságokat tároló adatbázist a `label` beállítás eltávolításával kell újrakonfigurálni. A `login.conf` állományból ne felejtjük el kivenni az összes `label` beállítást és a `cap_mkdb` paranccsal újragenerálni az adatbázist.

Ilyen akkor is előfordulhat, amikor a házirend valamilyen módon korlátozza a `master.passwd` állomány vagy adatbázis elérhetőségét. Ezt általában az okozza, hogy a rendszergazda az állományt olyan címke alatt módosítja, amely ütközik a rendszerben alkalmazott általános házirenddel. Ebben az esetekben a rendszer próbálja meg beolvasni a felhasználók adatait, azonban mivel közben az állomány új címkét örökölt, nem fér hozzá. Ha a `sysctl(8)` paranccsal letiltjuk a házirendet, minden vissza fog térni a rendes kerékvágásba.

Megjegyzések

1. Más feltételek mellett másmilyen hibák keletkezhetnek. Például, ha egy olyan objektumot próbálunk újracímkézni, amely nincs a felhasználó birtokában, esetleg nem is létezik vagy írásvédett. Adódhat, hogy a kötelező házirend az állomány, a program, vagy az új címkeérték tulajdonságai miatt nem fogja lehetővé tenni egy futó program számára egy állomány újracímkézését. Nézzük erre egy példát: egy kevésbé sértetlen felhasználó megpróbálja megváltoztatni egy sokkal sértetlenebb állomány címkéjét. Vagy egy kevésbé sértetlen felhasználó sokkal sértetlenebbre akarja állítani egy kevésbé sértetlen állomány címkéjét.

Fejezet 17. Biztonsági események vizsgálata

Írta: Tom Rhodes és Robert Watson.

17.1. Áttekintés

A FreeBSD támogatja a biztonsági események aprólékos vizsgálatát. Ezzel egy megbízható, részletes és jól konfigurálható naplózási rendszert nyújtanak a rendszerben található biztonságot igénylő események széles köréhez, beleértve a bejelentkezéseket, a konfigurációs állományokban bekövetkező változásokat, állomány- és hálózati hozzáféréseket. Az így létrehozott naplóbejegyzések felbecsülhetetlen értékűnek bizonyulhatnak egy élő rendszer felügyelete során, vagy egy hálózati támadás észleléséhez, esetleg egy összeomlás okainak kielemezéséhez. A FreeBSD ehhez a Sun által kifejlesztett BSM technológia API-ját és állományformátumát valósítja meg, és így képes együttműködni a Sun Solaris valamint az Apple Mac OS X biztonsági rendszereivel egyaránt.

Ebben a fejezetben a biztonsági események vizsgálatának telepítéséhez és beállításához szükséges ismeretek tekintjük át. Ennek keretében szó esik a vizsgálati házirendekről, valamint mutatunk egy példát a vizsgálatok beállítására.

A fejezet elolvasása során megismerjük:

- mit jelent az események vizsgálata és hogyan működik;
- hogyan kell beállítani az események vizsgálatát FreeBSD-n a különböző felhasználók és programok esetén;
- hogyan értelmezzük a vizsgálati nyomokat a vizsgálatot szűkítő és -elemző segédprogramok segítségével.

A fejezet elolvasásához ajánlott:

- alapvető UNIX-os és FreeBSD-s ismeretek (3 fejezet);
- a rendszermag konfigurálásával és fordításával kapcsolatos tudnivalók alapszintű ismerete (8 fejezet);
- az informatikai biztonság alapfogalmainak és annak a FreeBSD-re vonatkozó részleteinek minimális ismerete (14 fejezet).

FigyelemAz események vizsgálatával kapcsolatos ismert korlátozások: nem mindegyik biztonságot érintő esemény vizsgálható, mint például az egyes bejelentkezési típusok, mivel azok nem megfelelően hitelesítik a belépő felhasználókat. Ilyenek például az X11-alapú felületek és az egyéb, erre a célra alkalmas, más által fejlesztett démonok.

A biztonsági események vizsgálata során a rendszer képes nagyon részletes naplókat készíteni az érintett tevékenységekről. Így egy kellően forgalmas rendszeren az állománymozgások alapos nyomkövetése bizonyos konfigurációkon akár gigabyte-okat is kitehet hetente. A rendszergazdáknak ezért mindig javasolt számolniuk a nagy forgalmú események biztonsági vizsgálatának tárigényével. Például, emiatt érdemes lehet egy egész állományrendszert szánni erre a feladatra a `/var/audit` könyvtárban, és így a többi állományrendszer nem látja kárát, ha véletlenül betelne ez a terület.

17.2. A fejezet fontosabb fogalmai

A fejezet elolvasása előtt meg kell ismernünk néhány fontos alapfogalmat:

- *esemény*: Vizsgálható eseménynek azt az eseményt nevezzük, amely egy vizsgálati alrendszerben naplózható. Biztonsági események lehetnek például: egy állomány létrehozása, egy hálózati kapcsolat felépítése, vagy egy felhasználó bejelentkezése. Egy esemény “jellegzetes”, ha visszakövethető valamelyik hitelesített felhasználóhoz, vagy “nem jellegzetes”, ha ez nem lehetséges. Nem jellegzetes esemény lehet minden olyan esemény, amely egy bejelentkezési folyamat hitelesítési lépése előtt történik, például egy belépési kísérlet hibás jelszóval.
- *osztály*: Eseményosztálynak az összefüggő események névvel ellátott halmazát tekintjük, és szűrési feltételekben használjuk ezeket. Általában alkalmazott osztályok: “file creation” (fc, állománylétrehozás), “exec” (ex, programindítás), és “login_logout” (lo, ki- és bejelentkezés).
- *rekord*: Rekordnak nevezzük a biztonsági eseményeket leíró biztonsági naplóbejegyzéseket. A rekordok tartalmazhatják a feljegyzett esemény típusát, az eseményt kiváltó tevékenységet (felhasználót), a dátumot és az időt, tetszőleges objektum vagy paraméter értékét, feltételek teljesülését vagy meghiúsulását.
- *nyom*: Vizsgálati nyomnak vagy naplóállománynak nevezzük a különféle biztonsági eseményeket leíró vizsgálati rekordok sorozatát. A nyomok többnyire nagyjából az események bekövetkezése szerinti időrendben következnek. Csak és kizárólag az erre felhatalmazott programok hozhatnak létre rekordokat a vizsgálati nyomban.
- *szűrési feltétel*: Szűrési feltételnek nevezzük egy olyan karakterláncot, amelyet események szűrésére használunk, és módosítókat valamint eseményosztályok neveit tartalmazza.
- *előválogatás*: Előválogatásnak nevezzük a folyamatot, amelynek során a rendszer beazonosítja azokat az eseményeket, amelyek a rendszergazda számára fontosak. Ezáltal elkerülhetjük olyan vizsgálati rekordok generálását, amelyek számunkra érdektelen eseményekről számolnak be. Az előválogatás szűrési feltételek sorát használja az adott felhasználóhoz tartozó adott biztonsági események vizsgálatának beállításához, akárcsak a hitelesített és a nem hitelesített programokat érintő globális beállítások meghatározásához.
- *leszűkítés*: Leszűkítésnek nevezzük a folyamatot, amelynek során a már meglevő biztonsági rekordokból válogatunk le tárolásra, nyomtatásra vagy elemzésre. Hasonlóan ez a folyamat, ahol a szükségtelen rekordokat eltávolítjuk a vizsgálati nyomból. A leszűkítés segítségével a rendszergazdák a vizsgálati adatok eltávolítására alakíthatnak ki házirendet. Például a részletesebb vizsgálati nyomokat érdemes egy hónapig megtartani, ennek lejártával viszont már inkább ajánlott leszűkíteni ezeket és archiválásra csak a bejelentkezési információkat megtartani.

17.3. A vizsgálat támogatásának telepítése

A eseményvizsgálathoz szükséges felhasználói programok a FreeBSD alaprendszer részét képezik. Az eseményvizsgálat támogatása alapértelmezés szerint megtalálható a rendszermagban, azonban egy saját rendszermag esetén már külön be kell kapcsolnunk a megfelelő támogatást, mégpedig a rendszermag konfigurációs állományában az alábbi sor hozzáadásával:

```
options AUDIT
```

Fordítsuk és telepítsük újra a rendszermagot az 8 fejezetben ismertetett folyamat szerint.

Ahogy a rendszermagot a bekapcsolt eseményvizsgálati támogatással sikerült lefordítanunk és telepítenünk, valamint a rendszerünk is újraindult, indítsuk el a vizsgáló démont a következő sor hozzáadásával az rc.conf(5) állományban:

```
auditd_enable="YES"
```

A vizsgálatot inntentől ténylegesen egy ismételt újraindítással vagy pedig az előbb említett démon manuális elindításával aktiválhatjuk:

```
/etc/rc.d/auditd start
```

17.4. A vizsgálat beállítása

A vizsgálatok beállításához szükséges összes konfigurációs állomány a `/etc/security` könyvtárban található. A következő állományok vannak itt a démon indítása előtt:

- `audit_class` - a vizsgálati osztályok definícióit tartalmazza.
- `audit_control` - a vizsgálati alrendszer különböző területeit vezérli, többek közt az alapértelmezett vizsgálati osztályokat, az vizsgálati adatok tárhelyén fenntartandó minimális lemezterületet, a vizsgálati nyom maximális méretét, stb.
- `audit_event` - a rendszerben jelenlevő vizsgálati események szöveges megnevezése és leírása, valamint a lista, hogy melyikük mely osztályban található.
- `audit_user` - felhasználónként változó vizsgálati elvárások, kombinálva a bejelentkezéskor érvényes globálisan alapértelmezett beállításokkal.
- `audit_warn` - az **auditd** által használt testreszabható shell szkript, aminek segítségével a szélsőséges helyzetekben figyelmeztető üzeneteket tudunk generálni, mint például amikor a rekordok számára fenntartott hely hamarosan elfogy, vagy amikor a nyomokat tartalmazó állományt archiváltuk.

Figyelem Az eseményvizsgálat konfigurációs állományait alapos körültekintés mellett szabad szerkeszteni és karbantartani, mivel a bennük keletkező hibák az események helytelen naplózását eredményezhetik.

17.4.1. Eseményszűrés feltételek

Az eseményvizsgálati beállítások során számtalan helyen felbukkanak a vizsgálni kívánt eseményeket meghatározó szűrés feltételek. Ezen feltételek eseményosztályok felsorolását tartalmazzák, mindegyiküket egy módosító vezeti be, ezzel jelezve, hogy az adott eseményosztályba tartozó rekordokat tartsuk meg vagy vessük el. Esetleg utalhatnak arra is, hogy vagy csak a sikerességet jelző rekordokat, vagy csak a sikertelenséget jelző rekordokat szűrjük ki. A szűrés feltételek balról jobbra értékelődnek ki, és két kifejezés összefűzéssel kombinálható.

A most következő lista tartalmazza a `audit_class` állományban található alapértelmezett eseményvizsgálati osztályokat:

- `all` - *all (mind)* - Minden eseményosztályra vonatkozik.
- `ad` - *administrive (adminisztrációs)* - olyan adminisztrációs tevékenységek, amelyek egyben az egész rendszeren végrehajtódnak.
- `ap` - *application (alkalmazás)* - az alkalmazások által meghatározott tevékenység.
- `cl` - *file close (állomány lezárása)* - a `close` rendszerhívás meghívásának vizsgálata.

- `ex` - *exec* (programindítás) - egy program indításának vizsgálata. A parancssorban átadott paraméterek és a környezeti változók vizsgálatát az `audit_control(5)` vezérli a `policy` beállításhoz tartozó `argv` és `envv` paraméterek segítségével.
- `fa` - *file attribute access* (állományjellemzők hozzáférése) - a rendszerbeli objektumok jellemzőinek hozzáférésnek vizsgálata, mint például a `stat(1)`, `pathconf(2)` és ehhez hasonló események.
- `fc` - *file create* (állomány létrehozása) - állományt eredményező események vizsgálata.
- `fd` - *file delete* (állomány törlése) - állományt törölő események vizsgálata.
- `fm` - *file attribute modify* (állományjellemzők módosítása) - állományok jellemzőit megváltoztató események vizsgálata, mint például a `chown(8)`, `chflags(1)`, `flock(2)`, stb.
- `fr` - *file read* (állományolvasás) - állományok megnyitásával olvasásra, olvasásával, stb. kapcsolatos események vizsgálata.
- `fw` - *file write* (állományírás) - állományok megnyitásával írásra, írásával, módosításával, stb. kapcsolatos események vizsgálata.
- `io` - *ioctl* - az `ioctl(2)` rendszerhívást használó események vizsgálata.
- `ip` - *ipc* - a folyamatok közti kommunikáció különféle formáinak, beleértve a POSIX csövek és System V IPC műveleteinek vizsgálata.
- `lo` - *login_logout* (ki- és bejelentkezés) - a rendszerben megjelenő `login(1)` és `logout(1)` események vizsgálata.
- `na` - *non attributable* (nem jellegzetes) - a nem jellegzetes események vizsgálata.
- `no` - *invalid class* (érvénytelen osztály) - egyetlen biztonsági eseményt sem tartalmaz.
- `nt` - *network* (hálózat) - a hálózathoz tartozó események vizsgálata, mint például a `connect(2)` és az `accept(2)`.
- `ot` - *other* (egyéb) - más egyéb események vizsgálata.
- `pc` - *process* (folyamat) - a folyamatokkal kapcsolatos műveletek, mint például az `exec(3)` és az `exit(3)` vizsgálata.

Az imént felsorolt eseményosztályok az `audit_class` és az `audit_event` állományok módosításával igény szerint testreszabhatóak.

A listában szereplő minden egyes eseményosztályhoz tartozik még egy módosító is, amely jelzi, hogy a sikeres vagy a sikertelen műveleteket kell-e szűrniük, valamint hogy a bejegyzés az adott típust vagy osztályt hozzáadja vagy elveszi az adott szűrésből.

- (üres) az adott típusból mind a sikereseket és mind a sikerteleneket feljegyzi.
- `+` az eseményosztályba tartozó sikeres eseményeket vizsgálja csak.
- `-` az eseményosztályba tartozó sikertelen eseményeket vizsgálja csak.
- `^` az eseményosztályból sem a sikereseket, sem pedig a sikerteleneket nem vizsgálja.
- `^+` az eseményosztályból nem vizsgálja a sikeres eseményeket.
- `^-` az eseményosztályból nem vizsgálja a sikertelen eseményeket.

Az alábbi példa egy olyan szűrési feltételt mutat be, amely a ki- és bejelentkezések közül megadja a sikereset és a sikerteleneket, viszont a programindítások közül csak a sikereseket:

```
lo,+ex
```

17.4.2. A konfigurációs állományok

A vizsgálati rendszer beállításához az esetek túlnyomó részében a rendszergazdának csupán két állományt kell módosítaniuk: ezek az `audit_control` és az `audit_user`. Az előbbi felelős a rendszerszintű vizsgálati jellemzőkért és házirendekért, míg az utóbbi az igények felhasználókénti finomhangolásához használható.

17.4.2.1. Az `audit_control` állomány

Az `audit_control` állomány határozza meg a vizsgálati alrendszer alapértelmezéseit. Ezt az állományt megnyitva a következőket láthatjuk:

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo
policy:cnt
filesz:0
```

A `dir` opciót használjuk a vizsgálati naplók tárolására szolgáló egy vagy több könyvtár megadására. Ha egynél több könyvtárra vonatkozó bejegyzés található az állományban, akkor azok a megadás sorrendjében kerülnek feltöltésre. Nagyon gyakori az a beállítás, ahol a vizsgálati naplókat egy erre a célra külön kialakított állományrendszeren tárolják, megelőzve ezzel az állományrendszer betelésekor keletkező problémákat a többi alrendszerben.

A `flags` mező egy rendszerszintű alapértelmezett előválogatási maszkot határoz meg a jellegzetes események számára. A fenti példában a sikeres és sikertelen ki- és bejelentkezéseket mindegyik felhasználó esetén vizsgáljuk.

A `minfree` opció megszabja a vizsgálati nyom tárolására szánt állományrendszeren a minimális szabad helyet, a teljes kapacitás százalékában. Amint ezt a küszöböt túllépjük, egy figyelmeztetés fog generálódni. A fenti példa a minimálisan szükséges rendelkezésre álló helyet húsz százalékra állítja.

A `naflags` opció megadja azokat az eseményosztályokat, amelyeket vizsgálni kell a nem jellegzetes események, mind például a bejelentkezési folyamatok vagy rendszerdémonok esetén.

A `policy` opció a vizsgálat különböző szempontjait irányító házirendbeli beállítások vesszővel elválasztott listáját tartalmazza. Az alapértelmezett `cnt` beállítás azt adja meg, hogy a rendszer a felmerülő vizsgálati hibák ellenére is folytassa tovább a működését (erősen javasolt a használata). A másik gyakorta alkalmazott beállítás az `argv`, amellyel a rendszer a parancsvégrehajtás részeként az `execve(2)` rendszerhívás parancssori paramétereit is megvizsgálja.

A `filesz` opció határozza meg a vizsgálati nyom automatikus szétvágása és archiválása előtti maximális méretét, byte-ban. Az alapértelmezett értéke a 0, amely kikapcsolja ezt az archiválást. Ha az itt megadott állományméret nem nulla és a minimálisan elvárt 512 KB alatt van, akkor a rendszer figyelmen kívül hagyja és erről egy figyelmeztetést ad.

17.4.2.2. Az `audit_user` állomány

Az `audit_user` állomány lehetővé teszi a rendszergazda számára, hogy az egyes felhasználók számára további vizsgálati szigorításokat határozzon meg. Minden sor egy-egy felhasználó vizsgálatának pontosítását adja meg két mező segítségével: az első közülük az `alwaysaudit` mező, mely felsorolja azokat az eseményeket, amelyeket minden esetben vizsgálni kell az adott felhasználó esetén, valamint a második a `neveraudit` mező, mely az adott felhasználó esetén a nem vizsgálandó eseményeket adja meg.

A most következő `audit_user` példában vizsgáljuk a `root` felhasználó ki- és bejelentkezéseit és sikeres programindításait, valamint a `www` felhasználó állománylétrehozásait és sikeres programindításait. Ha a korábban bemutatott `audit_control` példával együtt használjuk, akkor észrevehetjük, hogy a `lo` bejegyzés a `root` felhasználó esetén redundáns, illetve ilyenkor a ki/bejelentkezést a `www` felhasználó esetén is vizsgáljuk.

```
root:lo,+ex:no
www:fc,+ex:no
```

17.5. A vizsgálati alrendszer használata

17.5.1. A vizsgálati nyomok megtekintése

A vizsgálati nyomok a BSM bináris formátumban tárolódnak, ezért a tartalmának konvertálásához és módosításához külön segédprogramokra van szükség. A `praudit(1)` parancs a nyomállományokat egyszerű szöveges formátumra alakítja, az `auditreduce(1)` parancs pedig a nyomok elemzéséhez, archiválásához vagy nyomtatásához szükséges leszűkítéseket végzi el. Az `auditreduce` a szűrési feltételek paramétereinek széles skáláját kezeli, beleértve az eseménytípusokat, -osztályokat, felhasználókat, események dátumát vagy időpontját, állományok elérési útját vagy az általuk érintett objektumokat.

Például a `praudit` segédprogram képes kilistázni szövegesen egy adott vizsgálati napló teljes tartalmát:

```
# praudit /var/audit/AUDITFILE
```

ahol az `AUDITFILE` a kiírandó vizsgálati napló.

A vizsgálati nyomok tokenekből összeállított vizsgálati rekordok, amelyeket a `praudit` egymás után soronként megjelenít. Minden token adott típusú, például a `header` egy vizsgálati rekord fejlécét tartalmazza, vagy a `path`, amely a névfeloldásból származó elérési utat tartalmaz. A következő példa egy `execve` eseményt mutat be:

```
header,133,10,execve(2),0,Mon Sep 25 15:58:03 2006, + 384 msec
exec_arg,finger,doug
path,/usr/bin/finger
attribute,555,root,wheel,90,24918,104944
subject,robert,root,wheel,root,wheel,38439,38032,42086,128.232.9.100
return,success,0
trailer,133
```

Ez a vizsgálat egy sikeres `execve` hívást rögzít, ahol a `finger doug` parancs futott le. A paramétereket tartalmazó token magában foglalja a shell által a rendszermag felé jelzett parancsot és annak paraméterét egyaránt. A `path` token tárolja a végrehajtott állomány rendszermag által feloldott elérési útját. A `attribute` token erről a binárisról ad további információkat, különösen az állomány módjáról, amely segít megállapítani, hogy az adott alkalmazásnál be volt-e állítva a `setuid` bit. A `subject` token leírja az érintett folyamatot és rendre megjegyzi a vizsgált felhasználó azonosítóját, az aktuálisan érvényben levő felhasználó és csoport azonosítóját, a valós felhasználói és csoport azonosítót, a folyamat azonosítóját, a munkamenet azonosítóját, a port azonosítóját és a bejelentkezéshez használt hálózati címet. Vegyük észre, hogy a vizsgált felhasználó azonosítója és a valódi azonosítója eltér egymástól: a `robert` nevű felhasználó a `root` accountjára váltott a parancs futtatása előtt, de az eredetileg hitelesített felhasználójaként lett vizsgálva. Végezetül a `return` token jelzi a sikeres végrehajtást, és a `trailer` pedig zárja a rekordot.

17.5.2. A vizsgálati nyomok leszűkítése

Mivel a vizsgálatokhoz tartozó naplók akár egészen nagyok is lehetnek, ezért a rendszergazdának minden bizonnyal szüksége lehet a számára fontos, például egy adott felhasználóhoz tartozó rekordok kiválogatására:

```
# auditreduce -u trhodes /var/audit/AUDITFILE | praudit
```

Ezzel ki tudjuk szűrni a `trhodes` nevű felhasználóhoz tartozó összes vizsgálati rekordot az `AUDITFILE` állományból.

17.5.3. A naplók megtekintéséhez szükséges jogok továbbadása

Az `audit` csoport tagjai olvashatják a `/var/audit` könyvtárban található vizsgálati nyomokat. Alapértelmezés szerint ez a csoport üres, ezért csak a `root` képes ekkor vizsgálni a nyomokat. A többi felhasználó számára úgy tudunk olvasási jogot biztosítani, ha felvesszük őket az `audit` csoportba. Mivel a vizsgálati naplók tartalmának figyelése jelentős rálátást adhat a rendszerben jelenlevő felhasználók és folyamatok viselkedésére, ajánlott körültekintően kiosztani az olvasási jogokat.

17.5.4. Élő rendszerfelügyelet a vizsgálati csövekkel

A vizsgálati csövek az eszközök állományabsztrakcióit klónozzák le, és ezzel teszik lehetővé az alkalmazások számára, hogy menet közben megcsapolhassák a megfigyelt eszközök adatait. Ez az elsődleges célja a különböző betörésfigyelő és rendszerfelügyeleti eszközök készítőinek. A rendszergazda számára azonban a vizsgálati csövek megkönnyítik az élő megfigyelést, mert itt nem merülnek fel a nyomok jogosultságaiból vagy az archiválás miatt megszakadó eseményfolyamokból adódó problémák. Az élő eseményfolyamra az alábbi parancs kiadásával lehet rácsatlakozni:

```
# praudit /dev/auditpipe
```

Alapértelmezés szerint a vizsgálati csőhöz tartozó csomópontok kizárólag csak a `root` felhasználó részére érhetőek el. Az `audit` csoport tagjai úgy tudnak majd hozzáférni, ha felvesszük a következő `devfs` szabályt a `devfs.rules` állományba:

```
add path 'auditpipe*' mode 0440 group audit
```

A `devfs` állományrendszer beállításáról bővebben lásd a `devfs.rules(5)` oldalt.

Figyelem Könnyen gerjedést lehet előidézni a vizsgált események megfigyelésével, amikor is az egyes események megtekintése újabb vizsgálandó események sorozatát indítják el. Például, ha az összes hálózati forgalmat egyszerre vizsgáljuk és a `praudit(1)` egy SSH-munkameneten keresztül fut, akkor a vizsgálati események töméntelen áradata indul meg, mivel minden kiírandó esemény egy újabb eseményt indukál. Ennek elkerülése érdekében ajánlott a `praudit` parancsot részletes forgalmat nem figyelő vizsgálati csővel ellátott munkameneten keresztül elindítani.

17.5.5. A vizsgálati nyomok archiválása

A vizsgálati nyomokat egyedül a rendszermag képes írni, illetve csak a vizsgálati démon, az **auditd** képes felügyelni. A rendszergazdának ebben az esetben tehát nem szabad használniuk a `newsyslog.conf(5)` vagy a hozzá hasonló

eszközök használatát a vizsgálati naplók archiválásához. Helyettük a `audit` segédprogramot javasolt használni a vizsgálatok leállítására, a vizsgálati rendszer újrakonfigurálására vagy a napló archiválásának elvégzésére. Az alábbi parancs utasítja a vizsgálati démon, hogy hozzon létre egy új vizsgálati naplót és jelzi a rendszermagnak, hogy váltson erre az új naplóra. Az eddig használt naplót lezárja és átnevezi, ami ezután a rendszergazda által tetszőlegesen feldolgozható.

```
# audit -n
```

Figyelem Ha az `auditd` démon a parancs kiadásának pillanatában nem futna, akkor hiba történik és erről hibaüzenetet kapunk.

A `cron(8)` segítségével tizenként óránként kikényszeríthetjük a naplók váltását, ha felvesszük a `/etc/crontab` állományba az alábbi sort:

```
0 * /12 * * * root /usr/sbin/audit -n
```

Ez a változtatás akkor fog érvénybe lépni, ha elmentjük az új `/etc/crontab` állományt.

A vizsgálati nyomok mérete szerinti automatikus váltás is megvalósítható az `audit_control(5)` állományban szereplő `filesz` opció beállításával, amit meg is találhatunk ebben a fejezetben, a konfigurációs állományok beállításánál.

17.5.6. A vizsgálati nyomok tömörítése

Mivel a vizsgálati nyomok óriásira is megnőhetnek, sokszor felmerül az igény, hogy lehessen őket tömöríteni vagy más egyéb módon archiválni a vizsgálati démon által lezárt nyomokat. Az `audit_warn` szkript használható a különböző vizsgálatokhoz kapcsolódó események esetén elvégzendő műveletek megadásához, beleértve ebbe a vizsgálati nyomok váltásakor elvégzett szabályos lezárását. Például a következőket kell beleírnunk az `audit_warn` szkriptbe a nyomok lezárását követő tömörítéséhez:

```
#
# Lezáráskor tömöríti a vizsgálati nyomot.
#
if [ "$1" = closefile ]; then
    gzip -9 $2
fi
```

Egyéb archiválási tevékenységek lehetnek még: a nyomok felmásolása egy központi szerverre, a régebbi nyomok törlése, vagy a meglevő nyomok leszűkítése csak a fontos információkra. A szkript csak akkor fog lefutni, ha a vizsgálati nyomot sikerült szabályosan lezárni, így tehát a szabálytalan leálláskor megmaradó nyomok esetén nem.

A FreeBSD 6.3 és későbbi verzióiban, a `praudit` XML kimeneti formátumot is támogat, amely az `-x` kapcsolóval érhető el.

Fejezet 18. Háttértárak

18.1. Áttekintés

Ez a fejezet arról szól, hogy miként használjuk a lemezeinket a FreeBSD-vel. Itt többek közt szó esik a memória (alapú) lemezekről, a hálózaton keresztül csatlakoztatott meghajtókról, a szabványos SCSI/IDE tárolóeszközökről és az USB felületet használó eszközökről.

A fejezet elolvasása során megismerjük:

- a FreeBSD által alkalmazott terminológiát, amivel a fizikai lemezeken elhelyezkedő adatokat írja le (partíciók és slice-ok);
- hogyan bővítjük rendszerünket további merevlemezekkel;
- hogyan állítsuk be a FreeBSD-t USB tárolóeszközök használatára;
- hogyan állítsunk be virtuális állományrendszereket, például memórialemezeket;
- hogyan használjuk a kvótákat a lemezterület használatának korlátozására;
- hogyan védjük meg lemezeinket titkosítással az illetéktelenektől;
- FreeBSD alatt hogyan készítsünk és írjuk CD-ket, DVD-ket;
- a biztonsági mentések készítésének különböző lehetőségeit;
- hogyan használjuk a FreeBSD alatt rendelkezésünkre álló, biztonsági mentést készítő programokat;
- hogyan mentünk floppy lemezekre;
- mik az állományrendszerek pillanatképei és hogyan kell ezeket hatékonyan használni.

A fejezet elolvasásához ajánlott:

- a FreeBSD rendszermag beállításának és telepítésének ismerete (8 fejezet)

18.2. Az eszközök elnevezései

A most következő listában felsoroljuk a FreeBSD által ismert fizikai tárolóeszközöket és a hozzájuk tartozó elnevezéseket.

Táblázat 18-1. A fizikai lemezek elnevezésének szabályai

A meghajtó típusa	A meghajtóeszköz neve
IDE merevlemez	ad
IDE CD-meghajtók	acd
SCSI merevlemez és USB tárolóeszközök	da
SCSI CD-meghajtók	cd
Különböző nem szabványos CD-meghajtók	mcd (Mitsumi CD-ROM) és scd (Sony CD-ROM)
Floppy meghajtók	fd

A meghajtó típusa

SCSI szalagos meghajtók

IDE szalagos meghajtók

Flash meghajtó

RAID meghajtók

A meghajtóeszköz neve

sa

ast

fla (DiskOnChip® Flash eszköz)

aacd (Adaptec AdvancedRAID), mlxds és mlyd (Mylex), amrd (AMI MegaRAID), idad (Compaq Smart RAID), twed (3ware® RAID).

18.3. Lemezek hozzáadása

Eredetileg írta: David O'Brien.

Ebben a szakaszban arról lesz szó, hogy a jelenleg egyetlen meghajtót tartalmazó rendszerünket hogyan tudjuk bővíteni egy új SCSI-lemez hozzáadásával. Ehhez elsőként kapcsoljuk ki a számítógépünket és szereljük be a helyére az új meghajtót a számítógép, a lemezvezérlő és a meghajtó gyártójának utasításai alapján. Mivel ezt a műveletet rengeteg módon lehet elvégezni, ezért ennek pontos részleteivel ez a leírás most nem foglalkozik.

Jelentkezzünk be `root` felhasználóként. Miután beszereltük a meghajtót, a `/var/run/dmesg.boot` állomány végignézésével bizonyosodjunk meg róla, hogy a rendszer valóban megtalálta a lemezt. A példánk szerint ez a meghajtó tehát a `da1` nevet fogja viselni, amelyet a `/1` könyvtárba akarunk csatlakoztatni (ha IDE-meghajtót telepítettünk, akkor a hozzá tartozó eszköz neve `ad1` lesz).

Mivel a FreeBSD IBM PC kompatibilis számítógépeken fut, ezért nem szabad figyelmen kívül hagynunk a PC BIOS partícióit is. Ezek eltérnek a hagyományos BSD partícióktól. Egy PC-s lemeznek négy BIOS-os partícióbejegyzése lehet. Ha egy lemezt tényleg csak a FreeBSD-nek szánunk, akkor használhatjuk az ún. *dedikált* módot. Minden más esetben a FreeBSD-nek egy PC BIOS partícióban kell elhelyezkednie. A FreeBSD a PC BIOS partícióit *slice*-nak nevezi, ezzel különbözteti ezeket a hagyományos BSD partícióktól. Dedikált esetekben is használhatjuk, de elsősorban akkor kap fontosabb szerepet, amikor a FreeBSD-nek más operációs rendszerekkel kell megosztani a helyet. Ezzel el tudjuk kerülni, hogy a más operációs rendszerekben megtalálható, nem FreeBSD alapú `fdisk` parancs megzavarodjon.

A slice-ok használatakor a meghajtó `/dev/dals1e` néven kerül hozzáadásra. Így kell olvasni: egyes SCSI lemezes egység (második SCSI lemez), első slice (első PC BIOS partíció) és e BSD partíció. A dedikált esetben a meghajtó neve viszont egyszerűen csak `/dev/dale`.

Mivel a `bsdlable(8)` 32 bites egész számokat használ a szektorok számának tárolására, ezért lemezenként csak $2^{32}-1$ szektort tud ábrázolni, ami az esetek többségében 2 TB méretű címezhető területet jelent. Az `fdisk(8)` formátuma szerint sem a kezdőszektor, sem a hossz nem lehet $2^{32}-1$ -nél több, amivel így a partíciókat 2 TB, a lemezeket pedig 4 TB méretűre korlátozza. A `sunlabel(8)` formátuma partíciónként $2^{32}-1$ szektort enged meg és összesen 8 partíciót, amely ezáltal 16 TB terület lefedését teszi lehetővé. Nagyobb lemezekhez `gpt(8)` partíciók használatosak.

18.3.1. A `sysinstall(8)` használatával

1. Közlekedés a `sysinstall` programban

A `sysinstall` könnyen használható menüinek segítségével az új lemezen pillanatok alatt létre tudunk hozni partíciókat és megcímkézni ezeket. Ehhez vagy `root` felhasználóként jelentkezzünk be a rendszerbe, vagy adjuk

ki a `su` parancsot. A `sysinstall` parancs kiadása után lépünk be a `Configure` (Beállítások) menübe. A `FreeBSD Configuration Menu` menüben ezután keressük meg és válasszuk ki az `Fdisk` menüpontot.

2. Az **fdisk** partíciószerkesztő

Miután eljutottunk az **fdisk** alkalmazáshoz, az **A** lenyomásával felajánlhatjuk az egész lemezt a FreeBSD számára. Amikor előkerül a kérdés, hogy “remain cooperative with any future possible operating systems” (“működőképes maradjon-e a későbbiekben telepítendő operációs rendszerekkel”), akkor válaszoljunk rá **YES**-szel (tehát igen). A **W** gomb lenyomásával írjuk a lemezre a most elvégzett változtatásokat. Ezután már a **Q** használatával ki is léphetünk az `FDISK` szerkesztőből. A következő lépésben a “Master Boot Record”-ról fognak minket megkérdezni. Mivel most egy már működő rendszert bővítünk, ezért a válaszuk erre `None` lesz.

3. A lemezcímkék szerkesztése

Most lépünk ki a **sysinstall** alkalmazásból és indítsuk el újra. Kövessük az iménti útmutatásokat, de ezúttal a `Label` menüpontot válasszuk ki. Ezzel a `Disk Label Editor`-ba vagyis a lemezcímkék szerkesztőjéhez jutunk. Itt fogjuk létrehozni a hagyományos BSD partíciókat. Egy lemezen nyolc ilyen partíció lehet, a-tól h-ig. Közülük néhány partíció címkéjét megkülönböztetjük. Az a partíció jelöli a rendszer indításához használt partíciót, a gyökérpartíciót (/). Tehát a partíció csak a rendszerlemezünkön szerepelhet (tehát ahonnan indul a rendszer). A b partíció a lapozáshoz használt partíciókat jelöli és több lemezen is szerepelhet. A c partíción keresztül lehet elérni az egész lemezt dedikált módban vagy az egész FreeBSD slice-ot slice módban. A többi partíció tetszőlegesen felhasználható.

A **sysinstall** címkeszerkesztője az e betűvel szereti megjelölni a sem nem rendszerindító, sem nem lapozó partíciókat. A címkeszerkesztőben egyetlen állományrendszert a **C** lenyomásával lehet készíteni. Amikor erre válaszul megkérdezi a típusát (FS (állományrendszer) vagy swap (lapozóterület) legyen), akkor válasszuk az `FS` beállítást és adjuk meg a csatlakozási pontját (például `/mnt`). Amikor a lemezt telepítés után (post-install) adjuk hozzá, akkor a **sysinstall** valójában nem hoz létre hozzá bejegyzéseket az `/etc/fstab` állományban, ezért a csatlakozási pont megadása nem is feltétlenül fontos.

Most már készen állunk arra, hogy rögzítsük az új címkét a lemezre és létrehozunk vele egy állományrendszert. Ehhez nyomjuk le a **W** gombot. Ne foglalkozunk vele, ha a **sysinstall** nem képes csatlakoztatni az új partíciót. Ha ezzel megvagyunk, akkor lépünk ki a címkeszerkesztőből és a **sysinstall**ból is.

4. Befejezés

Most már csak annyi teendőnk maradt, hogy felvegyük az `/etc/fstab` állományba az új lemezhez tartozó bejegyzést.

18.3.2. Parancssoros eszközök használatával

18.3.2.1. Slice módban

Ezzel a beállítással a lemezünkre később más operációs rendszereket is telepíthetünk, és nem okoz gondot a saját `fdisk` segédprogramjaik működésében. Az új lemezek telepítésénél ezt a módszer ajánlatos követni. A dedikált módot viszont csak abban az esetben használjuk, ha erre nyomós okunk van!

```
# dd if=/dev/zero of=/dev/dal bs=1k count=1
# fdisk -BI dal # inicializáljuk az új lemezt
# bsdlabel -B -w dals1 auto # címkézzük meg
# bsdlabel -e dals1 # szerkesztjük át a frissen létrehozott címkét és vegyünk fel egy új partíciót
# mkdir -p /1
```

```
# newfs /dev/dals1e # ismételjük meg minden létrehozott partícióhoz
# mount /dev/dals1e /1 # csatlakoztassuk a partíció(ka)t
# vi /etc/fstab # vegyük fel a megfelelő bejegyzés(ek)e)t az /etc/fstab állományba
```

IDE-lemezek esetén az ad eszközt a da eszközzel helyettesítsük.

18.3.2.2. Dedikált módban

Amennyiben az új meghajtót nem akarjuk megosztani egyetlen más operációs rendszerrel sem, használhatjuk a dedicated (dedikált) módot. Ne felejtjük el azonban, hogy ez képes összezavarni a Microsoft operációs rendszereit, habár ebből semmilyen kárunk nem fog származni. Az IBM OS/2 operációs rendszere azonban “kisajátít” minden olyan partíciót, amelyet nem tud olvasni.

```
# dd if=/dev/zero of=/dev/dal bs=1k count=1
# bsdlable -Bw dal auto
# bsdlable -e dal # létrehozzuk az 'e' partíciót
# newfs /dev/dale
# mkdir -p /1
# vi /etc/fstab # felvesszük a /dev/dale partíciót
# mount /1
```

Egy másik megoldás:

```
# dd if=/dev/zero of=/dev/dal count=2
# bsdlable /dev/dal | bsdlable -BR dal /dev/stdin
# newfs /dev/dale
# mkdir -p /1
# vi /etc/fstab # felvesszük a /dev/dale partíciót
# mount /1
```

18.4. RAID

18.4.1. Szoftveres RAID

18.4.1.1. Összefűzött lemezek beállítása

Eredetileg készítette: Christopher Shumway. Ellenőrizte: Jim Brown.

A nagyobb méretű háttértárolók kiválasztásánál a legfontosabb tényezők a sebesség, megbízhatóság és a költség. Nagyon ritkán lehet csak ezt a hármat egyensúlyba hozni: általában a gyors és megbízható tárolóeszközök sok pénzbe kerülnek, valamint a költségek megtakarításához vagy a sebességet vagy pedig a megbízhatóságot kell feláldoznunk.

A továbbiakban egy olyan rendszert mutatunk be, ahol a elsősorban a költségek, majd csak ezután a sebesség és megbízhatóság kerültek előtérben. A rendszer adatátviteli sebességét a hálózat korlátozza. Habár emellett a megbízhatóság is nagyon fontos, a tárgyalat összefűzött meghajtó (Concatenated Disk, CCD) csak adatokat szolgáltat és a teljes tartalma bármikor visszaállítható, mivel rendelkezésre áll CD-n.

A feladat elvégzésére alkalmas háttértároló kiválasztásában elsőként a saját elvárásainkat kell tudnunk megfogalmazni. Ha nekünk jobban számít az árnál a sebesség vagy a megbízhatóság, akkor a mostaniaktól némileg eltérő konfigurációt kell majd építenünk.

18.4.1.1.1. A hardver telepítése

A rendszert tartalmazó IDE-lemez mellett három darab, egyenként 30 GB-os 5400-as percenkénti fordulatszámú Western Digital gyártmányú merevlemez alkotja majd a létrehozni kívánt, kb. 90 GB összméretű összefűzött lemezt. Ideális esetben minden IDE-lemez saját külön vezérlőn és kábelben van, de a költségek csökkentése miatt nem használtunk további IDE-vezérlőket. Ehelyett inkább jumperekkel úgy állítottuk be a lemezeket, hogy minden vezérlőre egy mester (master) és egy szolga (slave) módú merevlemez kapcsolódjon.

A beszerelés után beállítottuk a rendszer BIOS-át, hogy automatikusan felismerje a csatlakoztatott lemezeket. De ami még fontosabb, hogy a FreeBSD is észlelte ezeket az indítás során:

```
ad0: 19574MB <WDC WD205BA> [39770/16/63] at ata0-master UDMA33
ad1: 29333MB <WDC WD307AA> [59598/16/63] at ata0-slave UDMA33
ad2: 29333MB <WDC WD307AA> [59598/16/63] at ata1-master UDMA33
ad3: 29333MB <WDC WD307AA> [59598/16/63] at ata1-slave UDMA33
```

Megjegyzés: Ha a FreeBSD nem látná az összes lemezt, akkor ellenőrizzük a jumperek helyes beállítását. Napjainkban a legtöbb IDE-meghajtón találunk egy "Cable Select" jumpert is. Ezzel *nem* a mester/szolga módot állítjuk be! A megfelelő jumper beazonosításához olvassuk el a meghajtóhoz tartozó dokumentációt.

A következő lépésben azt vesszük nagytű alá, hogyan lehet ezeket az állományrendszer részévé tenni. Ezzel kapcsolatban a vinum(8) (21 fejezet) és a ccd(4) elolvasása ajánlatos. Erre a célra itt most a ccd(4) használatát választottuk.

18.4.1.1.2. A CCD beállítása

A ccd(4) meghajtó segítségével több ugyanolyan lemezt tudunk összefűzni egyetlen logikai állományrendszerré. A ccd(4) használatához arra is szükségünk van, hogy a ccd(4) támogatása jelen legyen a rendszermagban. A következő sor tegyük bele a rendszermag konfigurációs állományába, fordítsuk újra és telepítsük a rendszermagot:

```
device    ccd
```

A ccd(4) támogatása modulként is betölthető.

A ccd(4) beállításához először a bsdlablel(8) programmal meg fel kell címkéznünk a lemezeket:

```
bsdlablel -w ad1 auto
bsdlablel -w ad2 auto
bsdlablel -w ad3 auto
```

Így létrejön egy-egy BSD típusú címke a ad1c, ad2c és ad3c eszközökre, amely így lefedi a lemez egész területét.

Most pedig változtassuk meg a lemezcímke típusát. Ehhez használjuk ismét a bsdlablel(8) programot:

```
bsdlablel -e ad1
bsdlablel -e ad2
bsdlablel -e ad3
```

Az EDITOR környezeti változóban megadott szövegszerkesztővel (ez általában a vi(1)) megnyílik minden egyes lemezhez a jelenlegi lemezcímke.

Egy módosítatlan lemezcímke valahogy így néz ki:

```
8 partitions:
#          size      offset      fstype    [fsize bsize bps/cpg]
  c: 60074784         0      unused          0      0      0 # (Cyl.    0 - 59597)
```

A ccd(4) számára hozzunk létre egy új e partíciót. Ezt lényegében a c partíció lemásolásával keletkezik, de nála az fstype (az állományrendszer típusa) oszlopban mindenképpen **4.2BSD** szerepeljen! A lemezcímke most már valahogy így fog kinézni:

```
8 partitions:
#          size      offset      fstype    [fsize bsize bps/cpg]
  c: 60074784         0      unused          0      0      0 # (Cyl.    0 - 59597)
  e: 60074784         0      4.2BSD          0      0      0 # (Cyl.    0 - 59597)
```

18.4.1.1.3. Az állományrendszer kiépítése

Most, miután felcímkeztük az összes lemezünket, lássunk neki a ccd(4) kiépítésének. Ezt a ccdconfig(8) meghívásával és az alábbihoz hasonló paraméterek átadásával tehetjük meg:

```
ccdconfig ccd0❶ 32❷ 0❸ /dev/ad1e❹ /dev/ad2e /dev/ad3e
```

A paraméterek rövid leírása és használata:

- ❶ Az első paraméter a létrehozandó eszköz, ami jelen esetünkben a /dev/ccd0c. A /dev/ részt nem kötelező megadni.
- ❷ A kihagyás nagysága az állományrendszerben. A kihagyás határozza meg a lemezblokkban alkalmazott csíkozás (striping) vastagságát, ami általában 512 byte. Ennek megfelelően a 32-es kihagyás 16 384 byte-os csíkokat ad meg.
- ❸ A ccdconfig(8) beállításai. Ha engedélyezni akarjuk a lemezek tükrözését, akkor itt megadhatjuk. Mivel ez a konfiguráció most nem nyújt tükrözést a ccd(4) számára, ezért állítsuk nullára (0).
- ❹ A ccdconfig(8) parancsnak utolsóként azokat az eszközöket kell felsorolni, amelyeket tömbbe akarunk fűzni. Minden eszközt teljes elérési úttal adjuk meg.

A ccdconfig(8) futtatása után a ccd(4) beállítódik. Most már állományrendszert is rakhatunk rá. A newfs(8) man oldalról szedjük össze a szükséges paraméterezést, vagy egyszerűen csak gépeljük be ennyit:

```
newfs /dev/ccd0c
```

18.4.1.1.4. Az egész önműködővé tétele

A ccd(4) eszközt általában minden egyes indítás után használni akarjuk. Ennek eléréséhez először ezt be kell állítanunk. Az alábbi parancs kiadásával írássuk be a jelenlegi beállításainkat tükröző /etc/ccd.conf állományt:

```
ccdconfig -g > /etc/ccd.conf
```

Az újraindítás során az `/etc/rc` parancs futtatja le a `ccdconfig -C` parancsot, ha az `/etc/ccd.conf` állomány létezik. Ez automatikusan beállítja a ccd(4) eszközöket, így ilyenkor tudjuk csatlakoztatni is ezeket.

Megjegyzés: Ha egyfelhasználós módban indítjuk a rendszert, mielőtt még a `mount(8)` paranccsal csatlakoztatni tudnánk a ccd(4) eszközt, a tömb beállításához meg kell hívnunk a következő parancsot:

```
ccdconfig -C
```

Ha a rendszerindításkor automatikusan csatlakoztatni akarjuk a ccd(4) eszközt, akkor az `/etc/fstab` állományba helyezzünk el egy hozzá tartozó bejegyzést:

```
/dev/ccd0c          /media      ufs      rw      2      2
```

18.4.1.2. A Vinum kötetkezelő

A Vinum kötetkezelő egy blokkos eszközmeghajtó, ami virtuális lemezes meghajtókat valósít meg. Elkülöníti a lemezes hardvereszközöket a blokkos eszközmeghajtók felületétől és a kettő között úgy képezi le az adatokat, hogy a hagyományos lemezes tárolással szemben megnövekedett rugalmasságot, teljesítményt és megbízhatóságot kapunk. A `vinum(8)` ismeri a RAID-0, RAID-1 és RAID-5 modelleket egyaránt, melyeket önmagukban és együttesen kombinálva is használhatunk.

A 21 fejezet bővebben ismerteti a `vinum(8)` rendszerét.

18.4.2. Hardveres RAID

A FreeBSD rengeteg különböző típusú hardveres RAID-vezérlőt ismer. Ezek az eszközök a FreeBSD külön erre a célra szánt támogatása nélkül képesek vezérelni a RAID-alrendszert.

A rajta levő BIOS segítségével a kártya a legtöbb lemezműveletet egyedül kezeli. A következőkben egy Promise IDE RAID vezérlőt alkalmazó rendszert fogunk beállítani. Miután telepítettük a kártyát és indítjuk a rendszert, bekéri a szükséges információkat. Kövessük az utasításokat és lépünk be a kártya beállító képernyőjére. Itt tudjuk kombinálni az összes csatlakoztatott meghajtónkat. Amikor ezzel a végeztünk, a lemezek egyetlen lemezként fognak a FreeBSD számára viselkedni. A többi RAID-szint is ehhez hasonlóan állítható be.

18.4.3. Az ATA RAID-1 tömbök újraszervezése

A FreeBSD lehetőséget a tömbben levő meghibásodott eszközök menet közben elvégezhető cseréjére. Ehhez arra van szükségünk, hogy még újraindítás előtt elcsípjuk a hibát.

Hiba esetén valami hasonlót fogunk látni a `/var/log/messages` állományban vagy a `dmesg(8)` kimenetében:

```
ad6 on monster1 suffered a hard error.
ad6: READ command timeout tag=0 serv=0 - resetting
ad6: trying fallback to PIO mode
ata3: resetting devices .. done
ad6: hard error reading fsbn 1116119 of 0-7 (ad6 bn 1116119; cn 1107 tn 4 sn 11)\\
```

```
status=59 error=40
ar0: WARNING - mirror lost
```

További információkat az `atacontrol(8)` programtól szerezhethetünk:

```
# atacontrol list
ATA channel 0:
    Master:      no device present
    Slave:      acd0 <HL-DT-ST CD-ROM GCR-8520B/1.00> ATA/ATAPI rev 0

ATA channel 1:
    Master:      no device present
    Slave:      no device present

ATA channel 2:
    Master:      ad4 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:      no device present

ATA channel 3:
    Master:      ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:      no device present

# atacontrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: DEGRADED
```

1. A lemez biztonságos eltávolításához először válasszuk le (`detach`) a meghibásodott lemezhez tartozó csatornát:

```
# atacontrol detach ata3
```

2. Cseréljük ki a lemezt.
3. Csatlakoztassuk újra (`attach`) az ATA csatornát:

```
# atacontrol attach ata3
Master:      ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
Slave:      no device present
```

4. Tartalékként (`spare`) adjuk hozzá az új lemezt a tömbhöz:

```
# atacontrol addspare ar0 ad6
```

5. Szervezzük újra (`rebuild`) a tömböt:

```
# atacontrol rebuild ar0
```

6. A folyamat előrehaladását a következő parancs begépelésével tudjuk figyelni:

```
# dmesg | tail -10
[a kimenet többi része]
ad6: removed from configuration
ad6: deleted from ar0 disk1
ad6: inserted into ar0 disk1 as spare

# atacontrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: REBUILDING 0% completed
```

7. Várjunk a művelet befejeződéséig.

18.5. USB tárolóeszközök

Írta: Marc Fonvieuille.

Manapság már számos külső tárolóeszköz az USB (Universal Serial Bus) közvetítésével csatlakozik a számítógéphez: merevlemez, pen drive-ok, CD-írók stb. A FreeBSD ezeket az eszközöket is ismeri.

18.5.1. Beállítás

A USB tárolóeszközöket kezelő meghajtó, az `umass(4)` felelős az USB alapú tárolóeszközök támogatásáért. Ha a `GENERIC` rendszermagot használjuk, akkor semmit sem kell változtatnunk. Ha saját rendszermagunk van, akkor gondoskodjunk róla, hogy a következő sorokat beraktuk a rendszermag beállításait tartalmazó állományba:

```
device scbus
device da
device pass
device uhci
device ehci
device usb
device umass
```

Az `umass(4)` meghajtó a SCSI alrendszeren keresztül éri el az USB tárolóeszközöket, tehát az USB eszközeinket a rendszer SCSI eszközként látja. Az alaplapon található USB chipkészlet típusától függően vagy csak a `device uhci`, vagy USB 1.x esetén pedig a `device ohci` bejegyzésre lesz szükségünk. De abból sem származik kárunk, ha mind a kettőt meghagyjuk. Az USB 2.0 szabványú vezérlőket a `ehci(4)` meghajtó (`device ehci`) támogatja. Ha módosítani kellett a konfigurációs állományt, akkor ne felejtjük el újrafordítani és telepíteni sem a rendszermagot.

Megjegyzés: Ha az USB eszközünk egy CD- vagy DVD-író, akkor a következő sorral a SCSI CD-meghajtók meghajtóját, a `cd(4)` eszközt kell beépítenünk a rendszermagba:

```
device cd
```

Mivel az író is SCSI eszközként látszik, ezért az `atapicam(4)` nem szerepelhet a rendszermag beállításai között.

18.5.2. A beállítások kipróbálása

A beállításaink készen állnak a kipróbálásra: csatlakoztassuk a számítógéphez az USB eszközünket és a rendszerüzeneteket tároló pufferben (`dmesg(8)`) hamarosan meg is jelenik a hozzá tartozó meghajtó:

```
umass0: USB Solid state disk, rev 1.10/1.00, addr 2
GEOM: create disk da0 dp=0xc2d74850
da0 at umass-sim0 bus 0 target 0 lun 0
da0: <Generic Traveling Disk 1.11> Removable Direct Access SCSI-2 device
da0: 1.000MB/s transfers
da0: 126MB (258048 512 byte sectors: 64H 32S/T 126C)
```

Természetesen a gyártóra, márkára, az eszköz leírójára (`da0`) és egyebekre vonatkozó részletek eltérhetnek.

Mivel az USB eszköz SCSI eszközként látszik, ezért a `camcontrol` parancs használható a rendszerhez csatlakoztatott USB tárolóeszközök listázásához:

```
# camcontrol devlist
<Generic Traveling Disk 1.11>          at scbus0 target 0 lun 0 (da0,pass0)
```

Ha a meghajtón állományrendszer is található, akkor képesek vagyunk csatlakoztatni. A 18.3 Szakasz elolvasása segíthet az USB meghajtón partíciókat kialakítani és formázni, amennyiben szükséges.

FigyelemA rendszer biztonsága szempontjából nem tekinthető megbízhatónak, ha olyan felhasználók számára is engedélyezzük tetszőleges meghajtók csatlakoztatását (például a `vfs.usermount` engedélyezésével), amelyekben nem bízunk meg. A FreeBSD által támogatott állományrendszerek döntő többsége nem nyújt védelmet a káros szándékkal telepített eszközök ellen.

Ha az eszközt normál felhasználókkal is csatlakoztathatóvá akarjuk tenni, akkor további lépések megtételére is szükségünk lesz. Először is a felhasználóknak valahogy el kell tudniuk érniük az USB tárolóeszköz csatlakoztatásakor keletkező eszközöket. Ezt úgy tudjuk megoldani, ha az érintett felhasználókat felvesszük az `operator` csoportba. Ebben a `pw(8)` lehet a segítségünkre. Másodsorban amikor ezek az eszközök létrejönnek, az `operator` csoportnak tudniuk kell ezeket olvasniuk és írniuk. Ezt úgy tudjuk megvalósítani, ha felvesszük a következő sorokat az `/etc/devfs.rules` állományba:

```
[localrules=5]
add path 'da*' mode 0660 group operator
```

Megjegyzés: Ha viszont vannak SCSI lemezeink is rendszerben, akkor a helyzet egy kicsit megváltozik. Tehát például a rendszerben már eleve vannak `da0`, `da1` és `da2` néven lemezek, akkor a második sort ennek megfelelően változtassuk meg:

```
add path 'da[3-9]*' mode 0660 group operator
```

Ezzel kizárunk minden, korábban már létező lemezt az `operator` csoportból.

Emellett még az `/etc/rc.conf` állományban engedélyoznünk kell a saját `devfs.rules(5)` szabályrendszerünket is:

```
devfs_system_ruleset="usb_rules"
```

Ezt követően be kell állítanunk a rendszermagban, hogy a hagyományos felhasználók képesek legyenek állományrendszereket csatlakoztatni. Ezt a legkönnyebb úgy tudjuk megtenni, ha az `/etc/sysctl.conf` állományba felvesszük a következő sort:

```
vfs.usermount=1
```

Azonban ne felejtjük el, hogy ez csak a rendszer következő indításától él. De a `sysctl(8)` parancs használatával is beállíthatjuk ezt az értéket.

Az utolsó lépésben hozzunk létre egy könyvtárat az állományrendszer csatlakoztatásához. Ezt a könyvtárat az a felhasználó fogja birtokolni, aki az állományrendszert csatlakoztatnia akarja. Ez például `root` felhasználóként úgy tudjuk megtenni, ha a felhasználónak létrehozunk egy könyvtárat `/mnt/felhasználó` néven (ahol a `felhasználó` nevet cseréljük a tényleges felhasználó nevére, a `csoport` nevet pedig a felhasználóhoz tartozó elsődleges csoport nevére):

```
# mkdir /mnt/felhasználó
```

```
# chown felhasználó:csoport /mnt/felhasználó
```

Most tegyük fel, hogy csatlakoztatunk egy USB pen drive-ot és ennek megfelelően megjelenik a /dev/da0s1 eszköz. Mivel az ilyen eszközökre általában gyárilag FAT állományrendszert tesznek, ezért így kell ezeket csatlakoztatni a mount(8) paranccsal:

```
% mount -t msdosfs -o -m=644,-M=755 /dev/da0s1 /mnt/felhasználó
```

Ha leválasztjuk az eszközt (miután kiadtuk a umount(8) parancsot), akkor a rendszerüzenetek között valami ilyesmit fogunk látni:

```
umass0: at uhub0 port 1 (addr 2) disconnected
(da0:umass-sim0:0:0:0): lost device
(da0:umass-sim0:0:0:0): removing device entry
GEOM: destroy disk da0 dp=0xc2d74850
umass0: detached
```

18.5.3. A témáról bővebben

A Lemezek hozzáadása és az Állományrendszerek csatlakoztatása és leválasztása című szakaszok elolvasása mellett a következő man oldalakat is ajánljuk: umass(4), camcontrol(8) és usbconfig(8) FreeBSD 8.x esetében, vagy usbdevs(8) a FreeBSD korábbi változatainál.

18.6. Lézeres tárolóeszközök (CD-k) létrehozása és használata

Írta: Mike Meyer.

18.6.1. Bevezetés

A CD-k számos lehetőségünkben eltérnek a hagyományos lemezektől. Kezdetben a felhasználók nem is voltak képesek írni ezeket. Olyannak tervezték, hogy a fejek sávok közti mozgásából fakadó késleltetés nélkül lehessen folyamatosan olvasni. A szállítása a maga idejében sokkal könnyebb volt minden vele egyforma méretű eszköznél.

A CD-ken is találhatunk sávokat, azonban ez csak a folyamatosan olvasható adat egy szakaszát jelenti, nem pedig a lemez fizikai tulajdonságát. Ha FreeBSD-n akarunk CD-t készíteni, akkor ehhez először össze kell állítanunk a CD egyes sávjaira kerülő adatokat és ezután rögzíteni ezeket a sávokat a CD-n.

Az ISO 9660 állományrendszert úgy tervezték, hogy megbirkózzon ezekkel az eltérésekkel. Sajnos ezzel együtt kőbe vették az állományrendszerek akkoriban érvényes korlátozásait is. Szerencsére lehetőséget ad bővítésre, ezáltal a helyesen megírt CD-k képesek úgy átlépni ezeket a határokat, hogy közben az általuk alkalmazott kiterjesztéseket nem ismerő rendszerekkel is együtt tudnak működni.

A sysutils/cdrtools port tartalmaz egy mkisofs(8) nevű programot, amellyel létre tudunk hozni ISO 9660 típusú állományrendszert tartalmazó adatállományt. Többféle kiterjesztést is ismer, amit majd a lentebb ismertett opciókkal érhetünk el.

A CD írásához használt konkrét segédeszköz attól függ, hogy ATAPI vagy esetleg más módon írunk. Az ATAPI CD-írók az alaprendszer részeként elérhető burned programon keresztül használhatóak. A SCSI és USB CD-írók esetén pedig a sysutils/cdrtools portban megtalálható cdrecord programot használhatjuk. Az

ATAPI/CAM modul segítségével a cdrecord és más SCSI-írókra készült programokat is tudunk használni ATAPI hardvereken.

Ha a CD-író szoftverünket grafikus felhasználói felületen keresztül szeretnénk használni, akkor az **X-CD-Roast** vagy a **K3b** alkalmazásokat érdemes szemügyre vennünk. Ezek az eszközök elérhetőek csomagként vagy a `sysutils/xcdrast` és `sysutils/k3b` portokból. ATAPI hardver esetén az **X-CD-Roast** és a **K3b** alkalmazások használatához szükségünk lesz az ATAPI/CAM modulra.

18.6.2. mkisofs

A `sysutils/cdrtools` port részeként elérhető `mkisofs(8)` program képes a UNIX típusú állományrendszer könyvtárszerkezetére alapján egy ISO 9660 típusú állományrendszert tartalmazó image-et készíteni. Legegyszerűbb módon így használhatjuk:

```
# mkisofs -o image.iso /az/elérési/út
```

Ezzel a paranccsal egy olyan `image.iso` nevű állományt hozunk létre, amely `/az/elérési/út` által megadott helyen található könyvtárszerkezetet mintázza ISO 9660 állományrendszer formájában. A folyamat során minden olyan állományt leképez szabványos ISO 9660 állományrendszerbeli névre, amely megfelel a szabvány elvárásainak, és kihagy minden olyan állományt, amely nem jellemző az ISO állományrendszerekre.

Számos opció lehet segítségünkre az ilyenkor felbukkanó akadályok leküzdésében. Ezek közül különösen fontos az `-R`, amely a UNIX rendszerek számára megszokott Rock Ridge kiterjesztéseket, valamint a `-J`, amely a Microsoft rendszerekben használt Joliet kiterjesztéseit, és végül a `-hfs`, amely a Mac OS alatt létrehozott HFS állományrendszerek kiterjesztéseit engedélyezi.

A kizárólag csak FreeBSD rendszereken használt CD-k esetében a `-U` megadásával kapcsolhatjuk ki az állománynevek mindenféle korlátozását. Az `-R` beállítás használatával olyan állományrendszer képét hozzuk létre, amely teljesen megegyezik a parancsban megadott könyvtárból induló fa tartalmával, habár több módon is sérti az ISO 9660 szabvány előírásait.

Az utolsó általános jelleggel használható beállítás a `-b`. Ezzel lehet megadni az "El Torito" szabványnak megfelelő rendszerindító CD készítéséhez szükséges rendszerindító image elérését. Ennél a beállításnál tehát meg kell adni a rendszerindításhoz használt lemez image-ét, amely a CD tartalmát magában foglaló könyvtárszerkezetben található valahol. A `mkisofs(8)` alapértelmezés szerint egy ún. "floppy emulációs" módban hozza létre az ISO image-et, ezért a rendszerindításhoz használatos lemez image-ének pontosan 1200, 1440 vagy 2880 KB méretűnek kell lennie. Egyes rendszerbetöltők, mint amilyen például a FreeBSD terjesztéséhez használt lemezekon található, nem használják ezt az emulációt. Ilyen helyzetekben a `-no-emul-boot` kapcsolót kell megadni. Tehát ha a `/tmp/sajátboot` könyvtárban van egy indítható FreeBSD rendszerünk, amelyben a `/tmp/sajátboot/boot/cdboot` a rendszerindító lemez image-e, akkor egy `/tmp/indítható.iso` nevű ISO 9660 formátumú állományrendszert tartalmazó image-et például így tudunk elkészíteni:

```
# mkisofs -R -no-emul-boot -b boot/cdboot -o /tmp/indítható.iso /tmp/sajátboot
```

Miután ezt megtettük, és a rendszermagunkban benne van az `md` eszköz támogatása, csatlakoztathatjuk is az állományrendszert:

```
# mdconfig -a -t vnode -f /tmp/indítható.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

Ezután már össze tudjuk vetni az `/mnt` és `/tmp/sajátboot` könyvtárak egyezőségét.

A mkisofs(8) viselkedését több más opcióval tudjuk finomhangolni, mint például az ISO 9660 kiosztás módosítása vagy a Joliet és HFS lemezek készítése. A mkisofs(8) man oldalon mindezekről bővebben olvashatunk.

18.6.3. burncd

Ha ATAPI CD-írónk van, akkor a burncd paranccsal írhatjuk az ISO image-et a lemezre. A burncd az alaprendszer része, és /usr/sbin/burncd néven érhető el. A használata igen egyszerű, csupán pár paramétere van:

```
# burncd -f eszköz data image.iso fixate
```

Ezzel a paranccsal rámásoljuk az *image.iso* állományt az *eszköz* eszközre. Az alapértelmezett eszköz a /dev/acd0. A burncd(8) man oldalán találjuk meg az írási sebességgel, a CD írás utáni kiadásával és az audio lemezek írásával kapcsolatos beállításokat.

18.6.4. cdrecord

Ha nincs ATAPI CD-írónk, akkor az íráshoz a cdrecord parancsot kell használnunk. A cdrecord nem az alaprendszer része: vagy a sysutils/cdrtools portból vagy a neki megfelelő csomagból kell telepítenünk. Az alaprendszerben végbemenő változások miatt a program bináris változatai hibázhatnak, aminek következtében csak “poháralátéteket” fogunk tudni gyártani. Ezért a rendszerrel együtt érdemes frissíteni ezt a portot is. Vagy ha a -STABLE verziót használjuk, akkor mindig érdemes a port elérhető legújabb verziójára frissíteni.

Miközben a cdrecord számos paraméterrel rendelkezik, az alapvető használata mégis egyszerűbb a burncd parancsnál. Egy ISO 9660 formátumú image-et ugyanis a következő módon tudunk felírni lemezre:

```
# cdrecord dev=eszköz image.iso
```

A cdrecord használatának trükkös része a megfelelő eszköz megtalálása, tehát a dev beállítás helyes megadása. Ehhez használjuk a cdrecord -scanbus paraméterét, amely az alábbihoz hasonló eredményt fog produkálni:

```
# cdrecord -scanbus
```

```
Cdrecord-Clone 2.01 (i386-unknown-freebsd7.0) Copyright (C) 1995-2004 Jörg Schilling
Using libscg version 'schily-0.1'
```

```
scsibus0:
```

```
  0,0,0    0) 'SEAGATE ' 'ST39236LW      ' '0004' Disk
  0,1,0    1) 'SEAGATE ' 'ST39173W      ' '5958' Disk
  0,2,0    2) *
  0,3,0    3) 'iomega ' 'jaz 1GB        ' 'J.86' Removable Disk
  0,4,0    4) 'NEC      ' 'CD-ROM DRIVE:466' '1.26' Removable CD-ROM
  0,5,0    5) *
  0,6,0    6) *
  0,7,0    7) *
```

```
scsibus1:
```

```
  1,0,0   100) *
  1,1,0   101) *
  1,2,0   102) *
  1,3,0   103) *
  1,4,0   104) *
  1,5,0   105) 'YAMAHA ' 'CRW4260      ' '1.0q' Removable CD-ROM
  1,6,0   106) 'ARTEC  ' 'AM12S        ' '1.06' Scanner
  1,7,0   107) *
```

Itt felsorolásra kerülnek a `dev` beállítás értékeként felhasználható eszközök. Keressük meg köztük a CD írókat és a `dev` értékének a három vesszővel elválasztott számot adjuk meg. Ebben az esetben a CD-író eszköz most az 1,5,0 lesz, tehát itt a helyes paraméterezés `dev=1,5,0`. Ezt az értéket könnyebben is meg lehet adni. Ennek részleteiről a `cdrecord(1)` man oldalán olvashatunk. Abban az esetben is érdemes fellapoznunk, ha az audio sávok írásáról, az írási sebesség korlátozásáról vagy más hasonló dolgokról akarunk olvasni.

18.6.5. Audio CD-k másolása

Audio CD-t úgy tudunk másolni, ha először állományok sorozatába mentjük a lemez tartalmát, majd ezeket az állományokat egy üres CD-re írjuk. Ennek konkrét folyamata azonban némileg eltér az ATAPI- és SCSI-meghajtók használata során.

SCSI-meghajtók esetén

1. A `cdda2wav` programmal mentsük le a lemez tartalmát.

```
% cdda2wav -v all -D2,0 -B -Owav
```
2. A `cdrecord` paranccsal írjuk fel a `.wav` kiterjesztésű állományokat.

```
% cdrecord -v dev=2,0 -dao -useinfo *.wav
```

Gondoskodjunk róla, hogy a `2,0` értéket a 18.6.4 Szakaszban megfelelően helyesen állítottuk be.

ATAPI-meghajtók esetén

Megjegyzés: Az ATAPI/CAM modul segítségével a `cdda2wav` parancs ATAPI meghajtókkal is használható. Ez a megoldás általában kedvezőbb (a hibák és bytesorrend ügyesebb kezelése, stb.) a legtöbb felhasználó számára, mint az itt ismertetett.

1. Az ATAPI CD meghajtója az egyes sávokat `/dev/acd0t nn` néven teszi elérhetővé, ahol a d a meghajtó sorszáma, a nn a sáv két számjeggyel kiírt sorszáma, amelyet szükség szerint balról nullával egészítenek ki. Így tehát az első meghajtó első sávja a `/dev/acd0t01`, a második a `/dev/acd0t02`, a harmadik a `/dev/acd0t03` és így tovább.

Ellenőrizzük, hogy ezek az eszközök jelen vannak a `/dev` könyvtárban. Amennyiben hiányoznának, kényszerítsük ki a lemez újbóli beolvasását:

```
# dd if=/dev/acd0 of=/dev/null count=1
```
2. Szedjük le az egyes sávokat a `dd(1)` használatával. A parancs kiadásakor meg kell adnunk egy blokkméretet is:

```
# dd if=/dev/acd0t01 of=track1.cdr bs=2352
# dd if=/dev/acd0t02 of=track2.cdr bs=2352
...
```
3. A `burncd` használatával írjuk fel a lemezre az imént lementett állományokat. Meg kell adnunk, hogy ezek audio állományok, és hogy a `burncd` a munka befejeztével zárja le (`fixate`) a lemezt.

```
# burncd -f /dev/acd0 audio track1.cdr track2.cdr ... fixate
```

18.6.6. Adat CD-k másolása

Az adatot tartalmazó CD-ket le tudjuk másolni egy olyan image-be, amely funkcionálisan megegyezik egy mkisofs(8) által létrehozott image-dzsel és amivel le tudunk másolni bármilyen adat CD-t. Az itt megadott példa azt feltételezi, hogy a CD-meghajtónk neve `acd0`. Helyére a saját CD-meghajtónk nevét kell behelyettesíteni.

```
# dd if=/dev/acd0 of=állomány.iso bs=2048
```

Most miután lementettük az image-et, írjuk fel CD-re a fentiek szerint.

18.6.7. Adat CD-k használata

Most, hogy már készítettünk egy szabványos adat CD-t, valószínűleg szeretnénk is valamilyen csatlakoztatni és elérni a rajta levő adatokat. Alapértelmezés szerint a `mount(8)` mindig azt feltételezi, hogy az állományrendszerek `ufs` típusúak. Ezért ha valami ilyesmivel próbálkozunk:

```
# mount /dev/cd0 /mnt
```

akkor egy `Incorrect super block` szövegű hibaüzenetet lesz a jutalmunk, és természetesen nem tudjuk csatlakoztatni a CD-t. Mivel a CD nem `ufs` állományrendszert tartalmaz, ezért az ilyen jellegű kísérleteink mind kudarcba fognak fulladni. Valahogy fel kell világosítanunk a `mount(8)` parancsot arról, hogy itt most egy `ISO9660` típusú állományrendszert akarunk csatlakoztatni, és akkor minden a helyére kerül. Ezt úgy tudjuk megtenni, ha a `mount(8)` parancsnak megadjuk a `-t cd9660` paramétert. Például, ha a `/dev/acd0` néven elérhető CD-meghajtóban levő lemezt akarjuk a `/mnt` könyvtárba csatlakoztatni, akkor ezt kell begépelnünk:

```
# mount -t cd9660 /dev/cd0 /mnt
```

Vegyük észre, hogy az eszköz neve (ez ebben a példában most `/dev/cd0`) lehet más is attól függően, hogy milyen csatolófelületet használ a CD-meghajtónk. Sőt, a `-t cd9660` valójában csak a `mount_cd9660(8)` parancsot indítja el. Ennek tükrében tehát az előbbi példát így rövidíthetjük le:

```
# mount_cd9660 /dev/cd0 /mnt
```

Ezen a módon bármilyen gyártmányú adat CD-t képesek vagyunk csatlakoztatni. Egyes ISO 9660 kiterjesztéseket használó lemezek azonban esetleg furcsán működhetnek. Például Joliet lemezek az összes állomány nevét kétbyte-os Unicode karakterben tárolják. A FreeBSD rendszermagja ugyan nem beszél a Unicode-ot, de a FreeBSD CD9660 meghajtója képes menetközben átkonvertálni a Unicode karaktereket. Ha bizonyos nem angol karakterek kérdőjeleként jelennének meg, akkor a `-C` beállítás használatával még egy helyi kódlapot is meg kell adnunk. Ezzel kapcsolatban bővebb tájékoztatásért forduljunk a `mount_cd9660(8)` man oldalhoz.

Megjegyzés: A `-C` beállítás segítségével csak akkor lesz képes a rendszermag elvégezni ezt az átalakítást, ha előtte betöltjük a `cd9660_iconv.ko` modult. Ezt megtehetjük úgy, hogy ha felvesszük a következő sort a `loader.conf` állományba:

```
cd9660_iconv_load="YES"
```

Indítsuk újra a számítógépünket, vagy közvetlenül töltsük be a modult a `kldload(8)` használatával.

Estenként előfordulhat, hogy kapunk egy `Device not configured` hibaüzenetet a CD-k csatlakoztatásakor. Ez általában arra utal, hogy a CD-meghajtó nem érzékeli a berakott lemezt, vagy éppen a meghajtó nem látható a

buszon. A CD-meghajtók esetében pár másodpercig eltarthat, amíg felismeri a berakott lemezt, ilyenkor mindig legyünk türelemmel.

Néha a SCSI CD-meghajtó nem látható, mert nem volt elég ideje válaszolni busz újraindítása előtt. Ha SCSI CD-meghajtónk van, akkor a következő beállítást tegyük hozzá a rendszermagunk konfigurációjához és fordítsuk újra a rendszermagukat.

```
options SCSI_DELAY=15000
```

Ezzel utasítjuk a SCSI buszunkat egy 15 másodperces várakozásra a rendszer indítása során, és így ezzel elég esélyt adunk arra, hogy a CD-meghajtó válaszolni tudjon a busz újraindítása előtt.

18.6.8. Nyers adat CD-k írása

Írhatunk közvetlenül is állományokat a CD-re, ISO 9660 formátumú állományrendszer használata nélkül. Sokan így oldják meg a mentést. Ezt sokkal gyorsabban lebonyolítható egy szabványos CD esetében:

```
# burncd -f /dev/acd1 -s 12 data archive.tar.gz fixate
```

Az ezen a módon megírt CD-ket szintén nyers módon kell olvasnunk:

```
# tar xzvf /dev/acd1
```

Az ilyen lemezeket nem tudjuk a normális CD-khez hasonlóan csatlakoztatni. Sőt, az ilyen CD-ket csak FreeBSD alatt tudjuk olvasni. Ha csatlakoztathatóvá akarjuk tenni a lemezt, vagy más operációs rendszerek alól is szeretnénk olvasni, akkor erre a célra a fentebb bemutatott mkisofs(8) parancsot kell használnunk.

18.6.9. Az ATAPI/CAM meghajtó használata

Írta: Marc Fonvieuille.

Ez a meghajtó lehetővé teszi az ATAPI eszközök (CD-ROM, CD-RW, DVD meghajtók stb...) számára, hogy a SCSI alrendszeren keresztül legyenek elérhetőek, így esetünkben is használhatóvá válnak olyan alkalmazások, mint például sysutils/cdrdao vagy a cdrecord(1).

A meghajtó használatához a következő sort kell a /boot/loader.conf állományba illeszteni:

```
atapicam_load="YES"
```

Indítsuk újra a számítógépet.

Megjegyzés: Amennyiben a rendszermagban az atapicam(4) statikus támogatását szeretnénk használni, úgy a következő sort kell a rendszermag konfigurációs állományába felvenni:

```
device atapicam
```

Továbbá a következő sorokra lesz még szükségünk:

```
device ata
device scbus
device cd
device pass
```

Ezeknek már eleve ott kell szerepelnie. Ezután fordítsuk újra és telepítsük a rendszermagot, majd indítsuk újra a számítógépet.

A rendszer indulásakor az írónak ehhez hasonló módon kell megjelennie:

```
acd0: CD-RW <MATSHITA CD-RW/DVD-ROM UJDA740> at ata1-master PIO4
cd0 at ata1 bus 0 target 0 lun 0
cd0: <MATSHITA CDRW/DVD UJDA740 1.00> Removable CD-ROM SCSI-0 device
cd0: 16.000MB/s transfers
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

A meghajtó most már elérhető a `/dev/cd0` eszközön keresztül, és például ennyi begépelésével csatlakoztatni tudunk róla egy CD-t a `/mnt` könyvtárba:

```
# mount -t cd9660 /dev/cd0 /mnt
```

root felhasználóként a következő paranccsal tudjuk lekérdezi az író SCSI címét:

```
# camcontrol devlist
<MATSHITA CDRW/DVD UJDA740 1.00> at scbus1 target 0 lun 0 (pass0,cd0)
```

Eszerint a `1,0,0` lesz az eszköz SCSI címe, amelyet a `cdrecord(1)` és más SCSI alkalmazások esetén adunk meg.

Az ATAPI/CAM és SCSI rendszerek tekintetében olvassuk el az `atapicam(4)` és `cam(4)` man oldalakat.

18.7. Lézeres tárolóeszközök (DVD-k) létrehozása és használata

Írta: Marc Fonvieille. Segítséget nyújtott benne: Andy Polyakov.

18.7.1. Bevezetés

A DVD a CD-hez képest a lézeres tárolóeszközök technológiájának újabb generációját képviseli. A DVD bármelyik CD-nél több adatot képes tárolni és napjaink ez a videók kiadásának szabványa.

Öt fizikailag írható formátummal határozhatjuk meg az írható DVD fogalmát:

- DVD-R: Ez volt az első elérhető írható DVD formátum. A DVD-R szabványát a DVD Fórum (<http://www.dvdforum.com/forum.shtml>) fektette le. Ez a formátum csak egyszer írható.
- DVD-RW: Ez a DVD-R szabvány újraírható változata. A DVD-RW körülbelül 1000 alkalommal írható újra.
- DVD-RAM: Ez is a DVD Fórum által támogatott újraírható formátum. A DVD-RAM cserélhető merevlemeznek látszik. Azonban ez típusú adathordozó nem kompatibilis legtöbb DVD-ROM hajtóval és DVD-Video lejátszóval. Csupán csak néhány DVD-író ismeri a DVD-RAM formátumot. A DVD-RAM használatáról a 18.7.9 Szakaszban találunk bővebben információkat.
- DVD+RW: Ezt az újraírható formátumot a DVD+RW szövetség (<http://www.dvdrw.com/>) alkotta meg. A DVD+RW lemezek nagyjából 1000 alkalommal írhatóak újra.
- DVD+R: Ez a formátum a DVD+RW formátum egyszer írható változata.

Az egyrétegű írható DVD-k összesen 4 700 000 000 byte-ot képesek rögzíteni, ami 4,38 GB vagy 4 485 MB (1 kilobyte itt 1024 byte).

Megjegyzés: Meg kell különböztetnünk fizikai tárolóeszközt és az alkalmazást. Például a DVD-Video állományok olyan jellegű elrendezését írja elő, ami bármelyik írható fizikai DVD eszközön megjelenhet: DVD-R, DVD+R, DVD-RW stb. Mielőtt kiválasztanánk az eszköz típusát, biztosnak kell lennünk benne, hogy az író és a DVD-Video lejátszó (ez lehet egy önálló lejátszó vagy egy számítógép DVD-ROM meghajtója) kompatibilis a szóbanforgó lemezzel.

18.7.2. Beállítás

A growisofs(1) programot fogjuk a DVD rögzítésére használni. Ez a program a **dvd+rw-tools** segédprogramok (sysutils/dvd+rw-tools) gyűjteményének része. A **dvd+rw-tools** az összes DVD médium típusát ismeri.

Ezek a segédprogramok a SCSI alrendszeren keresztül érik az eszközöket, ezért a használhatukhoz a rendszermagban szükségünk lesz az ATAPI/CAM támogatásra. Ha az írónk USB felületen csatlakozik, akkor mindez szükségtelen, és ehelyett a 18.5 Szakaszt kell elolvasnunk az USB eszközök beállításához.

Engedélyeznünk kell az ATAPI eszközök DMA hozzáférését is, amit a /boot/loader.conf állományban a következő sor hozzáadásával tudunk megtenni:

```
hw.ata.atapi_dma="1"
```

A **dvd+rw-tools** használatának megkezdése előtt a DVD-írónkkal kapcsolatban érdemes átolvasnunk a dvd+rw-tools hardverkompatibilitási jegyzeteit (angolul) (<http://fy.chalmers.se/~appro/linux/DVD+RW/hcn.html>).

Megjegyzés: Ha grafikus felületet szeretnénk használni, akkor érdemes egy pillanatast vetnünk a **K3bre** (sysutils/k3b), amely egy felhasználóbarát felületet ad a growisofs(1) és sok más íróprogram felé.

18.7.3. Adat DVD-k írása

A growisofs(1) a mkisofs parancs előlapja, tehát az állományrendszer létrehozásához a mkisofs(8) programot fogja meghívni és ezt írja fel a DVD-re. Ez azt jelenti, hogy az írási folyamat megkezdése előtt nem kell semmilyen image-et létrehoznunk.

A /az/elérési/út könyvtárból a következő paranccsal tudjuk kiírni az adatokat DVD+R vagy DVD-R lemezre:

```
# growisofs -dvd-compatible -Z /dev/cd0 -J -R /az/elérési/út
```

A -J -R beállítások a mkisofs(8) programhoz kerülnek át az állományrendszer létrehozásakor (itt most egy ISO 9660 állományrendszert hozunk létre, Joliet és Rock Ridge kiterjesztésekkel), használatának részleteit lásd mkisofs(8).

A -Z beállítást a kezdőmenetek létrehozásakor használjuk: több menetben akarjuk írni a lemezt vagy sem. A DVD eszközt, amely itt most a /dev/cd0, a saját konfigurációnknak megfelelően kell megadni. A -dvd-compatible paraméterrel lezárjuk a lemezt, így ezután további írás már nem lehetséges. Ezért cserébe jobb kompatibilitást kapunk a DVD-ROM meghajtókkal.

Előre legyártott image-dzsel is dolgozhatunk, tehát például, ha az *image.iso* állományt akarjuk kiírni, akkor ezt kell lefuttatnunk:

```
# growisofs -dvd-compatible -Z /dev/cd0=image.iso
```

Az írási sebességet magától beállítja a lemez és meghajtó képességeinek megfelelően. Az írási sebesség felülbírálásához használjuk a `-speed=` paramétert. A paraméterek lehetőségeiről a `growisofs(1)` man oldaláról tudhatunk meg többet.

Megjegyzés: 4,38 GB-nál több adat írásához egy hibrid UDF/ISO-9660 típusú állományrendszert kell létrehoznunk. Ezt úgy tudjuk elérni, ha `mkisofs(8)` és a többi hasonló program (például `growisofs(1)`) hívásakor még hozzátesszük az `-udf -iso-level 3` paramétereket. Ezekre csak lemezképek készítésekor vagy az állományok közvetlen lemezre írásakor van szükségünk. Az így létrehozott lemezeket a `mount_udf(8)` segédprogram segítségével UDF állományrendszerként tudjuk csatlakoztatni. Ezért csak olyan operációs rendszereken használható, amelyek ismerik ezt a formátumot, ellenkező esetben csak hibás állományokat fogunk látni a lemezen.

Példa ilyen lemezkép létrehozására:

```
# growisofs -dvd-compatible -udf -iso-level 3 -Z /dev/cd0 -J -R /az/új/adat/helye
```

Ha a lemezkép már eleve nagyobb méretű állományokat tartalmaz, a lemez írásakor a `growisofs(1)` programnak már nem kell további paramétereket átadnunk.

Lehetőleg mindig a `sysutils/cdrtools` legfrissebb verzióját használjuk (amely a `mkisofs(8)` programot is tartalmazza), mivel a régebbi verziók nem támogatják a nagyobb méretű állományokat. Ha problémák adódnak a programok használata során, akkor próbálkozzunk a fejlesztői változattal (`sysutils/cdrtools-devel`) és olvassuk el a `mkisofs(8)` man oldalát.

18.7.4. DVD-Video írása

A DVD-Video az állományok speciális szervezésére utal, amely az ISO 9660 és az mikró UDF (M-UDF) specifikációkon alapszik. A DVD-Video emellett egy adott adatszerkezeti hierarchiát is takar, ezért kell egy külön programmal, például a `multimedia/dvdauthor` segítségével összeállítani egy DVD-t.

Ha már a birtokunkban van egy DVD-Video állományrendszer képe, akkor az eddigiek szerint egyszerűen csak írjuk fel egy lemezre, ahogy azt az előző szakaszban is láthattuk. Ha összeállítottuk a DVD anyagát és például a `/a/video/elérési/útja` könyvtárba raktuk, akkor a következő paranccsal írathatjuk ki a DVD-Video formátumú lemezt:

```
# growisofs -Z /dev/cd0 -dvd-video /a/video/elérési/útja
```

A `-dvd-video` paramétert kell átadni a `mkisofs(8)` programnak, amelynek hatására létrehoz egy DVD-Video formátumú állományrendszert. Emellett a `-dvd-video` beállítás maga után vonja a `growisofs(1)` `-dvd-compatible` beállítását is.

18.7.5. A DVD+RW használata

Eltérően a CD-RW-től, egy érintetlen DVD+RW-t az első használat előtt meg kell formázni. A `growisofs(1)` program erről az első adandó alkalommal gondoskodik, és ez az *ajánlott*. Azonban a DVD+RW formázására használhatjuk a

dvd+rw-format parancsot is:

```
# dvd+rw-format /dev/cd0
```

Ezt a műveletet csak egyszer kell elvégezni, hiszen ne feledjük, hogy csak a szűz DVD+RW lemezeket kell megformázni. Ezután a DVD+RW-t a korábbi szakaszoknak megfelelően tudjuk írni.

Ha a DVD+RW-re új adatot akarunk írni (egy teljesen új állományrendszert, nem pedig adatokat hozzáfűzni), akkor nem kell üressé tenni a lemezt, egyszerűen csak elegendő felülírni az előzőeket (egy új kezdőmenet létrehozásával) valahogy így:

```
# growisofs -Z /dev/cd0 -J -R /az/új/adat/helye
```

A DVD+RW formátum felajánlja annak lehetőségét is, hogy könnyedén hozzá lehessen fűzni adatokat az előző íráshoz. A művelet során az új menetet összefűzi a meglévővel, tehát ez nem egy többmenetes írás, hanem a growisofs(1) *megnöveli* a lemezen található ISO 9660 állományrendszert.

Például, ha egy korábban megírt DVD+RW lemezen levő adatokhoz akarunk hozzáírni, akkor a következő parancsot kell kiadnunk:

```
# growisofs -M /dev/cd0 -J -R /az/új/adat/helye
```

A mkisofs(8) beállításainál a kezőmenetnél megadottakat érdemes ismét megadni.

Megjegyzés: Ha kompatibilisek akarunk maradni a többi DVD-meghajtóval, akkor adjuk meg `-dvd-compat` paramétert. Ez a DVD+RW esetében annyit jelent, hogy nem tudunk további adatokat hozzáfűzni.

Ha valamilyen okból mégis üressé szeretnénk tenni a lemez, akkor ír járhatunk el:

```
# growisofs -Z /dev/cd0=/dev/zero
```

18.7.6. A DVD-RW használata

A DVD-RW két lemezformátumot fogad el: a inkrementális soros hozzáférést és a korlátozott felülírást. Alapértelmezés szerint a DVD-RW lemezek soros elérésűek.

A még fel nem használt DVD-RW lemezek közvetlenül írhatóak külön formázás nélkül, habár a korábban már soros formátumban használt DVD-RW lemezeket egy új kezdőmenet létrehozása előtt üressé kell tenni.

Soros módban így kell letörölni egy DVD-RW lemezt:

```
# dvd+rw-format -blank=full /dev/cd0
```

Megjegyzés: A teljes törlés (`-blank=full`) egy 1x média esetén körülbelül egy órát vesz igénybe. A `-blank` beállítással egy gyorsított törlés zajlik le, amennyiben a DVD-RW lemezt Disk-At-Once (DAO) módban írjuk. A DVD-RW lemezeket az alábbi paranccsal tudjuk DAO módban írni:

```
# growisofs -use-the-force-luke=dao -Z /dev/cd0=image.iso
```

A `-use-the-force-luke=dao` beállítást nem kötelező megadni, mivel a growisofs(1) igyekszik a lehető leggyorsabban törölni a lemezt és megkezdeni a DAO módú írást.

A DVD-RW esetében valójában a korlátozott felülírást lenne érdemes használnunk, mivel ez a formátum sokkal rugalmasabb az alapértelmezés szerint felkínált inkrementális soros elérésnél.

A soros DVD-RW lemezekre ugyanúgy tudunk adatokat rögzíteni, mint az összes többi formátum esetében:

```
# growisofs -Z /dev/cd0 -J -R /az/adat/helye
```

Ha az előző íráshoz akarunk még hozzáfűzni adatokat, akkor ehhez a growisofs(1) -m beállítását kell használnunk. Azonban ha a DVD-RW lemezhet inkrementális soros módban adunk hozzá adatot, akkor ezzel egy új menetet hozunk létre a lemezen és így egy többmenetes lemezt kapunk.

A korlátozott felülírási DVD-RW formátum használata esetén nem kell mindegyik kezdőmenet előtt törölni a lemezt, egyszerűen csak felül kell írni a -z beállítással, hasonlóan a DVD+RW esetéhez. A DVD+RW -m beállításához hasonlóan lehetőségünk van a lemezen található ISO 9660 formátumú állományrendszer növelésére. Ennek az eredménye egy egymenetes DVD.

A következő paranccsal tudjuk a DVD-RW lemezt korlátozott felülírási módba tenni:

```
# dvd+rw-format /dev/cd0
```

Így tudunk visszaváltani a soros formátum használatára:

```
# dvd+rw-format -blank=full /dev/cd0
```

18.7.7. Több menet használata

Nagyon kevés DVD-ROM meghajtó ismeri a többmenetes DVD-ket, és legtöbbször is csak általában az első menetet olvassák. A DVD+R, DVD-R és DVD-RW formátumok soros formátumban képesek több mentetet is befogadni, viszont a DVD+RW és DVD-RW korlátozott felülírási formátuma esetén nem létezik több menet.

Az alábbi parancs egy újabb menetet ad hozzá egy megkezdett (le nem zárt) DVD+R, DVD-R vagy DVD-RW soros formátumú lemezhez:

```
# growisofs -M /dev/cd0 -J -R /az/új/adat/helye
```

Ha ezt a parancsot egy korlátozott felülírási DVD+RW vagy DVD-RW lemez esetén adjuk ki, akkor az új adatokat úgy fűzi hozzá, hogy egy új menetet összefűsüli a meglévővel. Ezzel egy egymenetes lemez keletkezik. Ilyenkor így bővítik a megkezdett lemezeket.

Megjegyzés: A menetek kezdése és befejezése általában felhasznál valamennyi helyet a lemezen. Ezért úgy tudjuk optimalizálni a lemez helykihasználtságát, hogy kevés menetben sok adatot viszünk fel rá. A DVD+R esetén 154, a DVD-R-nél körülbelül 2000, és a dupla rétegű DVD+R lemezeknél 127 menetet tudunk létrehozni.

18.7.8. További olvasnivalók

A DVD lemezeiről részletesebb információkat a dvd+rw-mediainfo /dev/cd0 parancs kiadásával tudunk lekérdezni.

A **dvd+rw-tools** használatáról a growisofs(1) man oldalon találunk információt, valamint a dvd+rw-tools honlapján (angolul) (<http://fy.chalmers.se/~appro/linux/DVD+RW/>) és a cdwrite levelezési lista (<http://lists.debian.org/cdwrite/>) archívumaiban (angolul).

Megjegyzés: Futassuk `dvd+rw-mediainfo` parancsot minden olyan esetben, amikor gondunk akad valamilyen lemez írásával. A kimenete nélkül szinte lehetetlen segítenünk bárkinek is.

18.7.9. A DVD-RAM használata

18.7.9.1. Beállítás

A DVD-RAM írók SCSI vagy ATAPI csatolófelülettel rendelkeznek. Az ATAPI eszközök esetén engedélyezni kell a DMA elérését, amit a `/boot/loader.conf` állományban az alábbi sor hozzáadásával tudunk megtenni:

```
hw.ata.atapi_dma="1"
```

18.7.9.2. A lemez előkészítése

Ahogy arra már korábban utaltunk a fejezet bevezetésében, a DVD-RAM úgy látható, mint egy cserélhető merevlemez. A hagyományos merevlemezekhez hasonlóan a DVD-RAM-ot is “elő kell készíteni” az első használatához. Ebben a példában a lemez teljes területét egy szabványos UFS2 állományrendszerrel töltjük fel:

```
# dd if=/dev/zero of=/dev/acd0 bs=2k count=1
# bsdlabel -Bw acd0
# newfs /dev/acd0
```

A DVD eszköz nevét, vagyis az `acd0` eszközt a saját rendszerünknek megfelelően kell módosítani.

18.7.9.3. A lemez használata

Miután az előbbi műveletet elvégeztük a DVD-RAM lemezen, már tudjuk is normális merevlemezként csatlakoztatni:

```
# mount /dev/acd0 /mnt
```

Ezt követően a DVD-RAM egyaránt olvasható és írható.

18.8. Hajlékonylemezek létrehozása és használata

Eredetileg készítette: Julio Merino. Átdolgozta: Martin Karlsson.

Néha hasznos lehet, ha az adatokat floppy lemezekben tároljuk, például olyankor, amikor más cserélhető tárolóeszköz már nem jöhet számításba, vagy amikor kis mennyiségű adatot kell átvinnünk az egyik számítógépről a másikra.

Ebben a szakaszban bemutatjuk hogyan kell FreeBSD alatt floppy lemezeket használni. Elsősorban a 3,5 colos DOS lemezek formázásával és használatával foglalkozik, de ezek fogalmak a többi hajlékonylemezes formátum esetében is hasonlóak.

18.8.1. A hajlékonylemezek formázása

18.8.1.1. Az eszköz

A floppy lemezek a többi eszközhöz hasonlóan a `/dev` könyvtárban érhetőek el. A nyers floppy lemezek eléréséhez egyszerűen csak használjuk a `/dev/fdN` hivatkozást.

18.8.1.2. A formázás

Használat előtt a floppy lemezeket alacsony szinten meg kell formázni. Ezt általában maga a gyártó végzi el, de a formázás gyakran hasznos lehet a lemez sértetlenségének ellenőrzésére. A legtöbb floppy lemez hivatalos kapacitása 1440 KB, de használhatjuk nagyobb (és kisebb) méretekben is.

A floppy lemezek alacsony szintű formázására az `fdformat(1)` parancsot használhatjuk. Ez a segédprogram paraméterként az eszköz nevét várja.

Figyeljünk a menetközben megjelenő hibaiüzenetekre, mivel ezek segítik eldönteni, hogy a lemez használható vagy sem.

18.8.1.2.1. A hajlékonylemezek formázása

A `/dev/fdN` eszközök segítségével tudunk megformázni egy floppy lemezt. Tegyük be egy 3,5 colos floppy lemezt a meghajtóba, majd adjuk ki a következő parancsot:

```
# /usr/sbin/fdformat -f 1440 /dev/fd0
```

18.8.2. A lemez címkézése

Miután alacsony szinten formáztuk a lemezt, tennünk kell rá egy lemezcímkét is. Ez a lemezcímke később meg fog semmisülni, de a rendszernek szüksége van rá, hogy pontosan meg tudja állapítani a lemez méretét és geometriáját.

Az új lemezcímke lefedi az egész lemezt, és tartalmazni fogja az összes információt a floppy geometriájáról. A lemezcímkék geometriaértékeit az `/etc/disktab` állományban találjuk meg felsorolva.

Most már futtathatjuk is a `bsdlabel(8)` parancsot:

```
# /sbin/bsdlabel -B -w /dev/fd0 fd1440
```

18.8.3. Az állományrendszer

A hajlékonylemez most már készen áll a magas szintű formázásra. Ennek során egy új állományrendszert teszünk rá, amelyet a FreeBSD képes írni és olvasni. Miután létrejött ez az új állományrendszer, a lemezcímke megsemmisül, így tehát ha újra meg akarjuk formázni a lemezt, akkor újra létre kell majd hoznunk a lemezcímkét.

A floppy állományrendszere lehet UFS vagy FAT. A FAT általánosságban véve jobb választás a floppy lemezek számára.

Az alábbi módon tudunk új állományrendszert tenni a floppyra:

```
# /sbin/newfs_msdos /dev/fd0
```

A lemez most már készen áll a használatra.

18.8.4. A hajlékonylemezek használata

A floppy lemezt használatához a `mount_msdosfs(8)` paranccsal kell csatlakoztatnunk. Ugyanerre a célra használhatjuk a Portgyűjteményből elérhető `emulators/mttools` portot is.

18.9. Szalagok létrehozása és használata

A legfontosabb szalagos adathordozók a 4 mm-es, 8 mm-es, QIC, a minikazettás és a DLT.

18.9.1. 4 mm-es (Digitális adattároló, avagy DDS: Digital Data Storage)

A 4 mm-es szalagok a QIC-szalagokat váltják fel a munkaállomások biztonsági mentésének eszközeként. Ez a tendencia csak tovább növekedett, ahogy a Conner felvásárolta az Archive-ot, a QIC típusú meghajtók legnagyobb gyártóját, majd leállított a QIC-meghajtók gyártását. A 4 mm-es meghajtók mérete kicsi és csendben is dolgoznak, de a megbízhatóság terén nem tudhatják maguknak mindazt a sikert, amit a 8 mm-es társaiknál könyvelhettünk el. A kazetták is sokkal olcsóbbak és kisebbek (3 x 2 x 0,5 col, ami 76 x 51 x 12 mm) a 8 mm-es kiadásénál. A 4 mm-es feje, hasonlóan a 8 mm-eséhez, valamilyen okból szintén viszonylag rövid ideig bírja, és mind a kettő spirális pásztázást használ.

Ezeknél a meghajtóknál az adatátvitel nagyjából 150 KB/mp-nél kezdődik és 500 KB/mp-nél végződik. Az adattárolási képességük 1,3 GB-tól indul és 2,0 GB-ig tart. A hardveres tömörítés, ami a legtöbb ilyen típusú meghajtónál elérhető, közel megduplázza a kapacitást. A többmeghajtós szalagos könyvtár egységek egyetlen szekrényben 6 meghajtót képes befogadni, a szalagok automatikus cserélgetésével. Az ilyen könyvtárak kapacitása a 240 GB-ot is elérheti.

A DDS-3 szabvány most már akár 12 GB (vagy tömörítve 24 GB) kapacitást is elérhetővé tesz.

A 4 mm-es meghajtók, hasonlóan a 8 mm-es meghajtókhoz, spirális pásztázást alkalmaznak. A spirális pásztázás összes előnye és hátránya ezért egyaránt él a 4 mm-es és 8 mm-es meghajtók esetén.

A szalagok 2 000 menet vagy 100 teljes mentes után kopnak el.

18.9.2. 8 mm-es (Exabyte)

A 8 mm-es szalagok a legelterjedtebb szalagos SCSI-meghajtók. A szalagok használatára ez a legjobb választás. Szinte mindegyik rendszerben egy 2 GB-os 8 mm-es Exabyte szalagos meghajtót használnak. A 8 mm-es meghajtók megbízhatóak, kényelmesek és csendesek. A kazetták olcsók és kicsik (4,8 x 3,3 x 0,6 col, azaz 122 x 84 x 15 mm). A 8 mm-es szalagok feje viszonylag csak rövid ideig bírja a szalag nagy mértékű oda-vissza mozgása miatt.

Az adatátvitel sebessége 250 KB/mp-től 500 KB/mp-ig terjed, valamint a 300 MB-tól egészen 7 GB-os méretig találkozhatunk velük. A meghajtókban elérhető hardveres tömörítés képes közel megduplázni a kapacitást. Ezek a meghajtók önálló egységként is beszerezhetők vagy egy 6 egységből álló és 120 szalagos szalagos könyvtár részeként. Ezek az egységek önállóan váltják a szalagokat. Az ilyen könyvtárak kapacitása eléri a közel 840 GB-ot.

Az Exabyte “Mammoth” modellje szalagonként 12 GB (tömörítéssel pedig 24 GB) adatot képes tárolni, viszont a hagyományos szalagos meghajtóknál nagyjából kétszer többbe kerül.

Az adatok spirális pásztázással kerülnek a szalagra, és a fejek adott (nagyjából 6 fokos) szögben állnak a szalag felett. A szalag a fejeket tartó orsó köre tekeredik, körülbelül 270 fokban. Ennek eredményképpen nagyobb adatsűrűség és szorosan zárt sávok jönnek létre, ahogy ebben a szögben a fej eljut a szalag egyik éléről a másikra.

18.9.3. QIC

A QIC-150 meghajtók és szalagok talán a legelterjedtebb szalagos egységek és adathordozók. A QIC szalagos meghajtók a legolcsóbb “komolynak tekinthető” biztonsági mentésre alkalmas meghajtók. Az olcsóság azonban megköveteli a maga árát. A QIC-szalagok a 4 és 8 mm-es szalagokkal szemben akár ötször is drágábbak lehetnek gigabyte-onként. De ha megelégszünk csupán féltucat szalaggal is, akkor a QIC jó vásárnak tűnhet. A QIC a *leginkább* elterjedtebb szalagos meghajtó. Minden rendszerben biztonságosan találunk valamilyen minőségben QIC-meghajtót. A QIC fizikailag hasonló (és gyakran azonos) felépítésű szalagokat gyárt rengeteg különböző adatsűrűséggel. Az ilyenkor keletkező súrlódások miatt a QIC-meghajtók egyáltalán nem nevezhetők csendesnek. Az ilyen típusú meghajtók az adatok rögzítése előtt külön hangjelenség kíséretében keresik meg a megfelelő pozíciót és tisztán hallható, ahogy olvasnak, írnak és keresnek. A QIC-szalagok mérete 6 x 4 x 0,7 col (avagy 152 x 102 x 17 mm).

Az adatátviteli sebesség nagyjából 150 KB/mp-től 500 KB/mp-ig terjedhet. A kapacitás szalagonként 40 MB és 15 GB között változhat. A legtöbb újabb QIC-meghajtó támogatja a hardveres tömörítést. QIC-meghajtókat azonban egyre kevésbé találhatunk, helyüket szépen lassan mindenhol átveszik a DAT-meghajtók.

A szalagokra sávokban rögzítik az adatokat. Ezek a sávok szalag felületének hosszanti tengelyén futnak az egyik végétől a másikig. A sávok száma valamint a sávok vastagsága a szalagok kapacitásától függően változnak. Ha nem is összes legújabb, de a legtöbb meghajtó legalább olvasás szintjén kompatibilis a régebbi típusokkal (de gyakran írásban is). A QIC híresen megbízható az adatbiztonság tekintetében (a mechanikája sokkal egyszerűbb és strapabíróbb a spirális pásztázással működő meghajtókénál).

A szalagokat 5000 mentés után érdemes lecserélni.

18.9.4. DLT

A DLT rendelkezik a legnagyobb adatátviteli sebességgel az itt összefoglalt mezőnyben. A 1/2 colos (12,5 mm-es) szalag egy egyorsós tokban foglal helyet (mérete 4 x 4 x 1 col, azaz 100 x 100 x 25 mm). A tok egyik oldalán végig egy csúszó kapu található. A meghajtó ezt a kaput nyitja ki és ezen keresztül húzza be a szalagot. A szalag elején található egy ovális lyuk, amibe a meghajtó “bele tud akaszzkodni”. A feszítő orsó a szalagos meghajtóban foglal helyet. Az összes többi szalag esetén (kivéve egyedül a 9 sávú szalagokat) mind a segéd- és feszítő orsók magában a kazettában találhatóak.

Az adatátviteli sebessége megközelítőleg 1,5 MB/mp, tehát háromszor nagyobb bármelyik 4 mm-es, 8 mm-es vagy QIC-szalagos egységénél. Az adattároló képessége kazettánként 10 GB-tól 20 GB-ig terjedhet. A meghajtók egyaránt elérhetők többkazettás, cserélgetős és többkazettás, többmeghajtós könyvtárakban is, melyek 5 kazettától

egészen 900 kazettáig, illetve 1 meghajtótól 20 meghajtóig képesek befogadni, így teljes tárterületük 50 GB-tól 9 TB-ig terjed.

A DLT Type V formátum tömörítéssel közel 70 GB-os kapacitást képes elérni.

A szalagra az adatok a haladási iránnyal párhuzamosan kerülnek fel (akárcsak a QIC-szalagok esetében). Egyszerre két sávot rögzít. A író/olvasó fejek élettartama viszonylag nagy. Ahogy a szalag megáll, a fej és a szalag között nincs szükség további relatív mozgásra.

18.9.5. AIT

Az AIT a Sony új formátuma, ami egészen 50 GB mennyiségű adatot képes tárolni (tömörítéssel) egyetlen szalagon. A szalagokat memóriachipekkel látják el, melyek a szalag tartalmát indexelik. Az indexek felhasználásával aztán a szalagos meghajtó villámgyorsan képes meghatározni a szalagon található állományok helyét, szemben az ilyenkor megszokott többperces művelettel. A **SAMS:Alexandria** és a hozzá hasonló szoftverek negyven vagy több AIT-szalagos könyvtárral is képesek egyszerre dolgozni, és közvetlenül a szalagok memóriájával veszik fel a kapcsolatot a tartalmuk megjelenítéséhez, a mentett állományok rendszerezéséhez, a helyes szalag megkereséséhez, betöltéséhez és visszatöltéséhez.

Az ilyen könyvtárak a 20 000 dolláros (kb. 3,5 millió forint) árkategóriába tartoznak, ami miatt csak egy kicsivel csúsznak ki a hobbi kategóriából.

18.9.6. Az új szalagok első használata

Amikor az első alkalommal akarunk beolvasni vagy írni egy új, teljesen üres szalagot, hibára fogunk futni. Egy ehhez hasonló konzolüzenet fog megjelenni:

```
sa0(ncr1:4:0): NOT READY asc:4,1
sa0(ncr1:4:0): Logical unit is in process of becoming ready
```

A szalag nem tartalmaz azonosító blokkot (Identifier Block) a nulladik blokkban. A QIC-525 szabvány átvétele óta mindegyik QIC szalagos meghajtó létrehozza ezt az azonosító blokkot. Tehát két megoldás létezik:

- Az `mt fsf 1` paranccsal felírunk egy ilyen azonosító blokkot a szalagra.
- A meghajtó előlapján található gomb segítségével dobassuk ki a szalagot.

Rakjuk vissza a szalagot és hajtsunk végre rajta egy `dump` parancsot.

A `dump` parancs erre egy `DUMP: End of tape detected` ("szalag vége") hibaüzenetet ad, majd a következő jelenik meg a konzolon: `HARDWARE FAILURE info:280 asc:80,96`.

Tekertessük vissza a szalagot az `mt rewind` paranccsal.

A szalag következő művelete most már sikeres lesz.

18.10. Biztonsági mentés hajlékonylemezekre

18.10.1. Hajlékonylemezre is lehet biztonsági mentést készíteni?

A floppy lemezek nem igazán felelnek meg biztonsági mentés készítésére, mivel:

- Nem megbízható adathordozók, különösen hosszabb időre.
- Esetükben a mentés és visszaállítás nagyon lassú.
- Kapacitásuk erősen korlátozott (annak már régen elmúlt az ideje, amikor egész merevlemezeket tudtunk lementeni egy tucat floppyra).

Habár ha máshogy nem tudunk biztonsági mentést készíteni, akkor a floppy lemezekkel még mindig jobban járunk, mint nélkülük.

Ha már mindenképpen floppy lemezeket kell használnunk, akkor igyekezzünk minél jobb minőségűeket beszerezni. Tehát az olyan floppyk, amik már évek óta kavarognak az irodában, erre a célra nem éppen bizonyulnak a legjobb választásnak. Ideális esetben egy megbízható gyártótól származó új floppykat használunk.

18.10.2. Tehát akkor hogyan mentsük az adatokat hajlékonylemezre?

Legegyszerűbben a `tar(1)` –M (többkötetes) opciójával tudunk floppy lemezre menteni, aminek használatával több floppyra kiterjedő mentéseket is készíthetünk.

Az aktuális könyvtár és a benne levő alkönyvtárak tartalmát (`root`) felhasználóként a következő paranccsal tudjuk lementeni:

```
# tar Mcvf /dev/fd0 *
```

Amikor az első floppy megtelik, a `tar(1)` kérni fogja a következő kötetet (volume) (mivel a `tar(1)` adathordozótól független módon hivatkozik a kötetekre, tehát ebben a környezetben a kötet egy floppy lemezt jelent):

```
Prepare volume #2 for /dev/fd0 and hit return:
```

Az üzenet fordítása:

```
Készítse elő a 2. kötetet a /dev/fd0 eszközön és nyomja le a  
return billentyűt
```

A folyamat egészen addig ismétlődik (a kötetek számának növekedésével), amíg az összes állomány lementésre nem kerül.

18.10.3. Lehet tömöríteni a mentéseket?

Sajnos a `tar(1)` többkötetes mentések esetén nem engedi a `-z` beállítás használatát. Természetesen ettől függetlenül a `gzip(1)` segítségével még be tudjuk tömöríteni az összes állományt, a `tar(1)` paranccsal floppyra menteni ezeket, majd a `gunzip(1)` paranccsal kitömöríteni.

18.10.4. Hogyan állítsuk vissza a biztonsági mentéseket?

Az egész mentés visszaállításához adjuk ki a következő parancsot:

```
# tar Mxvf /dev/fd0
```

Két módon tudunk csak bizonyos állományokat visszaállítani. Először is, tegyük be a mentés első lemezét és adjuk ki a következő parancsot:

```
# tar Mxvf /dev/fd0 állomány
```

A tar(1) segédprogram ezután sorban kérni fogja a többi lemezt egészen addig, amíg meg nem találja a keresett állományt.

Vagy ha pontosan tudjuk, hogy melyik lemezen található a keresett állomány, akkor az iménti parancs használatát azzal a lemezzel kezdjük. Vigyázzunk, mert ha a lemezen található első állomány az előző lemezen kezdődik, akkor a tar(1) figyelmeztetni fog minket, hogy nem állítja vissza még akkor sem, ha erre nem is kértük!

18.11. Mentési stratégiák

Eredetileg készítette: Lowell Gilbert.

Egy biztonsági mentés kidolgozása során az első követelmény gondoskodnunk az alábbi problémákról:

- Lemezhiba
- Az állományok véletlen törlése
- Az állományok véletlenszerű károsodása
- Számítógépek teljes megsemmisülése (például tűz által), belértve a közelében tárolt összes biztonsági mentést

Tökéletesen megoldható, hogy egyes rendszerek a fentebb felsorolt problémák mindegyikét teljesen eltérő technikával oldják meg. A nagyon személyes rendszerektől és a nagyon értéktelen adatoktól eltekintve szinte egyértelműen kizárt, hogy egyetlen technika képes lefedni az összes problémát.

Kelléktárunk néhány alapvető eszköze:

- Az egész rendszer mentése, amit egy megbízható helyre elzárt, tartós adattárolóra készítünk. Ez tulajdonképpen védelmet biztosít a fentebb megemlített összes probléma esetében, de lassú és kényelmetlen róla visszaállítani az adatokat. A közelben és/vagy neten is tarthatunk erről másolatokat, de még így is kényelmetlen az állományok visszaállítása, különösen az egyszerű felhasználók számára.
- Pillanatképek készítése az állományrendszerrel. Ez valójában csak olyan esetekben lehet a segítségünkre, amikor véletlenül töröltünk állományokat, ám ilyenkor *határozottan* jól jön, mivel igen gyorsan és könnyen lehet vele dolgozni.
- Az egész állományrendszer és/vagy az összes lemez másolata (például az rsync(1) időszakos alkalmazása a komplett gépre). Az általában az egyedi igényekkel bíró hálózatok esetében eshet a kezünkre. A lemezhiba ellen védelemben ez a megoldás általában a RAID alatt áll. A véletlenül törölt állományok visszaállításának tekintetében az UFS pillanatképeivel mérhető össze, de ez leginkább a saját igényeinktől függ.

- RAID alkalmazása. A lemezek meghibásodása esetén segíti minimalizálni vagy elkerülni a kiesést, ugyan gyakori lemezhibák árán (mivel ilyenkor több lemezt használunk) de kisebb sürgősséggel.
- Az állományok ujjenyomatának ellenőrzése. Az mtree(8) segédprogram nagyon hasznos tud lenni ebben az esetben. Habár ez nem egy mentési technika, mégis segít megállapítani, hogy mikor kell nyugdíjba küldenünk a biztonsági mentéseinket. Ez különösen az aktív nem használt mentésekre vonatkozik, ezeket bizonyos idő elteltével mindig érdemes ellenőrizni.

Nagyon könnyű lenne további technikákat is felsorolni, melyek legtöbbször az iméntiek valamilyen kombinációja lenne. A speciális igények általában speciális technikákat eredményeznek (például egy éles adatbázis biztonsági mentése általában az adott adatbáziskezelő rendszer közreműködését is elvárja). Mindig fontos tudni, hogy milyen veszélyek ellen védekezünk és hogyan kezeljük le ezeket.

18.12. Alapvető tudnivalók a biztonsági mentésről

A dump(8), tar(1) és cpio(1) a három legfontosabb biztonsági mentésekkel kapcsolatos program.

18.12.1. Mentés és helyreállítás

A UNIX típusú rendszerekben a biztonsági mentést hagyományosan a dump és restore programok végzik. A meghajtókat lemezblokkok összeségeként kezelik, az állományrendszerek által létrehozott állományok, linkek és könyvtárak szintje alatt. Eltérően más, biztonsági mentést végző szoftverektől, a dump az adott eszközön egy egész állományrendszert képes lementeni. Nem képes csak az állományrendszer vagy egy több állományrendszerre kiterjedő könyvtárszerkezet egy részét lementeni. A dump nem állományokat és könyvtárakat ír a szalagra, hanem nyers adatblokkokat, amelyek állományokat és könyvtárakat formáznak. A restore parancs az adatokat alapértelmezés szerint a /tmp könyvtárba tömöríti ki. Ha nem lenne elegendő helyünk a /tmp könyvtárban, akkor a TMPDIR környezeti változó átállításával ehelyett megadhatunk egy olyat, ahol már kellő mennyiségű terület áll rendelkezésre a restore akadálytalan lefutásához.

Megjegyzés: Ha a dump parancsot a gyökér könyvtárban adjuk ki, akkor nem fogja lementeni a /home vagy /usr vagy bármilyen más könyvtárat, mivel ezek jellemző módon más állományrendszerek csatlakozási pontja vagy más állományrendszerekre mutató szimbolikus linkek.

A dump parancsnak vannak olyan rigolyái, amelyek még az AT&T UNIX 6. verziójából (1975 környékéről) maradtak vissza. Az alapértelmezett paraméterezése 9 sávú szalagokat feltételezi (6250 bpi), nem pedig a napjainkban elterjedt nagy írássűrűségű (egészen 62 182 ftpi-s) adathordozókat. Ezek az alapértelmezések természetesen parancsorból felülbírálhatóak, és így a manapság alkalmazott szalagos meghajtók teljes kapacitása is kihasználható vele.

Emellett az rdump és rrestore programok segítségével hálózaton keresztül is le tudjuk menteni az adatainkat egy másik számítógépre csatlakoztatott szalagos egységre. Mind a két program az rcmd(3) és a ruserok(3) parancsokat használja a távoli szalagos meghajtó eléréséhez. Az rdump és rrestore paramétereinek a távoli számítógép használatához kell illeszkedniük. Amikor egy FreeBSD rendszerű számítógépet az rdump parancssal egy Sun rendszerű, komodo nevű számítógépre mentünk, amelyhez egy Exabyte szalagos meghajtó csatlakozik, akkor ezt a írjuk be:

```
# /sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nsa8 /dev/da0a 2>&1
```

Figyelem: az `.rhosts` állományon keresztül hitelesítésnek megvannak a maga biztonsági kockázatai. Ne felejtjük el felmérni ezt a saját környezetünkben sem.

A `dump` és `restore` parancsokat az `ssh` használatával még biztonságosabbá tehetjük.

Példa 18-1. A `dump` használata az `ssh` alkalmazással

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh -c blowfish \
    célfelhasználó@cél.gép.hu dd of=/nagyállományok/dump-usr-10.gz
```

Vagy az `RSH` környezeti változó megfelelő beállításával használhatjuk a `dump` beépített módszerét:

Példa 18-2. A `dump` használata az `ssh` alkalmazással, az `RSH` környezeti változó beállításával

```
# RSH=/usr/bin/ssh /sbin/dump -0uan -f célfelhasználó@cél.gép.hu:/dev/sa0 /usr
```

18.12.2. `tar`

A `tar(1)` is az AT&T UNIX 6. verziójáig nyúlik vissza (tehát nagyjából 1975-ig). A `tar` az állományrendszerrel szoros együttműködésben dolgozik, állományokat és könyvtárakat ír a szalagra. A `tar` ugyan nem ismeri a `cpio(1)` által felkínált összes lehetőséget, de nincs is szüksége olyan szokatlan parancssoros összekapcsolásokra, mint a `cpio` parancsoknak.

A FreeBSD 5.3 vagy későbbi változataiban a GNU `tar` és az alapértelmezés szerinti `bsdtar` egyaránt elérhető. A GNU változat a `gtar` paranccsal hívható meg. Az `rdump` parancshoz hasonló felírásban képes kezelni a távoli eszközöket. Tehát így tudjuk használni a `tar` parancsot a `komodo` nevű Sun számítógép Exabite szalagos meghajtójának elérésére:

```
# /usr/bin/gtar cf komodo:/dev/nsa8 . 2>&1
```

Ugyanez eltérhető a `bsdtar` használatával is, amikor az `rsh` programmal összekapcsolva küldünk át a távoli szalagos egységre.

```
# tar cf - . | rsh hálózati-név dd of=szalagos-eszköz obs=20b
```

Ha a hálózaton keresztül mentés során fontos számunkra a biztonság, akkor az `rsh` parancs helyett az `ssh` parancsot használjuk.

18.12.3. `cpio`

A `cpio(1)` eredetileg a UNIX szalagos programjai és szalagos egységei között közvetített. A `cpio` parancs (többek közt) képes a byte-ok sorrendjének felcserélésére, több különböző archívum formátuma szerint írni és adatokat közvetíteni más programok felé. Ez utóbbi lehetősége miatt a `cpio` kiválóan alkalmas a telepítőeszközök számára. A `cpio` nem képes bejárni a könyvtárszerkezetet, és az állományok listáját a szabványos bemeneten keresztül kell megadni neki.

A `cpio` nem támogatja a biztonsági mentés átküldését a hálózaton. Programok összekapcsolásával és az `rsh` használatával tudunk adatokat küldeni távoli szalagos meghajtókra.

```
# for f in könyvtár_lista; do
```

```
find $f >> mentési.lista
done
# cpio -v -o --format=newc < backup.list | ssh felhasználó@gép "cat > mentőeszköz"
```

Ahol a *könyvtár_lista* a menteni kívánt könyvtárak listája, a *felhasználó@gép* a mentést végző gép felhasználójának és hálózati nevének együttese, valamint a *mentőeszköz*, ahova a mentés kerül (például */dev/nsa0*).

18.12.4. pax

A *pax(1)* az IEEE/POSIX válasza a *tar* és *cpio* programokra. Az évek során a *tar* és a *cpio* különböző változatai egy kissé inkompatibilissé váltak. Ezért a szabványosításuk kiharcolása helyett inkább a POSIX létrehozott egy új archiváló segédprogramot. A *pax* megpróbálja írni és olvasni a *cpio* és *tar* formátumok legtöbb változatát, valamint emellett további saját formátumokat is kezel. A parancskészlete inkább a *cpio* parancsára emlékeztet, mintsem a *tar* parancsára.

18.12.5. Amanda

Az **Amanda** (Advanced Maryland Network Disk Archiver) egy kliens-szerver alapú mentési rendszer, nem pedig egy önálló program. Az **Amanda** szerver menti tetszőleges számú számítógép adatát egyetlen szalagra, melyek az **Amanda** klienst futtatják és hálózaton keresztül hozzá csatlakoznak. A nagy mennyiségű és nagy kapacitású lemezekkel rendelkező rendszerekben közvetlenül a mentéshez szükséges idő nem áll rendelkezésre a feladat elvégzéséhez. Az **Amanda** viszont képes megoldani ezt a problémát. Az **Amanda** képes egy "saját lemez" használatával egyszerre több állományrendszerről is biztonsági mentést készíteni. Az **Amanda** "archívumkészleteket" hoz létre: az **Amanda** konfigurációs állományában megadott állományrendszerekről készít teljes mentést egy adott idő alatt egy adott mennyiségű szalagra. Az "archívumkészlet" ezenkívül még tartalmaz egy napi inkrementális (vagy különbözeti) mentést is minden egyes állományrendszerről. A sérült állományrendszerek visszaállításához mindig a legújabb teljes biztonsági mentésre és a hozzá tartozó inkrementális mentésekre van szükségünk.

A konfigurációs állomány segítségével precíz irányítást gyakorolhatunk a létrehozott mentések és az **Amanda** által keltett hálózati forgalom felett. Az **Amanda** a fentiek közül bármelyik programmal képes az adatokat szalagra rögzíteni. Az **Amanda** portként vagy csomagként is elérhető, alapértelmezés szerint nem települ.

18.12.6. Ne csináljunk semmit

A "Ne csináljunk semmit" nem egy újabb számítógépes program, hanem egy igen gyakran alkalmazott mentési stratégia. Nem kell beruházni. Nem kell semmilyen biztonsági mentési rendet követni. Egyszerűen semmit se csinálunk. Ha véletlenül valami történne az adatainkkal, akkor csak mosolyogjunk és törődünk bele!

Amennyiben az időnk és adataink keveset vagy éppen semmit se érnek, akkor a "Ne csináljunk semmit" az elérhető legjobb biztonsági mentési megoldás számítógépünk számára. De legyünk óvatosak, mert a UNIX egy igen hasznos eszköz, és fél éven belül könnyen úgy találhatjuk magunkat, hogy mégis csak vannak értékes adataink.

A "Ne csináljunk semmit" tökéletesen megfelelő mentési módszer a */usr/obj* és a hozzá hasonló módon a számítógépen automatikusan generált könyvtárak és állományok esetében. Ugyanilyen példa lehetne a kézikönyv HTML vagy PostScript változata. Ezek a formátumok ugyanis az SGML források alapján keletkeznek, így a HTML vagy PostScript állományok mentése nem életbevágó. Az SGML állományokat viszont már annál inkább mentsük!

18.12.7. Melyik a legjobb?

`dump(8)` Pont. Elizabeth D. Zwicky komolyan letesztelte az itt felsorolt összes programot. A UNIX állományrendszerek jellegzetességeinek és rajtuk az összes adatunk megőrzésének egyértelműen a `dump` felel meg a legjobban. Elizabeth a minden egyes program tesztjéhez olyan állományrendszereket hozott létre, amelyek rengeteg különféle szokatlan helyzetet tartalmaztak (valamint néhány nem annyira szokatlant). Az érintett jellegzetességek: lyukas állományok, lyukas állományok és egy halom nulla, állományok érdekes karakterekkel a nevükben, olvashatatlan és írhatatlan állományok, eszközök, a mentés közben méretüket változtató állományok, a mentés közben keletkező és megszűnő állományok és még sok minden más. Az eredményeit a LISA V-ben jelentette meg 1991 októberében. Lásd A biztonsági mentéshez és archiváláshoz használt programok tesztje (angolul) (<http://www.coredumps.de/doc/dump/zwicky/testdump.doc.html>).

18.12.8. Az adatok helyreállítása vészhelyzetben

18.12.8.1. A katasztrófa előtt

Csupán négy lépést kell megtennünk az esetleges katasztrófák bekövetkezésének esetére.

Először is két példányban nyomtassuk ki az egyes lemezek lemezcímekjét (például a `bsdlabeled da0 | lpr` paranccsal) valamint az állományrendszerek táblázatát (az `/etc/fstab` állományt) és az összes rendszerindításkor megjelenő üzenetet.

A második lépésben készítenünk kell egy “élő” rendszerrel rendelkező CD-lemezt. Ezen a lemezen megtalálható minden, ami el tudunk indítani egy helyreállításhoz elegendő rendszert. Ekkor a felhasználó futtatni tudja például a `dump(8)`, `restore(8)`, `fdisk(8)`, `bsdlabeled(8)`, `newfs(8)`, `mount(8)` és a többi segédprogramot. Ez az image a FreeBSD/i386 8.1-RELEASE kiadáshoz az

<ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.1/FreeBSD-8.1-RELEASE-i386-livefs.iso> címről tölthető le.

A harmadik lépésben igyekezzünk minél gyakrabban szalagra menteni. Mindig gondoljuk arra, hogy a legutolsó mentés óta létrehozott változtatásaink teljesen el fognak veszni. A mentéseket tartalmazó szalagokat tegyük írásvédetté.

A negyedik lépésben ellenőrizzük a a második lépésben készített helyreállító lemezünket és a biztonsági mentéseket tartalmazó szalagokat. Jegyezzük le az eljárást. Ezeket a jegyzeteket is rakjuk el rendszerindító lemezzel, a kinyomtatott adatokkal és a mentéseket tartalmazó szalagokkal együtt. Ezek a jegyzetek megvédenek minket attól, hogy a helyreállítás közbeni kétségbeesésünkben nehogy véletlenül tönkretegyük a biztonsági mentéseinket. (Hogy miként is? Például ha a `tar xvf /dev/sa0` parancs helyett izgalomunkban a `tar cvf /dev/sa0` parancsot gépeljük be, akkor azzal felülírjuk a biztonsági mentéseinket).

A fokozott biztonság kedvéért minden alkalommal készítsünk rendszerindító lemezt és legalább két mentést. Az egyiket valamilyen távoli helyen tároljuk. Ez a távoli hely NE ugyanannak az épületnek az alagsora legyen! Számos cég alaposan megtanulta ezt a szabályt a Világkereskedelmi központ tragédiája kapcsán. Ez a távoli hely számítógépeinkből és merevlemez megajtóinkól is fizikailag jól elkülöníthető, jelentős távolságban legyen.

18.12.8.2. A katasztrófa után

Az alapvető kérdés: a hardver túlélte? Ha rendszeresen készítettünk biztonsági mentéseket, akkor a szoftverek miatt egyáltalán nem kell aggódnunk.

Ha a hardver megsérült, akkor a számítógép használatának újból megkezdése előtt javasolt cserélni a meghibásodott alkatrészeket.

Ha a hardverrel minden rendben találtunk, akkor helyezzük be a helyreállításhoz használatos “élő” rendszert tartalmazó lemezt a CD-meghajtóba, és indítsuk el vele a számítógépet. Ezután nemsokára a telepítési menü jelenik meg. Itt a megfelelő ország után a **Fixit -- Repair mode with CDROM/DVD/floppy or start a shell** (“Helyreállítás CD/DVD/floppy használatával, vagy parancssor indítása”), majd a **CDROM/DVD -- Use the live filesystem CDROM/DVD** (“A CD/DVD-n található élő rendszer használata”) menüpontokat válasszuk. A **restore** és a többi segédprogram a `/mnt2/rescue` könyvtárban lesznek elérhetőek.

Egyenként állítsuk vissza az egyes állományrendszereket.

A `mount` parancssal próbáljuk meg csatlakoztatni az első lemezünk rendszerindító partícióját (például `mount /dev/da0a /mt`). Ha a lemezcímke megsérült, akkor `bsdlabel` alkalmazásával partícionáljuk újra a lemezt és címkézzük meg a korábban kinyomtatott címke adatainak megfelelően. A `newfs` segítségével újra hozzuk létre az állományrendszereket. Írható-olvasható módban csatlakoztassuk újra a lemez rendszerindító partícióját (`mount -u -o rw /mnt`). A biztonság mentést végző program és a biztonsági mentést tartalmazó szalagok használatával állítsuk helyre az állományrendszer tartalmát (például `restore vrf /dev/sa0`). Válasszuk le az állományrendszert (például `umount /mnt`). Mindegyik sérült állományrendszerre ismételjük a folyamatot.

Ahogy működőképpé vált a rendszerünk, mentsük az adatainkat új szalagokra. Akármilyen is okozta a rendszer összeomlását vagy az adatvesztést, ismét lecsaphat. Ha most áldozunk erre még egy órát, akkor azzal a későbbiekben számos kellemetlenségtől óvhatjuk meg magunkat.

18.13. Hálózat, memória és állomány alapú állományrendszerek

Átdolgozta és feljavította: Marc Fonvieille.

A számítógépünkben létező fizikai lemezek, például floppyk, CD-k, merevlemezek és egyéb mellett a lemezek egy másik formáját is képes megérteni a FreeBSD — a *virtuális lemezeket*.

A virtuális lemezek tekinthetőek többek közt az olyan hálózati állományrendszerek, mint például a Hálózati állományrendszer (Network File System, NFS) és a Coda, valamint a memóriában és állományokban létrehozott állományrendszerek.

Attól függően, hogy a FreeBSD melyik változatát használjuk, az állomány és memória alapú állományrendszerek létrehozásához, illetve használatához különböző segédprogramokra lesz szükségünk.

Megjegyzés: A `devfs(5)` a felhasználó számára láthatatlan módon hozza létre az eszközök leíróit.

18.13.1. Állomány alapú állományrendszerek

FreeBSD alatt az `mdconfig(8)` segédprogram segítségével tudunk memórialemezeket (`md(4)`) beállítani és engedélyezni. Az `mdconfig(8)` használatához be kell töltenünk az `md(4)` modult vagy hozzá kell tennünk a rendszermagunk beállításait tartalmazó állományhoz:

```
device md
```

Az `mdconfig(8)` parancs háromféle memória alapú virtuális lemezt ismer: a `malloc(9)`, állományok vagy lapozóterület használatával létrehozott memórialemezeket. Így lehet például csatlakoztatni a floppyk vagy CD-k állományokban tárolt image-eit.

Egy meglevő állományrendszer image-ének csatlakoztatása:

Példa 18-3. Egy meglevő állományrendszer image-ének csatlakoztatása az `mdconfig` parancssal

```
# mdconfig -a -t vnode -f image -u 0
# mount /dev/md0 /mnt
```

Új állományrendszer létrehozása az `mdconfig(8)` használatával:

Példa 18-4. Új állomány alapú lemez létrehozása az `mdconfig` parancssal

```
# dd if=/dev/zero of=új-image bs=1k count=5k
5120+0 records in
5120+0 records out
# mdconfig -a -t vnode -f új-image -u 0
# bsdlable -w md0 auto
# newfs md0a
/dev/md0a: 5.0MB (10224 sectors) block size 16384, fragment size 2048
    using 4 cylinder groups of 1.25MB, 80 blks, 192 inodes.
super-block backups (for fsck -b #) at:
    160, 2720, 5280, 7840
# mount /dev/md0a /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md0a      4710    4  4330    0%    /mnt
```

Ha az `-u` beállítással nem adjuk meg az egység számát, akkor az `mdconfig(8)` az `md(4)` automatikus kiosztásán keresztül fog egy használatban még nem levő eszközt kiválasztani. Az így kiosztott egység neve az `md4` névhez hasonlóan jelenik meg a szabványos kimeneten. Az `mdconfig(8)` használatának részleteiről olvassuk el a hozzá tartozó man oldalt.

Az `mdconfig(8)` egy nagyon sokoldalú segédeszköz, habár használatakor viszonylag sok parancsot kell kiadni egy állomány alapú állományrendszer létrehozásához. A FreeBSD azonban alpból tartalmaz még egy `mdmfs(8)` nevű segédprogramot is, ami az `md(4)` lemezeket az `mdconfig(8)` segítségével állítja be, létrehoz rajtuk egy UFS típusú állományrendszert a `newfs(8)` segítségével és csatlakoztatja a `mount(8)` parancssal. Így például, ha az iménti állományrendszert akarjuk létrehozni és csatlakoztatni, akkor egyszerűen csak gépeljünk be ennyit:

Példa 18-5. Állomány alapú lemezek beállítása és csatlakoztatása az `mdmfs` parancssal

```
# dd if=/dev/zero of=új-image bs=1k count=5k
5120+0 records in
5120+0 records out
# mdmfs -F új-image -s 5m md0 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md0      4718    4  4338    0%    /mnt
```

Ha az `md` paramétert az egység száma nélkül adjuk meg, akkor `mdmfs(8)` az `md(4)` automatikus kiosztására támaszkodva fog egy addig még nem használt eszközt kiválasztani. A `mdmfs(8)` használatának pontos részleteivel kapcsolatban lásd a hozzá tartozó man oldalt.

18.13.2. Memória alapú állományrendszerek

A memória alapú állományrendszerek esetében általában a “lapozóállomány alapú” megközelítést alkalmazzák. A lapozóállomány alapúság nem arra utal, hogy a memórialemezt alpból kilapozzák lemezre, hanem inkább arra, hogy a memórialemez olyan területen jön létre, amelyet szükség esetén lemezre lehet lapozni. Memória alapú lemezeket a (rendszermag szintű) `malloc(9)` használatával is létre lehet hozni, de a `malloc` alapú memórialemezeknél, különösen a nagyon nagyok esetében, a rendszer könnyen össze tud omlani, ha kifut a rendelkezésére álló memóriából.

Példa 18-6. Új memória alapú lemez létrehozása az `mdconfig` paranccsal

```
# mdconfig -a -t swap -s 5m -u 1
# newfs -U md1
/dev/md1: 5.0MB (10240 sectors) block size 16384, fragment size 2048
      using 4 cylinder groups of 1.27MB, 81 blks, 192 inodes.
      with soft updates
super-block backups (for fsck -b #) at:
 160, 2752, 5344, 7936
# mount /dev/md1 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md1      4718    4 4338    0%    /mnt
```

Példa 18-7. Új memória alapú lemez létrehozása az `mdmfs` paranccsal

```
# mdmfs -s 5m md2 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md2      4846    2 4458    0%    /mnt
```

18.13.3. Memórialemezek leválasztása a rendszerről

Amikor már nem akarunk tovább használni egy memória vagy állomány alapú állományrendszert, érdemes visszaadnunk az általuk felhasznált erőforrásokat a rendszernek. Elsőként válasszuk le magát az állományrendszert, majd az `mdconfig(8)` segítségével kapcsoljuk le a lemezt a rendszerről és szabadítsuk fel az általa felhasznált erőforrásokat.

Például az `/dev/md4` eszközt így lehet lekapcsolni és felszabadítani:

```
# mdconfig -d -u 4
```

A beállított `md(4)` eszközökkel kapcsolatos többi információt az `mdconfig -l` paranccsal tudjuk lekérdezni.

18.14. Az állományrendszerek pillanatképei

Írta: Tom Rhodes.

A FreeBSD a Soft Updates mellett felkínál egy másik lehetőséget: az állományrendszerekről készíthető pillanatfelvételeket.

Ezek a pillanatképek lehetővé teszik a felhasználók számára, hogy adott állományrendszerekről képeket hozzanak létre és azt állományként kezeljék. A pillanatképeket az adott állományrendszerben kell létrehozni, és a felhasználók állományrendszerenként hűsznál többet nem hozhatnak belőlük létre. Az aktív pillanatképek a szuperblokkban kerülnek rögzítésre, ezért az állományrendszerek leválasztása és újracsatlakoztatása esetén is megmaradnak, még újraindítás után is. Amikor egy pillanatképre már nincs tovább szükségünk, egy szimpla `rm(1)` paranccsal eltávolítható. A pillanatképek tetszőleges sorrendben eltávolíthatóak, habár ilyenkor az összes általuk lefoglalt hely nem szabadul fel, mivel más pillanatképeknek még szüksége lehet bizonyos blokkjaira.

Miután az `mksnap_ffs(8)` paranccsal létrehoztunk egy pillanatképet tartalmazó állományt, beállítódik rá a módosíthatatlanságot jelentő `snapshot` állományjelző. Egyedül az `unlink(1)` parancs képez ez alól kivételt, mivel segítségével a pillanatképek eltávolíthatóak.

A pillanatképek a `mount(8)` paranccsal hozhatóak létre. A következő módon tudjuk a `/var` egy pillanatképét elkészíteni a `/var/snapshot/snap` állományban:

```
# mount -u -o snapshot /var/snapshot/snap /var
```

Vagy a `mksnap_ffs(8)` meghívásával is készíthetünk pillanatképeket:

```
# mksnap_ffs /var /var/snapshot/snap
```

Az állományrendszeren (például `/var`) a pillanatképeket tartalmazó állományokat a `find(1)` paranccsal kereshetjük meg:

```
# find /var -flags snapshot
```

Ahogy elkészítettünk egy pillanatképet, több mindenre is felhasználhatjuk:

- Egyes rendszergazdák a pillanatképeket biztonsági mentésekhez használják, mivel ezek gond nélkül áttehetőek CD-re vagy szalagra.
- Az állományrendszerek sértetlenségét ellenőrző program, az `fsck(8)` is lefuttatható egy ilyen pillanatképen. Feltéve, hogy az állományrendszer csatlakoztatásakor tiszta volt, mindig egy tiszta (és változásokat nem tartalmazó) eredményt kell kapnunk. Ennek megléte elengedhetetlen a háttérben futtatható `fsck(8)` működéséhez.
- Futassuk le a `dump(8)` segédprogramot a pillanatképen. Az így létrehozott mentés megegyezik az állományrendszer adott pillanatban felvett állapotával. Az `-L` beállítás megadásával maga a `dump(8)` is képes egyetlen parancsban pillanatfelvételt készíteni, ebből létrehozni a mentést, majd eltávolítani.
- A pillanatképet képesek vagyunk a `mount(8)` paranccsal az állományrendszer befagyasztott változataként csatlakoztatni:

```
# mdconfig -a -t vnode -f /var/snapshot/snap -u 4
# mount -r /dev/md4 /mnt
```

Így már a `/mnt` könyvtárba csatlakoztatva be tudjuk járni a befagyasztott `/var` állományrendszert. Minden a pillanatfelvétel készítésének időpontjának megfelelő állapotban fog maradni. Az egyetlen kivétel talán annyi, hogy

korábbi pillanatképek nulla méretű állományként fognak megjelenni. Mikor befejeztük a pillanatképek használatát, a `umount(8)` paranccsal le tudjuk választani:

```
# umount /mnt
# mdconfig -d -u 4
```

A `softupdates` és az állományrendszerek pillanatképeinek használatával, illetve műszaki leírásukkal kapcsolatban látogassuk meg Marshall Kirk McKusick honlapját a <http://www.mckusick.com/> címen (angolul).

18.15. Az állományrendszerek kvótái

A kvóták használata az operációs rendszerben egy olyan választható lehetőség, aminek segítségével állományrendszerenként korlátozni tudjuk az egyes felhasználók vagy csoporttagok által elhasznált lemezterület és/vagy állományok mennyiségét. Ezt leggyakrabban olyan időosztásos rendszerekben használják ki, ahol szükség lehet az egyes felhasználókra vagy csoportokra eső erőforrások mennyiségének szabályozására. Ezzel tudjuk megakadályozni, hogy a felhasználók vagy csoportok elfogyassák az összes rendelkezésre álló lemezterületet.

18.15.1. A kvóták használatának beállítása

Mielőtt nekilátnánk a kvóták használatának, meg kell győződnünk róla, hogy a rendszermagunkban megvan hozzá a szükséges támogatás. A kvótákat a következő sorral lehet engedélyezni a rendszermag beállításait tartalmazó állományban:

```
options QUOTA
```

A gyári `GENERIC` rendszermag ezt alapból nem engedélyezi, ezért ehhez mindenképpen be kell állítani, le kell fordítani és telepíteni egy kell saját rendszermagot. A saját rendszermag létrehozásához kövessük a 8 fejezet utasításait.

Ha ezzel megvagyunk, akkor a következő sorral bővítjük ki az `/etc/rc.conf` állományt:

```
enable_quotas="YES"
```

A kvótákat kezelő rendszer indításának finomabb szabályozására létezik még egy további beállítási lehetőség is. A rendszer indítása során általában az egyes állományrendszerek kvótáját a `quotacheck(8)` program ellenőrzi. A `quotacheck(8)` gondoskodik róla, hogy a kvótákat tároló adatbázis ténylegesen az állományrendszeren található adatokat tükrözi. Ez egy nagyon időigényes folyamat, ami rányomja bélyegét a rendszer elindulásához szükséges idő mennyiségére is. Amennyiben szeretnénk megtakarítani ezt a lépést, tegyük bele az `/etc/rc.conf` állományba a direkt erre a célra kialakított beállítást:

```
check_quotas="NO"
```

Végezetül az állományrendszereken az `/etc/fstab` megfelelő módosításával tudjuk egyenként engedélyezni a lemezkvóták használatát. Itt lehet bekapcsolni az állományrendszerek felhasználókra vagy csoportokra, esetleg mind a kettőjükre vonatkozó kvotáikat.

Ha felhasználói szintű kvótákat akarunk engedélyezni egy állományrendszeren, akkor az `/etc/fstab` állományban az állományrendszer beállításai közé vegyük fel a `userquota` opciót. Például így:

```
/dev/dals2g /home ufs rw,userquota 1 2
```

Ehhez hasonlóan tudjuk engedélyezni a `userquota` helyett a `groupquota` opció használatával a csoportszintű kvótákat is. A felhasználói- és csoportszintű kvóták együttes engedélyezéséhez így kell átírni az állományrendszer bejegyzését:

```
/dev/dals2g    /home    ufs rw,userquota,groupquota 1 2
```

Alapértelmezés szerint az állományrendszerekhez tartozó kvóták a gyökerükben található `quota.user` valamint `quota.group` állományokban tárolódnak. Erről részletesebben az `fstab(5)` man oldalon olvashatunk. Noha még az `fstab(5)` man oldala szerint is megadható más elérési út a kvótákat tároló állományokhoz, semmiképpen sem javasoljuk ezt, mert úgy tűnik, hogy a kvótákat kezelő különböző segédprogramok ezzel nem képesek rendesen megbirkózni.

Most kell újraindítani a rendszerünket az új rendszermaggal. Az `/etc/rc` magától le fogja futtatni a kezdeti kvótaállományok létrehozásához szükséges parancsokat az `/etc/fstab` állományban megadott állományrendszereken. Ennek megfelelően tehát nem nekünk kell kézzel létrehoznunk ezeket az állományokat.

Hétköznapi esetben egyáltalán nem kell manuális futtatnunk a `quotacheck(8)`, `quotaon(8)` vagy `quotaoff(8)` parancsokat. Habár ha tisztában szeretnénk lenni a pontos működésükkel, akkor mindenképpen lapozzuk fel a hozzájuk tartozó man oldalakat.

18.15.2. A kvóták beállítása

Ahogy sikerült beállítani a kvóták használatát, egyből ellenőrizzük is a működőképességüket. Ezt legegyszerűbben a következő paranccsal tehetjük meg:

```
# quota -v
```

Itt egy sorban összefoglalva láthatjuk a jelenlegi lemezhasználatot és az egyes állományrendszereken engedélyezett kvóták korlátait.

Most már készenállunk arra, hogy az `edquota(8)` paranccsal végre korlátokat is beállítsunk a kvótákhoz.

Számos beállítás áll rendelkezésünkre a felhasználók vagy csoportok által lefoglalható lemezterület vagy a létrehozható állományok számának korlátozását illetően. A helyfoglalást szabályozhatjuk lemezterület alapján (blokk kvóta) vagy az állományok száma szerint (állományléíró kvóta), esetleg a kettő kombinációjával. A korlátok további két kategóriára bonthatóak: erősre és gyengére.

Az erős korlátot (hard limit) nem lehet túllépni. Ahogy a felhasználó eléri a számára kiszabott erős korlátot, semmilyen további területet nem használhat fel a kérdéses állományrendszeren. Például, ha a felhasználónak az állományrendszeren 500 kilobyte-os erős korlátot állítottunk be, és éppen 490 kilobyte-nál tart, akkor a felhasználó innen már csak 10 kilobyte-nyi helyet foglalhat le. 11 kilobyte lefoglalása már nem fog sikerrel járni.

Ezzel szemben a gyenge korlátok (soft limit) egy adott ideig átléphetőek. Ezt az időt türelmi időnek (grace period) nevezik, ami alapértelmezés szerint egy hét. Ha a felhasználó a gyenge korláton felül marad a türelmi idő után is, akkor ezt a gyenge korlát erőssé válik és semmilyen további helyfoglalásra nem lesz lehetősége. Amikor a felhasználók újra a gyenge korlát alá kerül, a türelmi idő is visszaáll a beállított értékére.

A most következő példában az `edquota(8)` parancsot mutatjuk be. Amikor meghívjuk az `edquota(8)` parancsot, akkor elindul az `EDITOR` környezeti változónak megfelelő szövegszerkesztő, illetve ennek hiányában a `vi`, és lehetőségünk nyílik a kvóta korlátainak módosítására.

```
# edquota -u teszt
```

```
Quotas for user teszt:
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
      inodes in use: 7, limits (soft = 50, hard = 60)
/usr/var: kbytes in use: 0, limits (soft = 50, hard = 75)
          inodes in use: 0, limits (soft = 50, hard = 60)
```

Normális esetben minden kvótával rendelkező állományrendszerhez két sort kapunk. Közülük az egyik sorban szerepelnek a blokkok korlátai, a másikban az állományleírók korlátai. Ha valamelyiküket meg akarjuk változtatni, akkor egyszerűen csak át kell írunk az adott korlát értékét. Például növeljük meg a felhasználók 50-es gyenge és 75-ös erős blokk korlátját 500-as gyenge és 600-as erős korlátra. Ehhez szerkesszük át a

```
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
```

sort erre:

```
/usr: kbytes in use: 65, limits (soft = 500, hard = 600)
```

Az új korlátok akkor fognak érvénybe lépni, miután kiléptünk a szövegszerkesztőből.

Néha hasznos lehet a korlátokat adott felhasználói azonosítókhoz beállítani. Ezt az `edquota(8)` parancs `-p` paraméterével tudjuk elvégezni. Először is állítsuk be egy felhasználónak a beállítani kívánt korlátokat, majd futtassuk le az `edquota -p teszt kezdőuid-véguid` parancsot. Például ha a `teszt` nevű felhasználónak állítottuk be a számunkra megfelelő korlátokat, akkor a következő paranccsal lehet a rá vonatkozó korlátokat kiterjeszteni a 10 000 és 19 999 közötti azonosítójú felhasználókra:

```
# edquota -p teszt 10000-19999
```

Erről bővebben az `edquota(8)` man oldalán kaphatunk felvilágosítást.

18.15.3. A kvóták korlátainak és a lemezhasználat ellenőrzése

A kvóták korlátait és a lemez jelenlegi kihasználtságát a `quota(1)` vagy `repquota(8)` parancsokkal is ellenőrizhetjük. A `quota(1)` parancs segítségével ellenőrizhető az egyes felhasználók vagy csoportok kvótája és lemezhasználat. A felhasználók csak a saját adataikhoz férhetnek hozzá, illetve mindazon csoportokéhoz, aminek tagjai. Egyedül a rendszeradminisztrátor képes látni az összes felhasználó és csoport kvótáját. A `repquota(8)` paranccsal kérdezhető le az összes kvóta és lemezhasználat rövid kimutatása minden olyan állományrendszeren, ahol azok engedélyezettek.

A következő kimenet a `quota -v` parancstól származik, ahol a felhasználónak két állományrendszeren is vannak kvótái:

```
Disk quotas for user teszt (uid 1002):
```

Filesystem	usage	quota	limit	grace	files	quota	limit	grace
/usr	65*	50	75	5days	7	50	60	
/usr/var	0	50	75		0	50	60	

A fenti példában látható, hogy a felhasználó a `/usr` állományrendszeren pillanatnyilag 15 kilobyte-tal van az 50 kilobyte-os gyenge korlátja felett és 5 napja van hátra a türelmi időből. Vegyük észre a szám mellett levő csillagot (*), amivel a rendszer jelzi, hogy a felhasználó túllépte a korlátját.

A `quota(1)` parancs kimenetében általában nem jelennek meg azok az állományrendszerek, amelyeken a felhasználónak ugyan vannak kvótái, de nem foglal rajtuk lemezterületet. A `-v` beállítás megadásával ezek az állományrendszerek is láthatóvá válnak, mint ahogy azt a fenti példában is megfigyelhettük a `/usr/var` esetében.

18.15.4. Kvóták NFS-en keresztül

A kvóták az NFS szerver kvótákért felelős alrendszerében is engedélyezhetőek. Az `rpc.rquotad(8)` démon teszi az NFS klienseken futtatott `quota(1)` parancsok számára elérhetővé a kvótákkal kapcsolatos információkat, aminek köszönhetően a felhasználók távolról is képesek lekérdezni a kvótáikat.

Az `rpc.rquotad` aktiválásához a következőt kell beállítani az `/etc/inetd.conf` állományban:

```
rquotad/1          dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

Majd ne felejtsük el újraindítani az `inetd` démon sem:

```
# /etc/rc.d/inetd restart
```

18.16. A lemezpartíciók titkosítása

Írta: Lucky Green.

A FreeBSD kitűnő futásközbeni védelmet ajánl fel az adatok illetéktelen hozzáférése ellen. Az állományok engedélyei és a kötelező hozzáférés-vezérlés (Mandatory Access Control, MAC, lásd 16 fejezet) segítenek megvédeni érzékeny adatainkat az illetéktelenek ellen az operációs rendszer futása és a számítógép működése során. Azonban az operációs rendszerben kezelt engedélyek teljesen hatástalanok abban az esetben, ha a támadó fizikailag is képes hozzáférni a számítógépünkhöz, eltávolítani a merevlemezt és egy másik operációs rendszer segítségével kielemezni a rajta található fontos adatainkat.

Függetlenül attól, hogy a támadó valójában miként is férközzött hozzá a merevlemezünkhöz, vagy miként kapcsolta le a számítógépünket, a FreeBSD megtalálható **GEOM alapú lemeztitkosítás (gbde)** és a `geli` titkosítási alrendszer egyaránt képes védelmet nyújtani a számítógépen található állományrendszerek számára az értékes adatok után kutató igen motivált betörők ellen. A csupán egyes állományokra kiterjedő körmönfont titkosítási módszerekkel szemben a `gbde` és a `geli` az egész állományrendszert észrevétlen módon titkosítja. Titkosítatlan adat nem is kerül a merevlemezre.

18.16.1. A lemez titkosítása a gbde használatával

1. Váljunk `root` felhasználóvá

A **gbde** beállításához rendszeradminisztrátori jogosultságokra lesz szükségünk.

```
% su -
Password:
```

2. Adjuk hozzá a `gbde(4)` támogatását a rendszermag konfigurációs állományához

Tegyük a következő sort a rendszermag beállításait tartalmazó állományba:

```
options GEOM_BDE
```

Fordítsuk újra a rendszermagot a 8 fejezetben leírtak szerint.

Indítsuk el a számítógépet az új rendszermaggal.

3. A rendszermag újrafordítása helyett a `kldload` paranccsal is betölthetjük a `gbde(4)` modulját:

```
# kldload geom_bde
```

18.16.1.1. A titkosított merevlemez előkészítése

A következő példa azt feltételezi, hogy a rendszerünkhöz egy új merevlemez adunk hozzá, amin egyetlen titkosított partíció foglal helyet. Ezt a partíciót a `/private` könyvtárba fogjuk csatlakoztatni. A **gbde** használható a `/home` és a `/var/mail` titkosítására is, de ennek megvalósítása olyan bonyolult utasításokat igényel, amelyek meghaladják ennek a bevezetésnek a kereteit.

1. Az új merevlemez hozzáadása

A 18.3 Szakaszban bemutatottak szerint adjuk hozzá a rendszerünkhöz az új merevlemez. A példában az új lemez partícióját a `/dev/ad4s1c` néven fogjuk tudni elérni. A `/dev/ad0s1*` eszközök a példában szereplő FreeBSD rendszer szabványos partícióit jelölik.

```
# ls /dev/ad*
/dev/ad0          /dev/ad0s1b      /dev/ad0s1e      /dev/ad4s1
/dev/ad0s1        /dev/ad0s1c      /dev/ad0s1f      /dev/ad4s1c
/dev/ad0s1a       /dev/ad0s1d      /dev/ad4
```

2. Hozzunk létre egy könyvtárat a gbde zárolásainak tárolásához

```
# mkdir /etc/gbde
```

A **gbde**nek azért van szüksége a zárolásokat rögzítő állományokra, hogy hozzá tudjon férni a titkosított partíciókhoz. Amennyiben ezt nem tudja megtenni, a **gbde** anélkül nem lesz képes visszafejteni a titkosított partíciókon tárolt adatokat, hogy az ezeket elérni akaró szoftvereknek ne kelljen jelentősebb mértékben manuálisan beavatkozni. Mindegyik titkosított partíció külön zároló állományt használ.

3. A gbde partíció inicializálása

A **gbde** által használt partíciókat használatuk előtt inicializálni kell. Ezt a műveletet azonban csak egyszer kell elvégezni:

```
# gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c.lock
```

A **gbde(8)** ekkor elindít egy szövegszerkesztőt és benne egy sablon segítségével be tudjuk állítani a különböző konfigurációs értékeket. Az UFS1 vagy UFS2 használata esetén állítsuk a szektorméretet 2048-ra:

```
$FreeBSD: src/sbin/gbde/template.txt,v 1.1 2002/10/20 11:16:13 phk Exp $
#
# Sector size is the smallest unit of data which can be read or written.
# Making it too small decreases performance and decreases available space.
# Making it too large may prevent filesystems from working. 512 is the
# minimum and always safe. For UFS, use the fragment size
#
sector_size      =          2048
[...]
```

A megjegyzés fordítása:

A szektorméret az adatok írásának és olvasásának legkisebb egysége. Ha túlságosan kicsire választjuk meg, akkor csökken a teljesítmény és csökken a rendelkezésre álló hely. Ha viszont túlságosan nagyra hagyjuk, akkor azzal akadályozzuk az állományrendszerek munkáját. 512 a legkisebb érték, amely mindig megbízható. Az UFS esetén használjuk a fragmensek méretét.

A **gbde(8)** kétszer is rá fog kérdeni az adatok titkosítására használt jelmondatra. A jelmondatnak természetesen mind a kétszer ugyanannak kell lennie. A **gbde** védelmének hatékonysága teljesen mértékben az általunk választott jelmondat minőségétől függ ¹.

A `gbde init` parancs létrehoz egy zároló állományt a **gbde** partícióhoz, amely ebben a példában az `/etc/gbde/ad4s1c.lock` néven keletkezett. A **gbde** zároló állományainak “.lock” névre kell végződnie, mivel az `/etc/rc.d/gbde` indítóskript csak ebben az esetben észleli rendesen.

FigyelemA **gbde** zároló állományait a titkosított partíciók tartalmával együtt *kell* lementeni. Miközben a zároló állomány törlése nem tudja megakadályozni, hogy az elszánt támadó visszafejtse a **gbde** által titkosított partíciót, addig a zároló állomány nélkül a jogos tulajdonos órási mennyiségű munka befektetése nélkül képtelen lesz hozzáférni a rajta levő adatokhoz. Ez utóbbitól egyébként a `gbde(8)` és a rendszer tervezője is totálisan elhatárolja magát.

4. A titkosított partíció illesztése a rendszermaghoz

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
```

Ekkor a titkosított partíció illesztéséhez a rendszer kérni fogja az inicializálás során választott jelmondatot. Ezután az új titkosított eszköz megjelenik a `/dev` könyvtárban `/dev/eszköznév.bde` néven:

```
# ls /dev/ad*
/dev/ad0          /dev/ad0s1b      /dev/ad0s1e      /dev/ad4s1
/dev/ad0s1        /dev/ad0s1c      /dev/ad0s1f      /dev/ad4s1c
/dev/ad0s1a       /dev/ad0s1d      /dev/ad4          /dev/ad4s1c.bde
```

5. Állományrendszer kialakítása egy titkosított eszközön

Ahogy sikerült a titkosított eszközt illeszteni a rendszermaghoz, létre is tudunk hozni egy állományrendszert rajta. Erre a célra a `newfs(8)` remekül használható. Mivel egy új UFS2 állományrendszerek inicializálása sokkal gyorsabb a régi UFS1 állományrendszerek inicializálásánál, ezért a `newfs(8)` használata esetén az `-O2` beállítás megadása ajánlott.

```
# newfs -U -O2 /dev/ad4s1c.bde
```

Megjegyzés: A `newfs(8)` parancsot egy illesztett **gbde** partíción kell végrehajtani, amit onnan ismerhetünk meg, hogy az eszköz nevében szerepel a `*.bde` kiterjesztés.

6. A titkosított partíció csatlakoztatása

Hozzunk létre egy csatlakozási pontot a titkosított állományrendszer számára.

```
# mkdir /privát
```

Csatlakoztassuk a titkosított állományrendszert.

```
# mount /dev/ad4s1c.bde /privát
```

7. Ellenőrizzük a titkosított állományrendszer működőképességét

A titkosított állományrendszert most már látja a `df(1)` program és készen áll a használatra.

```
% df -H
Filesystem      Size  Used Avail Capacity  Mounted on
/dev/ad0s1a     1037M   72M   883M      8%      /
/devfs          1.0K   1.0K    0B    100%   /dev
/dev/ad0s1f      8.1G   55K   7.5G      0%   /home
/dev/ad0s1e     1037M   1.1M   953M      0%   /tmp
/dev/ad0s1d      6.1G   1.9G   3.7G     35%   /usr
```

```
/dev/ad4s1c.bde    150G    4.1K    138G    0%    /private
```

18.16.1.2. Létező titkosított állományrendszerek csatlakoztatása

A rendszer minden egyes indítása után az összes titkosított állományrendszert tényleges használata előtt újra illeszteni kell a rendszermaghoz, ellenőrizni az épségét és csatlakoztatni. Az ehhez szükséges parancsokat `root` felhasználóként kell kiadni.

1. A `gbde` partíció illesztése a rendszermaghoz

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
```

A `gbde` partíció inicializálása során megadott jelmondatot kell megadnunk a művelet elvégzéséhez.

2. Az állományrendszer épségének ellenőrzése

Mivel a titkosított állományrendszerek az automatikus csatlakoztatáshoz még nem szerepeltethetők az `/etc/fstab` állományban, ezért az ilyen állományrendszereket csatlakoztatásuk előtt manuálisan ellenőriztetni kell a `fsck(8)` lefuttatásával.

```
# fsck -p -t ffs /dev/ad4s1c.bde
```

3. A titkosított állományrendszer csatlakoztatása

```
# mount /dev/ad4s1c.bde /privát
```

A titkosított állományrendszer most már készen áll a használatra.

18.16.1.2.1. A titkosított partíciók önálló csatlakoztatása

Lehet írni olyan szkriptet, amely a titkosított partíciókat magától illeszti, ellenőrzi és csatlakoztatja, de biztonsági megfontolásokból semmi esetben sem szabad tartalmaznia a `gbde(8)` jelszavát. Ehelyett azt javasoljuk, hogy az ilyen szkripteknek külön meg kelljen adni a jelszót konzolon vagy az `ssh(1)` használatán keresztül.

De használhatjuk a mellékelt `rc.d` szkriptet is. A szkript paramétereit az `rc.conf(5)` állományon keresztül adhatjuk meg, például:

```
gbde_autoattach_all="YES"
gbde_devices="ad4s1c"
gbde_lockdir="/etc/gbde"
```

Ilyenkor a `gbde` által használt jelmondatot a rendszer indításakor kell megadni. Miután begépeztük a megfelelő jelmondatot, a titkosított `gbde` partíció magától csatlakoztatásra kerül. Ez akkor lehet hasznos, ha a `gbde` megoldását hordozható számítógépeken alkalmazzuk.

18.16.1.3. A `gbde` által alkalmazott titkosítási módszerek

A `gbde(8)` a szektorok tartalmát 128 bites AES használatával CBC módban titkosítja. A lemezen található minden egyes szektort eltérő AES kulccsal kódolja. A `gbde` kriptográfiai felépítését, valamint mindazt, hogy az egyes szektorok kulcsai miként származtathatóak a felhasználó által megadott jelmondatból, a `gbde(4)` man oldalán olvashatjuk.

18.16.1.4. Kompatibilitási problémák

A `sysinstall(8)` nem kompatibilis a **gbde** által titkosított eszközökkel. A `sysinstall(8)` indítása előtt minden `*.bde` eszközt ki kell iktatni a rendszermagból, különben az eszközök keresése során össze fog omlani. A példánkban használt titkosított eszközt a következő paranccsal kell lekapcsolni:

```
# gbde detach /dev/ad4s1c
```

Továbbá megjegyezzük azt is, hogy a `vinum(4)` nem használja a `geom(4)` alrendszert, ezért a **gbde** alkalmazása során nem használhatunk **Vinum**-köteteket.

18.16.2. A lemezek titkosítása a `geli` használatával

Írta: Daniel Gerzo.

A FreeBSD 6.0 változatától kezdve egy új kriptográfiai GEOM osztály is a rendelkezésünkre áll, melyet pillanatnyilag Pawel Jakub Dawidek fejleszt. A `geli` segédprogram némileg különböző a `gbde` megoldásától — más lehetőségeket kínál fel és a titkosítást is egy eltérő séma mentén valósítja meg.

A `geli(8)` legfontosabb jellemzői a következők:

- A `crypto(9)` keretrendszerét használja — tehát ha rendelkezünk kriptográfiai hardverrel, akkor a `geli` automatikusan használni fogja.
- Több kriptográfiai algoritmust is ismer (melyek jelenleg az AES, Blowfish és a 3DES).
- Segítségével a rendszerindításhoz használt (gyökér) partíció is titkosítható. Ilyenkor a szükséges jelmondatot a rendszer indításakor kell megadni.
- Két független kulcsot (például egy “kulcsot” és egy “céges kulcsot”) is használhatunk vele.
- A `geli` gyors — egyszerűen csak szektorról szektorra titkosít.
- Lehetővé teszi a mesterkulcsok mentését is visszaállítását. Ha a felhasználó véletlenül megsemmisítené a kulcsát, akkor a biztonsági mentésből helyreállított kulcsok segítségével vissza tudjuk szerezni az adatainkat is.
- Segítségével a lemezeket véletlenszerű, egyszeri jelszavakkal is illeszthetjük — ez különösen fontos lapozóterületek és ideiglenes állományrendszerek esetében.

A `geli` által felkínált lehetőségekről a `geli(8)` man oldalán találhatunk többet.

A következő lépések bemutatják, hogyan lehet a FreeBSD rendszermagjában engedélyezni a `geli` támogatását, és hogyan lehet létrehozni és használni egy `geli` titkosítással rendelkező adathordozót.

A `geli` alkalmazásához legalább a FreeBSD 6.0-RELEASE vagy későbbi változatára van szükségünk. Mivel a rendszermagot is módosítanunk kell, ezért rendszeradminisztrátori jogosultságok kellenek a műveletek elvégzéséhez.

1. A `geli` támogatásának hozzáadása a rendszermaghoz

Vegyük hozzá a következő sorokat a rendszermag beállításait tartalmazó állományhoz:

```
options GEOM_ELI
device crypto
```

Fordítsuk újra a rendszermagot a 8 fejezetben leírtak szerint.

Betölthetjük a `geli` modulját is a rendszer indításakor. Ehhez a következő sort kell betenni a `/boot/loader.conf` állományba:

```
geom_eli_load="YES"
```

A `geli(8)` most már használható a rendszermagban.

2. A mesterkulcs legenerálása

A most következő példában egy kulcsot tartalmazó állomány létrehozását illusztráljuk, amit a `/privát` könyvtárba csatlakoztatott titkosított adathordozó mesterkulcsához fogunk használni. A kulcs állomány a mesterkulcs titkosításához felhasznált véletlenszerű adatot fogja tartalmazni, valamint rajta kívül még a mesterkulcsot egy jelmonddal is védjük. Az adathordozó szektormérete 4 kilobyte-os lesz. Emellett még bemutatjuk, hogyan kell illeszteni egy `geli`-adathordozót, állományrendszert létrehozni rajta, csatlakoztatni, dolgozni vele és lekapcsolni.

A nagyobb teljesítmény érdekében javasolt nagyobb szektorméretet választani (mint például 4 kilobyte).

A mesterkulcsot egy jelmonddal fogjuk védeni és a kulcsok készítéséhez használt adatforrás a `/dev/random` lesz. A `/dev/da2.eli`, amelyet mit csak adathordozónak fogunk csak hívni, szektorainak mérete 4 kilobyte lesz.

```
# dd if=/dev/random of=/root/da2.key bs=64 count=1
# geli init -s 4096 -K /root/da2.key /dev/da2
Enter new passphrase:
Reenter new passphrase:
```

Nem kötelező egyszerre használni a jelmondatot és a kulcs állományt. A mesterkulcs elzárásának bebiztosítására bármelyik módszer alkalmas.

Ha a kulcs állomány a `"-"` paraméterrel adjuk meg, akkor a szabványos bemenetről olvassa be a program. Ez a példa több kulcs használatát mutatja be.

```
# cat kulcs1 kulcs2 kulcs3 | geli init -K - /dev/da2
```

3. Az adathordozó illesztése a generált kulccsal

```
# geli attach -k /root/da2.key /dev/da2
Enter passphrase:
```

Az új titkosítatlan eszköz neve `/dev/da2.eli` lesz.

```
# ls /dev/da2*
/dev/da2  /dev/da2.eli
```

4. Az új állományrendszer kialakítása

```
# dd if=/dev/random of=/dev/da2.eli bs=1m
# newfs /dev/da2.eli
# mount /dev/da2.eli /privát
```

A titkosított állományrendszer most már `df(1)` számára is látszik és használható:

```
# df -H
Filesystem      Size  Used Avail Capacity  Mounted on
/dev/ad0s1a     248M   89M  139M    38%      /
/devfs          1.0K   1.0K    0B   100%    /dev
/dev/ad0s1f     7.7G   2.3G   4.9G    32%    /usr
/dev/ad0s1d     989M   1.5M   909M     0%    /tmp
/dev/ad0s1e     3.9G   1.3G   2.3G    35%    /var
/dev/da2.eli    150G   4.1K  138G     0%    /private
```

5. Az adathordozó leválasztása és lekapcsolása

Miután befejeztük a munkát a titkosított partíción, és a `/privát` partícióra már nincs tovább szükségünk, érdemes leválasztanunk és kiiktatnunk a `geli` titkosítású partíciót a rendszermagból.

```
# umount /privát
# geli detach da2.eli
```

A `geli(8)` használatáról bővebben a saját man oldalán tájékozódhatunk.

18.16.2.1. A `geli rc.d` szkriptjének használata

A `geli` mellett találhatunk egy saját `rc.d` szkriptet, amely jelentősen leegyszerűsíti a `geli` használatát. A `geli` például így paraméterezhető az `rc.conf(5)` állományon keresztül:

```
geli_devices="da2"
geli_da2_flags="-p -k /root/da2.key"
```

Ennek segítségével a `/dev/da2` eszközt `geli` adathordozóként állítjuk be a `/root/da2.key` állományban található mesterkulcs felhasználásával, de az illesztéskor a `geli` nem kér jelmondatot (ezt csak akkor fogja tenni, ha a `geli init` parancs kiadásához hozzátesszük a `-p` beállítást). A rendszer leállítása előtt pedig a `geli` adathordozó így automatikusan leválasztásra kerül.

Az `rc.d` beállításával kapcsolatos tudnivalókat a kézikönyv `rc.d` szkriptekről szóló szakaszában ismerhetjük meg.

18.17. A lapozóterület titkosítása

Írta: Christian Briffner.

A FreeBSD-ben a lapozóterület titkosítása nagyon könnyen beállítható és már a FreeBSD 5.3-RELEASE változata óta elérhető. Attól függően, hogy konkrétan a FreeBSD melyik verzióját használjuk, a konfigurációhoz kapcsolódó beállítások némileg eltérhetnek. A FreeBSD 6.0-RELEASE változatától kezdődően a `gbde(8)` és a `geli(8)` alrendszerek is használhatóak a lapozóterület titkosítására. A korábbi verziókban egyedül csak a `gbde(8)` érhető el. Mind a két rendszer az `encswap rc.d` szkriptet használja.

Az előző szakaszban, vagyis a lemezzartók titkosításában már röviden összefoglaltuk a különböző titkosítással foglalkozó alrendszereket.

18.17.1. Miért kellene titkosítanunk a lapozóterületet?

Hasonlóan a lemezzartók titkosításához, a lapozóterület titkosításának is az a célja, hogy védjük az érzékeny információkat. Képzeljük el, hogy egy olyan alkalmazással dolgozunk, amely jelszavakat kezel. Amíg ezek a jelszavak a memóriában maradnak, addig minden a legnagyobb rendben van. Azonban amikor az operációs rendszernek lát a fizikai memória felszabadításához kilapozni ezeket az adatokat, a jelszavak titkosítatlanul kerülnek a lemez felületére és egy támadó számára könnyű prédává válnak. Ilyen helyzetekben csak lapozóterület titkosítása jelenthet megoldást.

18.17.2. Előkészületek

Megjegyzés: A szakasz további részében a `ad0s1b` lesz a lapozásra használt partíció.

Egészen mostanáig nem titkosítottuk a lapozóterületet. Így elképzelhető, hogy a lemezre már titkosítatlanul kikerültek jelszavak vagy bármilyen más érzékeny adatok. A csorba kiköszörülésére a lapozóterületen található összes adatot írjuk felül véletlenszerűen generált szeméttel:

```
# dd if=/dev/random of=/dev/ad0s1b bs=1m
```

18.17.3. A lapozóterület titkosítása a `gbde(8)` használatával

Ha a FreeBSD 6.0-RELEASE vagy újabb változatát használjuk, akkor az `/etc/fstab` állományban tegyük hozzá a `.bde` utótagot az a lapozóterülethez tartozó eszköz nevéhez.

# Device	Mountpoint	FStype	Options	Dump	Pass#
/dev/ad0s1b.bde	none	swap	sw	0	0

A FreeBSD 6.0-RELEASE előtti kiadások esetében a következő sort is hozzá kell tennünk az `/etc/rc.conf` állományhoz:

```
gbde_swap_enable="YES"
```

18.17.4. A lapozóterület titkosítása a `geli(8)` használatával

A `gbde(8)` használatához hasonlóan a `geli(8)` által felajánlott titkosítást is alkalmazhatjuk a lapozóterület védelmére. Ilyenkor az `/etc/fstab` állományban az `.eli` utótagot kell hozzátenni a lapozóterülethez tartozó eszköz névhez.

# Device	Mountpoint	FStype	Options	Dump	Pass#
/dev/ad0s1b.eli	none	swap	sw	0	0

Az `geli(8)` az AES algoritmust alapértelmezés szerint 256 bites kulccsal használja.

Ezek az alapértelmezések megváltoztathatóak az `/etc/rc.conf` állományban a `geli_swap_flags` beállítás használatával. A következő sor arra utasítja az `encswap rc.d` szkriptet, hogy a `geli(8)` és a Blowfish algoritmus használatával hozzon létre egy lapozópartíciót 128 bites kulccsal, 4 kilobyte-os szektormérettel és a “detach on last close” (“lekapcsolás használat után”) beállítással:

```
geli_swap_flags="-e blowfish -l 128 -s 4096 -d"
```

A FreeBSD 6.2-RELEASE verzió előtti rendszerekben a következő sort kell használni:

```
geli_swap_flags="-a blowfish -l 128 -s 4096 -d"
```

A többi beállításhoz a `geli(8)` man oldalán a `onetime` parancs leírását érdemes áttanulmányozni.

18.17.5. Ellenőrizzük a működését

Miután újraindítottuk a rendszert, a titkosított lapozóterület helyes működését a `swapinfo` paranccsal ellenőrizhetjük le.

A `gbde(8)` esetében:

```
% swapinfo
Device          1K-blocks    Used    Avail Capacity
/dev/ad0s1b.bde    542720         0    542720     0%
```

Valamint a `geli(8)` esetében:

```
% swapinfo
Device          1K-blocks    Used    Avail Capacity
/dev/ad0s1b.eli    542720         0    542720     0%
```

Megjegyzések

1. A könnyen megjegyezhető ám mégis biztonságos jelmondatok megválasztásához a Diceware Passphrase (<http://world.std.com/~reinhold/diceware.html>) honlapján találunk egy kis segítséget (angolul).

Fejezet 19. GEOM: A moduláris lemezszervező rendszer

Írta: Tom Rhodes.

19.1. Áttekintés

Ez a fejezet a FreeBSD-ben található GEOM rendszert mutatja be. Ez a rendszer tömöríti az általa is alkalmazott fontosabb RAID-vezérlő segédprogramokat. A fejezet nem részletezi, hogy a GEOM konkrétan milyen módon kezeli és vezérli az I/O-t, ahogy azt sem, hogyan működik az alapjául szolgáló alrendszer vagy hogy néz ki annak forráskódja. Az ilyen jellegű információk a `geom(4)` man oldalon, valamint az ott felsorolt helyeken találhatóak meg. Továbbá, ez a fejezet magukról a RAID-konfigurációkról sem ad pontos tájékoztatást. Kizárólag csak a GEOM által is támogatott RAID-besorolásokról esik szó.

A fejezet elolvasása során megismerjük:

- a GEOM segítségével milyen fajtájú RAID támogatást érhetünk el;
- hogyan kell használni a rendszer által nyújtott alapvető segédeszközöket a különféle RAID-szintek konfigurálásához, karbantartásához és kezeléséhez;
- hogyan kell a GEOM-on keresztül tükrözni, csíkozni, titkosítani és távolról összekapcsolni lemezes eszközöket;
- hogyan kell a GEOM rendszerben összekapcsolt lemezeknél felmerülő hibákat felderíteni.

A fejezet elolvasásához ajánlott:

- megérteni, hogyan kezeli a FreeBSD a lemezes eszközöket (18 fejezet);
- ismerni, hogyan konfiguráljunk és telepítsünk egy új FreeBSD rendszermagot (8 fejezet).

19.2. A GEOM bemutatása

A GEOM rendszer adatszolgáltatókon vagy speciális `/dev`-állományokon keresztül hozzáférést és vezérlést tesz lehetővé bizonyos osztályokhoz — Master Boot Recordokhoz, BSD-címkékhez stb. Számos szoftveres RAID konfiguráció támogatásával a GEOM transzparens elérést tesz lehetővé mind az operációs rendszer, mind pedig az általa felkínált segédprogramok számára.

19.3. RAID0 - Csíkozás

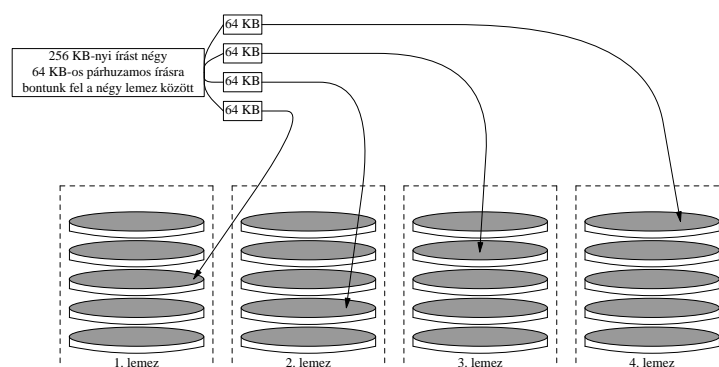
Írta: Tom Rhodes és Murray Stokely.

A csíkozás módszerét használjuk abban az esetben, amikor több lemezmeghajtót akarunk egyetlen kötetűvé összevonni. A GEOM lemezalrendszer szoftveres támogatást nyújt a RAID0, más néven a lemezcsíkozás megvalósításához.

Egy RAID0 rendszerben az adatokat blokkokra bontva írjuk fel a tömbben található lemezek között szétosztva. Így ahelyett, hogy meg kellene várnunk 256 kb-nyi adat egyetlen lemezre írását, egy RAID0 rendszerben egyszerre

íródik 64 kb-nyi adat négy különböző lemezre, és ezáltal gyorsabb elérést szolgáltat. Ez a gyorsaság további lemezvezérlők használatával még jobban fokozható.

Az egy RAID0-csíkozásban résztvevő lemezek mindegyikének azonos méretűnek kell lennie, mivel az írásra és olvasásra irányuló I/O-kérések a párhuzamos kiszolgálás érdekében összefésülődnek.



Csíkozás kialakítása formázatlan ATA-lemezekkel

1. Töltsük be a `geom_stripe.ko` modult:

```
# kldload geom_stripe
```

2. Bizonyosodjunk meg róla, hogy a rendszerünkben található egy szabad csatlakozási pont. Ha majd ezt a kötetet szánjuk rendszerünk gyökérpartíciójának, használjunk erre a célra egy másik könyvtárat, például a `/mnt`-ot:

```
# mkdir /mnt
```

3. Keressük meg a csíkozásra felhasználni kívánt lemezek eszközneveit, és hozzunk létre belőlük egy új csíkozott eszközt. Például, ha két használatban nem levő, particionálatlan ATA-lemezt, név szerint a `/dev/ad2` és `/dev/ad3` eszközöket akarjunk csíkozni:

```
# gstripe label -v st0 /dev/ad2 /dev/ad3
Metadata value stored on /dev/ad2.
Metadata value stored on /dev/ad3.
Done.
```

4. Az így létrejött új köteten most hozzunk létre egy általános címkét, vagy más néven egy partíciós táblát, és telepítsük fel rá a rendszer alapértelmezett rendszerindító programját:

```
# bsdlabel -wB /dev/stripe/st0
```

5. Ezzel meg kellett jelennie további másik két eszköznek is a `/dev/stripe` könyvtárban, a `st0` eszköz mellett. Ezek többek közt az `st0a` és az `st0c`. Itt már ki is tudunk alakítani egy állományrendszert az `st0a` eszközön a `newfs` használatával:

```
# newfs -U /dev/stripe/st0a
```

Sok-sok számot fogunk látni cikázni a képernyőn, majd néhány másodperc múlva befejeződik a folyamat. Létrehoztuk a kötetet, ami most már készen áll a becsatolásra.

A kialakított lemezcsíkozást így tudjuk kézzel csatlakoztatni:

```
# mount /dev/stripe/st0a /mnt
```

A csíkozott állományrendszert a rendszerindítás folyamán automatikusan becsatlakoztathatjuk, ha elhelyezzük az alábbi kötetinformációkat az `/etc/fstab` állományba. Erre a célra `stripe` néven létrehozunk egy állandó csatlakozási pontot:

```
# mkdir /stripe
# echo "/dev/stripe/st0a /stripe ufs rw 2 2" \
    >> /etc/fstab
```

A `geom_stripe.ko` modult is automatikusan be kell tölteni a rendszerindítás során. Ehhez a következő sort kell hozzáadni a `/boot/loader.conf` állományhoz:

```
# echo 'geom_stripe_load="YES"' >> /boot/loader.conf
```

19.4. RAID1 - Tükrözés

A tükrözés számos vállalatnál és háztartásban alkalmazott technológia, amely az adatok megszakítás nélküli lementésére használatos. Amikor tükrözést használunk, az egyszerűen csak arra utal, hogy a B lemez ugyanazokat az adatokat tartalmazza, mint az A lemez. Vagy amikor a C és D lemez tartalma egyezik meg az A és B lemezekével. Függetlenül a lemezek kiosztásától, itt az a lényeg, hogy az egyik lemez teljes területe vagy az egyik partíciója le van másolva. Később az ezen a módon lementett adatok könnyen visszaállíthatóak anélkül, hogy ez a szolgáltatásban vagy az elérhetőségben bármilyen kimaradást okozna, és akár még fizikailag is biztonságosan tárolhatóak.

Először is szereznünk kell két egyforma méretű lemezt, valamint a példák feltételezik, hogy ezek a lemezek közvetlen elérésű (da(4)) SCSI-lemezek.

19.4.1. Az elsődleges lemezek tükrözése

Tegyük fel, hogy a FreeBSD az első, `da0` nevű lemezmeghajtón található, és a `gmirror(8)` számára ezt szeretnénk megadni az elsődleges adatok tárolásához.

A tükrözés létrehozásának megkezdése előtt a `kern.geom.debugflags` `sysctl(8)` változó megfelelő beállításával engedélyezzünk további nyomkövetési információkat és hozzáférést az eszközhöz:

```
# sysctl kern.geom.debugflags=17
```

Most építsük fel a tükrözést. Kezdjük az egészet a metaadatok elhelyezésével az elsődleges lemezmeghajtón, tehát tulajdonképpen az alábbi parancs segítségével hozzuk létre a `/dev/mirror/gm` eszközt:

FigyelemA rendszerindító meghajtóról készített tükrözés adatvesztést okozhat a lemez utolsó szektorában. Ennek kockázata csökkenthető, ha közvetlenül a FreeBSD friss telepítése után állítjuk be a tükrözést.

```
# gmirror label -vb round-robin gm0 /dev/da0
```

Erre a rendszernek a következő módon kell reagálnia:

```
Metadata value stored on /dev/da0.
Done.
```

A GEOM inicializálásához szükségünk lesz a `/boot/kernel/geom_mirror.ko` modul betöltésére:

```
# gmirror load
```

Megjegyzés: A parancs sikeres lefutása után a `/dev/mirror` könyvtárban létrehoz egy `gm0` eszközeíró.

A `geom_mirror.ko` modul betöltését így tudjuk engedélyezni a rendszer indításakor:

```
# echo 'geom_mirror_load="YES"' >> /boot/loader.conf
```

Nyissuk meg az `/etc/fstab` állományt, és cseréljük le benne az összes korábbi `da0` hivatkozást az újonnan kialakított `gm0` tükrözés eszközeírójával.

Megjegyzés: Ha `vi(1)` szövegszerkesztőt használjuk, akkor a következő módon tudjuk ezt egyszerűen megtenni:

```
# vi /etc/fstab
```

A `vi(1)` indítása után a `:w /etc/fstab.bak` kiadásával készítsünk az `fstab` állomány jelenlegi tartalmáról másolatot. Ezután a `:%s/da/mirror/gm/g` parancs használatával cseréljük ki az összes `da0` hivatkozást a `gm0` eszköz nevére.

Az így keletkező `fstab` állomány nagyjából következő módon fog kinézni. Most teljesen független, hogy SCSI vagy ATA meghajtókkal dolgozunk, a RAID eszköz neve mindig `gm` lesz:

# Eszköz	Csatlakozási pont	Típus	Beállítások	Dump	Menet
<code>/dev/mirror/gm0slb</code>	<code>none</code>	<code>swap</code>	<code>sw</code>	<code>0</code>	<code>0</code>
<code>/dev/mirror/gm0sla</code>	<code>/</code>	<code>ufs</code>	<code>rw</code>	<code>1</code>	<code>1</code>
<code>/dev/mirror/gm0sld</code>	<code>/usr</code>	<code>ufs</code>	<code>rw</code>	<code>0</code>	<code>0</code>
<code>/dev/mirror/gm0slf</code>	<code>/home</code>	<code>ufs</code>	<code>rw</code>	<code>2</code>	<code>2</code>
<code>#/dev/mirror/gm0s2d</code>	<code>/store</code>	<code>ufs</code>	<code>rw</code>	<code>2</code>	<code>2</code>
<code>/dev/mirror/gm0sle</code>	<code>/var</code>	<code>ufs</code>	<code>rw</code>	<code>2</code>	<code>2</code>
<code>/dev/acd0</code>	<code>/cdrom</code>	<code>cd9660</code>	<code>ro,noauto</code>	<code>0</code>	<code>0</code>

Indítsuk újra a rendszert:

```
# shutdown -r now
```

Ennek megfelelően a rendszer indítása közben a `da0` eszköz helyett a `gm0` eszközt fogjuk használni. Miután sikeresen befejeződött a rendszerindítás, a `mount` parancs kiadásával a saját szemünkkel is meggyőződhetünk az eredményről:

```
# mount
```

Filesystem	1K-blocks	Used	Avail	Capacity	Mounted on
<code>/dev/mirror/gm0sla</code>	<code>1012974</code>	<code>224604</code>	<code>707334</code>	<code>24%</code>	<code>/</code>
<code>devfs</code>	<code>1</code>	<code>1</code>	<code>0</code>	<code>100%</code>	<code>/dev</code>
<code>/dev/mirror/gm0slf</code>	<code>45970182</code>	<code>28596</code>	<code>42263972</code>	<code>0%</code>	<code>/home</code>
<code>/dev/mirror/gm0sld</code>	<code>6090094</code>	<code>1348356</code>	<code>4254532</code>	<code>24%</code>	<code>/usr</code>
<code>/dev/mirror/gm0sle</code>	<code>3045006</code>	<code>2241420</code>	<code>559986</code>	<code>80%</code>	<code>/var</code>
<code>devfs</code>	<code>1</code>	<code>1</code>	<code>0</code>	<code>100%</code>	<code>/var/named/dev</code>

A parancs kimenete az elvárásainknak megfelelően remekül néz ki. Zárásképpen a szinkronizálás megkezdéséhez a következő paranccsal illesszük be a `da1` eszközt a tükrözésbe:

```
# gmirror insert gm0 /dev/da1
```

A tükrözés állapota a létrejöttét követően az alábbi paranccsal ellenőrizhető:

```
# gmirror status
```

Az iménti parancs eredményének nagyjából a következőnek kell lennie miután a felépítettük a tükrözést és szinkronizáltuk az adatokat:

Name	Status	Components
mirror/gm0	COMPLETE	da0 da1

Hiba esetén a tükrözés továbbra is folytatódik, azonban ilyenkor a példában szereplő COMPLETE helyett a DEGRADED jelzést fogjuk látni.

19.4.2. Hibakeresés

19.4.2.1. A rendszer nem hajlandó elindulni

Ha a rendszerünk ehhez hasonló módon indul:

```
ffs_mountroot: can't find rootvp
Root mount failed: 6
mountroot>
```

Indítsuk újra a gépünket a kikapcsoló gomb vagy a reset segítségével. A rendszerindító menüben válasszuk a hatodik opciót (6). Ennek eredményeképpen megkapjuk a loader(8) parancssorát. Töltsük be a modult manuálisan:

```
OK? load geom_mirror
OK? boot
```

Ha ez beválik, akkor valamiért a modult nem sikerült rendesen betölteni. Ellenőrizzük, hogy a `/boot/loader.conf` állományban a neki szereplő megfelelő bejegyzés helyesen szerepel. Amennyiben a probléma továbbra is fennáll, helyezzük el a következő sort a rendszermag konfigurációs állományába, majd fordítsuk újra és telepítsük:

```
options GEOM_MIRROR
```

Ezzel várhatóan orvosoltuk a problémát.

19.4.3. A meghibásodott lemezek cseréje

A lemezek tükrözésének egyik legcsodálatosabb előnye, hogy a menet közben meghibásodott meghajtókat gond, és így feltehetően adatvesztés nélkül ki tudjuk cserélni.

Vegyük az iménti RAID-1 konfigurációt, és tételezzük fel, hogy a `da1` eszköz felmondta a szolgálatot és cserére szorul. A meghajtó leváltásához keressük meg a hibás eszközt, majd állítsuk le a rendszert. Tegyük be a helyére az újat és indítsuk újra a rendszerünket. Miután elindult az operációs rendszer, a következő parancsok kiadásával tudjuk logikailag is lecserélni a meghibásodott lemezt:

```
# gmirror forget gm0
# gmirror insert gm0 /dev/da1
```

Innen a `gmirror status` parancsával kísérhetjük figyelemmel a tükrözés újraszervezésének menetét. Csupán ennyi az egész.

19.5. Eszközök hálózati illesztése a GEOM-ban

A GEOM távoli eszközök, például lemezek, CD-meghajtók stb. használatát is támogatja a hálózati illesztést szolgáló segédprogramjaival, hasonlóan az NFS-hez.

Kezdeként létre kell hozni a megosztást elősegítő állományt. Ez az állomány határozza meg, ki és milyen szinten jogosult használni a megosztott erőforrásokat. Például ha megosztjuk az első SCSI-lemezen a negyedik slice-ot, az alábbi `/etc/gg.exports` állomány tökéletesen megfelel:

```
192.168.1.0/24 RW /dev/da0s4d
```

Ezzel a belső hálózaton levő összes számítógép képes lesz elérni a `da0s4d` partíción található állományrendszert.

Az eszköz megosztásához először gondoskodnunk kell róla, hogy ne legyen csatlakoztatva, majd ezután indítsuk el a `gated(8)` szerver démonját:

```
# gated
```

Ezt követően a `mount` felhasználásával csatoljuk az eszközt a kliensen, az alábbi parancs kiadásával:

```
# ggatec create -o rw 192.168.1.1 /dev/da0s4d
ggate0
# mount /dev/ggate0 /mnt
```

Innentől kezdve az eszköz elérhető lesz a `/mnt` csatlakozási ponton keresztül.

Megjegyzés: Fontos kiemelnünk, hogy ez a művelet eredménytelen, ha az adott eszközt vagy maga a szerver, vagy pedig valamelyik másik kliens már korábban csatolta.

Amikor az eszközre már nincs tovább szükségünk, biztonságosan le tudjuk választani az `umount(8)` paranccsal, hasonlóan bármelyik más lemezes eszközhöz.

19.6. A lemezes eszközök címkézése

A rendszer indítása közben a FreeBSD rendszermagja a talált eszközöknek megfelelően mindegyiknek létrehoz egy-egy eszközeíró. Ezzel a próbálgatásos módszerrel együtt jár néhány gond, például mi történik akkor, ha az új lemezes eszközt USB-n keresztül adjuk a rendszerhez? Nagyon valószínű, hogy ez az eszköz megkapja a `da0` nevet és ezzel az eredeti `da0` eszköz eltolódik a `da1` névhez. Ennek köszönhetően az `/etc/fstab` állományban felsorolt állományrendszerek csatolása veszélybe kerül, aminek következtében akár meghiúsulhat a rendszerindulás is.

Az egyik lehetséges megoldása a problémának, ha sorbafűzzük a SCSI eszközeinket, és így a SCSI-kártyához kapcsolt újabb eszköz egy addig nem használt számot fog birtokba venni. Mi helyzet azonban az USB-s eszközökkel,

amelyek kiüthetik az elsődleges SCSI-lemezeinket? Ez egyébként azért történhet meg, mert az USB-s eszközöket általában hamarabb keresi a rendszer, mint a SCSI kártyán levő eszközöket. Megoldhatjuk úgy ezt a gondot, hogy csak azután csatlakoztatjuk az említett eszközöket, miután a rendszer elindult. Megoldhatjuk viszont úgy is, hogy csak egyetlen ATA-meghajtót használunk és soha nem soroljuk fel a SCSI eszközöket az `/etc/fstab` állományban.

Ezeknél kínálkozik azonban egy jobb megoldás! A `glabel` nevű segédprogrammal a rendszergazda vagy a felhasználó úgy tudja címkézni a lemezmeghajtókat, hogy azok a `/etc/fstab` állományban szereplő címkéket használják. Mivel a `glabel` a címkét az adott szolgáltató utolsó szektorában tárolja el, ez a címke megmarad az újraindítás után is. Ha ezt a címkét eszközként használjuk, az állományrendszerek mindig ugyanarról a meghajtóról fognak csatlakozni, függetlenül attól, hogy milyen eszközeleíron keresztül érjük el ezeket.

Megjegyzés: Egyáltalán nem állítottuk, hogy egy címke csak állandó lehet. A `glabel` segítségével egyaránt létre lehet hozni állandó és átmeneti címkéket, de csak az állandó címke képes az újraindítás után is megmaradni. A két címketípus közti különbségeket a `glabel(8)` man oldal tárgyalja részletesebben.

19.6.1. Címketípusok és példák

A címkéknek két típusa létezik, az általános címke és az állományrendszer-címke. A címkék lehetnek állandóak vagy ideiglenesek. Az állandó címkék a `tunefs(8)` vagy `newfs(8)` parancsokkal hozhatóak létre. Ezek a címkék az adott állományrendszer típusa alapján elnevezett alkönyvtárakban jönnek létre a `/dev` könyvtárban belül. Például az UFS2 állományrendszer-címkék a `/dev/ufs` könyvtárban keletkeznek. Állandó címkék a `glabel label` paranccsal hozhatóak létre. Az ilyen címkék nem függenek az állományrendszerek típusától, a `/dev/label` könyvtárban jönnek létre.

Az ideiglenes címkék a következő induláskor elvesznek. Ezek a címkék a `/dev/label` könyvtárban keletkeznek, és ideálisak a kísérletezgetésre. Ideiglenes címkéket a `glabel create` paranccsal hozhatunk létre. Ezzel kapcsolatosan részletesebb felvilágosítást a `glabel(8)` man oldalon találhatunk.

Ha egy UFS2 állományrendszerre szeretnénk tenni egy állandó címkét az adataink megsemmisítése nélkül, adjuk ki a következő parancsot:

```
# tunefs -L home /dev/da3
```

Figyelem Ha az érintett állományrendszeren nincs üres hely, ennek a parancsnak a használata adatvesztéshez vezethet. Ilyen esetben inkább a felesleges állományok eltávolításával kellene törődnünk, nem pedig címkék hozzáadásával.

Ezután egy címkének kell megjelennie a `/dev/ufs` könyvtárban, amelyet vegyünk is fel az `/etc/fstab` állományba:

```
/dev/ufs/home          /home                  ufs      rw          2          2
```

Megjegyzés: Az állományrendszert tilos csatlakozni a `tunefs` futtatása alatt!

Most már a megszokott módon csatlakozhatjuk az állományrendszert:

```
# mount /home
```

Ettől a ponttól kezdve, amíg a `geom_label.ko` modul betöltődik a rendszerindítás során a `/boot/loader.conf` állományon keresztül, vagy a `GEOM_LABEL` opció megtalálható a rendszermag konfigurációs állományában, az eszközeíró a rendszerre nézve minden komolyabb következmény nélkül megváltozhat.

Állományrendszereket létrehozhatunk alapértelmezett címkével is a `newfs -L` paraméterével. Erről részletesebben a `newfs(8)` man oldalon olvashatunk.

Az alábbi paranccsal tudjuk törölni a címkét:

```
# glabel destroy home
```

A következő példában azt láthatjuk, hogyan címkézzük fel a rendszerindító lemezünk partícióit.

Példa 19-1. Partíciók címkézése a rendszerindító lemezen

A rendszerindításra használt lemezen levő partíciók felcímkézésével a rendszer képes lesz akkor is minden probléma nélkül elindulni, amikor áthelyezzük egy másik vezérlőre vagy átrakjuk egy másik számítógépbe. Például most tegyük fel, hogy van egy ATA csatlós lemezünk, amelyet a rendszer `ad0` néven ismert fel. Továbbá azt is feltételezzük, hogy a FreeBSD telepítése esetén megszokott partícionálási sémát választottuk, ahol `/`, `/var`, `/usr` és `/tmp` állományrendszereink, valamint egy lapozóterületünk van.

Indítsuk újra a rendszerünket és a `loader(8)` menüjében a **4** billentyű lenyomásával válasszuk az egyfelhasználós módot. Ezt követően adjuk ki a következő parancsokat:

```
# glabel label rootfs /dev/ad0s1a
GEOM_LABEL: Label for provider /dev/ad0s1a is label/rootfs
# glabel label var /dev/ad0s1d
GEOM_LABEL: Label for provider /dev/ad0s1d is label/var
# glabel label usr /dev/ad0s1f
GEOM_LABEL: Label for provider /dev/ad0s1f is label/usr
# glabel label tmp /dev/ad0s1e
GEOM_LABEL: Label for provider /dev/ad0s1e is label/tmp
# glabel label swap /dev/ad0s1b
GEOM_LABEL: Label for provider /dev/ad0s1b is label/swap
# exit
```

A rendszer indítása ezután többfelhasználós módban folytatódik. A rendszerindítás befejeződése után nyissuk meg az `/etc/fstab` állományt és írjuk át a hagyományos eszközneveket a hozzájuk tartozó címkékre. Az `/etc/fstab` végleges változata ennek megfelelően körülbelül így fog kinézni:

# Eszköz	Csatlakozási pont	Típus	Beállítások	Dump	Menet
<code>/dev/label/swap</code>	<code>none</code>	<code>swap</code>	<code>sw</code>	<code>0</code>	<code>0</code>
<code>/dev/label/rootfs</code>	<code>/</code>	<code>ufs</code>	<code>rw</code>	<code>1</code>	<code>1</code>
<code>/dev/label/tmp</code>	<code>/tmp</code>	<code>ufs</code>	<code>rw</code>	<code>2</code>	<code>2</code>
<code>/dev/label/usr</code>	<code>/usr</code>	<code>ufs</code>	<code>rw</code>	<code>2</code>	<code>2</code>
<code>/dev/label/var</code>	<code>/var</code>	<code>ufs</code>	<code>rw</code>	<code>2</code>	<code>2</code>

A rendszer most már újraindítható. Ha mindent jól csináltunk, akkor a rendszer indítása problémáktól mentesen fog zajlani és a `mount` parancs eredménye a következő lesz:

```
# mount
/dev/label/rootfs on / (ufs, local)
devfs on /dev (devfs, local)
/dev/label/tmp on /tmp (ufs, local, soft-updates)
```

```
/dev/label/usr on /usr (ufs, local, soft-updates)
/dev/label/var on /var (ufs, local, soft-updates)
```

A FreeBSD 7.2 kiadásától kezdődően a glabel(8) osztály az UFS esetén támogatja az ufsid, az állományrendszer egyedi rendszerszintű azonosítójából származtatott új címketípus használatát. Ezek a címkék a rendszer indítása során a /dev/ufsid könyvtárban jönnek automatikusan létre. Az ufsid címkéken keresztül tudunk az /etc/fstab állományban állományrendszereket csatlakoztatni. A jelenleg aktív állományrendszereket és azok ufsid azonosítóit a glabel status parancssal tudjuk lekérdezni:

```
% glabel status
                Name      Status  Components
ufsid/486b6fc38d330916    N/A    ad4s1d
ufsid/486b6fc16926168e    N/A    ad4s1f
```

Ebben a példában az ad4s1d képviseli a /var állományrendszert, míg a ad4s1f a /usr állományrendszert. Az adott ufsid értékek megadásával az /etc/fstab állományban a következőképpen tudjuk csatlakoztatni ezeket az állományrendszereket:

```
/dev/ufsid/486b6fc38d330916    /var      ufs      rw      2      2
/dev/ufsid/486b6fc16926168e    /usr      ufs      rw      2      2
```

Minden ufsid címkével rendelkező partíció csatlakoztatható ezen a módon. Ekkor nem kell manuálisan létrehoznunk a számunkra állandó címkéket, így automatikusan élvezhetjük az eszköznévétől független csatlakoztatás előnyeit.

19.7. Naplózó UFS GEOM-on keresztül

A FreeBSD 7.0-ás verziójának megjelenésével egy rég várt kiegészítés, a naplózás vált végre elérhetővé vált. Maga az implementáció a GEOM alrendszeren keresztül érhető el, és a gjournal(8) segédprogram segítségével könnyedén beállítható.

Mit is jelent a naplózás? A naplózás támogatásával a rendszer egy naplót vezet az állományrendszert érintő tranzakciókról — például az olyan változtatásokról, amelyek egy komplett írási műveletet eredményeznek — mielőtt még a metaadatok és lemezírási műveletek szabályosan befejeződnenek. Ez a könyvelés később visszajátszható az állományrendszerben lezajlott tranzakciók reprodukálásához, és ezzel megelőzhetőek az állományrendszerben keletkező esetleges ellentmondások.

Ez egy újabb módszer az adatvesztés és az állományrendszerben előforduló ellentmondások elkerülésére. Eltérően a Soft Updates módszertől, ahol a metaadatok frissítését biztosítják és követik nyomon, vagy a Snapshots módszertől, ahol pillanatképeket tárolunk az állományrendszerről, itt egy konkrét naplót tárolunk a lemez erre a célra fenntartott részén, amely bizonyos esetekben akár egy teljes külön merevlemez is lehet.

Ellentétben a többi naplózó állományrendszertől, a gjournal módszere blokk alapú és nem az állományrendszer részeként került implementálásra — csupán a GEOM egyik bővítménye.

A gjournal támogatásához a FreeBSD rendszermag konfigurációs állományában be kell állítani a következő opciót — amely a 7.0 és későbbi rendszereken alapbeállítás:

```
options UFS_GJOURNAL
```

Amennyiben naplózással rendelkező köteteket szeretnénk a rendszerindítás során csatlakoztatni, a `/boot/loader.conf` állományban következő sor hozzáadásával töltsük be a `geom_journal.ko` modult:

```
geom_journal_load="YES"
```

Szükség esetén ezt a funkciót akár a rendszermagba is beépíthetjük, ha felvesszük a következő sort a rendszermag konfigurációs állományába:

```
options GEOM_JOURNAL
```

Ha ezt aktiváltuk, egy szabad állományrendszeren az alábbi lépéseken keresztül tudunk létrehozni egy naplót, feltéve, hogy a `da4` egy új SCSI-meghajtó:

```
# gjournal load
# gjournal label /dev/ad4
```

Ennél a pontnál lennie kell egy `/dev/da4` és egy `/dev/da4.journal` eszközeírónak. Hozzunk létre egy állományrendszert ezen az eszközön:

```
# newfs -O 2 -J /dev/da4.journal
```

Ez a parancs létrehoz egy UFS2 állományrendszert a naplóval rendelkező eszközön.

Csatoljuk is be a `mount` segítségével az eszközt kívánt csatlakozási pontra:

```
# mount /dev/da4.journal /mnt
```

Megjegyzés: Ha több slice-unk is van, akkor a napló mindegyik slice-hoz külön létrejön. Például, ha az `ad4s1` és `ad4s2` egyaránt slice-ok, akkor a `gjournal` legyártja az `ad4s1.journal` és `ad4s2.journal` eszközeírókat.

A jobb teljesítmény elérése érdekében kívánatos lehet a naplót egy másik lemezen tartani. Ilyen esetekben a naplózás bekapcsolásához a naplót biztosító szolgáltatót vagy tárolóeszközt a naplózni kívánt eszköz után kell szerepeltetni. A naplózás akár az aktuálisan használt állományrendszeren is aktiválható a `tunefs` használatával. Az állományrendszer módosításakor viszont mindig érdemes biztonsági másolatot készíteni! Az esetek többségében a `gjournal` hibát fog jelezni, mivel nem tudja létrehozni a naplót, azonban ez nem védi meg az adatainkat a `tunefs` helytelen használata által okozott sérülésektől.

A rendszerindító lemezen is lehet naplózást használni. Ennek részleit a Naplózó UFS használata asztali számítógépeken (http://www.FreeBSD.org/doc/hu_HU.ISO8859-2/articles/gjournal-desktop) című cikkből ismerhetjük meg.

Fejezet 20. Támogatott állományrendszerek

Írta: Tom Rhodes.

20.1. Áttekintés

Az állományrendszerek szerves részét képezik napjaink operációs rendszereinek. Segítségükkel a felhasználók adatokat tölthetnek fel és tárolhatnak a számítógépen, szabályozhatják a hozzáférésüket, és természetesen működtethetik a merevlemezeiket. A különféle operációs rendszerekben általában azért annyi közös, hogy mindannyiukhoz tartozik egy natív, vagyis általuk alaptól ismert állományrendszer. A FreeBSD esetében ezt konkrétan a Fast File System vagy röviden FFS, amely az eredeti Unix™ File System, vagy más néven UFS megoldásain alapszik. A FreeBSD tehát a merevlemezekben ebben a natív állományrendszerben tárol adatokat.

A FreeBSD természetesen ezen kívül még ismer számos egyéb állományrendszert, ezáltal képes adatokat olvasni más operációs rendszerek részéről is kezelhető partíciókról, például helyi USB-eszközökről, flashkártyákról és merevlemezekről. Továbbá ismeri néhány más operációs rendszer natív állományrendszerét, mint például a Linux Extended File System (EXT) vagy éppen a Sun Z File System (ZFS).

FreeBSD alatt az egyes állományrendszerek ismerete változó. Bizonyos esetekben elegendő csupán egy megfelelő modul betöltése, máskor viszont egy komplett eszközkészlet segítségével tudunk velük dolgozni. Ez a fejezet igyekszik a Sun-féle Z állományrendszerrel kezdődően bemutatni a FreeBSD felhasználói számára más állományrendszerek használatát.

A fejezet elolvasása során megismerjük:

- a natív és támogatott állományrendszerek közti különbségeket;
- a FreeBSD által ismert állományrendszereket;
- hogyan engedélyezzünk, állítsunk be és érjünk el nem natív állományrendszereket.

A fejezet elolvasásához ajánlott:

- a UNIX és FreeBSD alapjainak ismerete (3 fejezet);
- a rendszermag konfigurációjának és fordításának alapvető fogásainak ismerete (8 fejezet);
- a különböző külső fejlesztésű szoftverek telepítésének ismerete (4 fejezet);
- a lemezek és egyéb tárolóeszközök, valamint a FreeBSD alatt az eszközök elnevezésének minimális ismerete (18 fejezet).

20.2. A Z állományrendszer (ZFS)

A Sun Z állományrendszere egy új, közös tárolási módszeren nyugvó technológia. Ez annyit jelent a gyakorlatban, hogy mindig csak annyi helyet foglal, amennyire az adatoknak közvetlenül szüksége van. Emellett úgy alakították ki, hogy az adatok épségét minél inkább védje, ezért például megtalálhatjuk benne a pillanatképek készítését, a másolatok létrehozását és az adatok sértetlenségének ellenőrzését. Továbbá egy RAID-Z néven bemutatott új replikációs modellt is támogat. A RAID-Z alapvetően a RAID-5 megoldáshoz hasonlít, azonban írás során keletkező hibák ellen igyekszik védelmet nyújtani.

20.2.1. A ZFS finomhangolása

A ZFS funkcióit megvalósító alrendszer alapértelmezés szerint meglehetősen sok erőforrást kíván, ezért nem árt a legjobb hatékonyságra behangolnunk a mindennapokban felmerülő igények mentén. Mivel ez még egy fejlesztés és tesztelés alatt álló része a FreeBSD-nek, elképzelhető, hogy ez a jövőben változik, viszont jelen pillanatban a következő lépéseket javasoljuk.

20.2.1.1. Memória

Hasznos, ha a rendszerünkben legalább 1 GB memória található, de inkább 2 vagy több az ajánlott. Az itt szereplő példákban ehelyett azonban mindenhol csupán 1 GB-ot feltételezünk.

Néhányaknak sikerült 1 GB-nál kevesebb központi memóriával is használni ezt az állományrendszert, azonban ilyenkor nagyon könnyen előfordulhat, hogy komolyabb terhelés esetén a FreeBSD a memória elfogyása miatt egyszerűen összeomlik.

20.2.1.2. A rendszermag beállításai

A rendszermag konfigurációs állományából javasolt eltávolítani az összes nem használt meghajtót és funkciót. A legtöbb meghajtó egyébként is elérhető modul formájában, és a `/boot/loader.conf` állományon keresztül minden gond nélkül betölthetőek.

Az i386 architektúránál szükségünk lesz az alábbi konfigurációs beállítás megadására, majd a rendszermag újrafordítására, végül a rendszer újraindítására:

```
options          KVA_PAGES=512
```

Ezzel az opcióval a rendszermag címtérét növeljük meg, aminek eredményeképpen a `vm.kvm_size` változót immáron az eredetileg 1 GB-os (PAE használata esetén pedig 2 GB-os) határ felé tudjuk állítani. Az itt megadandó értéket úgy tudjuk meghatározni, ha a beállítani kívánt méret MB-okban számolt értékét elosztjuk négygyel. A példában tehát az 512 egy 2 GB nagyságú címtérre ad meg.

20.2.1.3. A rendszertöltő beállításai

A `kmem` címtérét az összes FreeBSD által ismert architektúra esetében érdemes megnövelnünk. A teszteléshez használt rendszeren 1 GB fizikai memória állt rendelkezésre, itt a `/boot/loader.conf` állományban a következő értékek megadásával minden remekül működött:

```
vm.kmem_size="330M"
vm.kmem_size_max="330M"
vfs.zfs.arc_max="40M"
vfs.zfs.vdev.cache.size="5M"
```

A ZFS finomhangolásával kapcsolatos további javaslatokat a <http://wiki.freebsd.org/ZFSTuningGuide> címen olvashatunk.

20.2.2. A ZFS használata

A Z állományrendszerhez létezik egy olyan mechanizmus, amelyen keresztül már a FreeBSD indítása során el tudjuk végezni a közös tárolók csatlakoztatását:

```
# echo 'zfs_enable="YES"' >> /etc/rc.conf
# /etc/rc.d/zfs start
```

A leírás fennmaradó részében feltételezzük, hogy három SCSI-lemezünk van, amelyeket rendre a *da0*, *da1* és *da2* eszközök formájában tudunk elérni. Az IDE lemezek tulajdonosainak értelemszerűen itt majd az *ad* eszközneveket kell használniuk a SCSI-eszközök hivatkozásai helyett.

20.2.2.1. Egyetlen közös tároló használata

A *zpool* kiadásával egyetlen lemezen is létre tudunk hozni egy egyszerű, nem redundáns ZFS partíciót:

```
# zpool create minta /dev/da0
```

Az új közös tárterület a *df* parancs felhasználásával rögtön láthatóvá válik:

```
# df
Filesystem 1K-blocks    Used   Avail Capacity  Mounted on
/dev/ad0s1a 2026030 235230 1628718    13%    /
devfs          1         1         0   100%  /dev
/dev/ad0s1d 54098308 1032846 48737598     2%   /usr
minta       17547136         0 17547136     0%  /minta
```

A parancs kimenetében tisztán láthatjuk, hogy a *minta* nevű tároló nem csak egyszerűen elkészült, hanem egyúttal *csatolódott*. Innentől már a többi állományrendszerhez hasonlóan tetszőlegesen elérhető, az alábbi példához hasonlóan állományok hozhatóak rajta létre vagy listázható a tartalma:

```
# cd /minta
# ls
# touch proba
# ls -al
total 4
drwxr-xr-x  2 root  wheel   3 Aug 29 23:15 .
drwxr-xr-x 21 root  wheel 512 Aug 29 23:12 ..
-rw-r--r--  1 root  wheel   0 Aug 29 23:15 proba
```

Sajnos azonban ez a tároló még ki sem használja a ZFS által felkínált lehetőségeket. Ezért most hozzunk létre egy állományrendszert ezen a tárolón belül és engedélyezzük rajta a tömörítést:

```
# zfs create minta/tomoritett
# zfs set compression=gzip minta/tomoritett
```

A *minta/tomoritett* most már egy tömörített Z állományrendszer. Próbáljuk ki mit tud, és másoljunk néhány nagyobb méretű állományt a */minta/tomoritett* könyvtárba.

Ezután a tömörítés akár ki is kapcsolható:

```
# zfs set compression=off minta/tomoritett
```

Az állományrendszer leválasztásához adjuk ki a lenti parancsot, majd ellenőrizzük az eredményét a *df* használatával:

```
# zfs umount minta/tomoritett
# df
Filesystem    1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a   2026030    235232    1628716    13%      /
devfs         1          1         0    100%     /dev
/dev/ad0s1d   54098308  1032864  48737580    2%      /usr
minta        17547008         0  17547008    0%      /minta
```

Tegyük ismét elérhetővé és csatlakoztassuk újra az állományrendszert, majd nézzük meg az eredményt a `df` paranccsal:

```
# zfs mount minta/tomoritett
# df
Filesystem    1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a   2026030    235234    1628714    13%      /
devfs         1          1         0    100%     /dev
/dev/ad0s1d   54098308  1032864  48737580    2%      /usr
minta        17547008         0  17547008    0%      /minta
minta/tomoritett 17547008         0  17547008    0%      /minta/tomoritett
```

A közös terület és az állományrendszer mellesleg a `mount` parancs kimenetéből is megfigyelhető:

```
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
minta on /minta (zfs, local)
minta/tomoritett on /minta/tomoritett (zfs, local)
```

Látható, hogy a létrehozásuk után a Z állományrendszerek teljesen hétköznapi módon viselkednek, de természetesen további lehetőségek is elérhetőek hozzájuk. A következő példában adat néven készítünk egy új állományrendszert. Mivel ide majd nagyon fontos állományokat akarunk elhelyezni, állítsuk be, hogy minden adatblokkból két példány legyen:

```
# zfs create minta/adat
# zfs set copies=2 minta/adat
```

A `df` újbóli kiadásával most már látható is ez az állományrendszer és annak tárfoglalása:

```
# df
Filesystem    1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a   2026030    235234    1628714    13%      /
devfs         1          1         0    100%     /dev
/dev/ad0s1d   54098308  1032864  48737580    2%      /usr
minta        17547008         0  17547008    0%      /minta
minta/tomoritett 17547008         0  17547008    0%      /minta/tomoritett
minta/adat    17547008         0  17547008    0%      /minta/adat
```

Vegyük észre, hogy a közös területen levő állományrendszerek mindegyikén ugyanannyi szabad terület van. A `df` segítségével a későbbiekben remekül megfigyelhető lesz, hogy az egyes állományrendszerek mindig csak annyi területet foglalnak el a közös területből, amennyire abban a pillanatban ténylegesen szükségünk van. A Z állományrendszerek esetén megszűnik a partíciók és kötetek fogalma, és több állományrendszer tárolódik egyazon

közös területen. Ha már nem akarjuk használni, egyszerűen csak töröljük le az állományrendszereket és ezt a közös tárolót:

```
# zfs destroy minta/tomorított
# zfs destroy minta/adat
# zpool destroy minta
```

Nyilván tapasztalhattunk már, hogy a lemezeink olykor menthetetlenül meghibásodnak. Amikor egy lemezes meghajtó tönkremegy, a rajta tárolt adatok általában elvesznek. Az ilyen jellegű kellemetlenségek elkerülésének egyik módja az ún. RAID-tömbök építése. A következő szakaszban bemutatjuk, hogy a Z állományrendszerek esetén hogyan tudunk ilyen tömböket készíteni.

20.2.2.2. RAID-Z tömbök

Korábban már utaltunk rá, hogy ebben a szakaszban három SCSI-lemez, vagyis a da0, da1 és da2 eszközök használatát feltételezzük (vagy természetesen ad0 és így tovább, ha IDE-lemezeket használunk). Egy RAID-Z formátumú közös tároló készítéséhez a következő parancsot kell kiadni:

```
# zpool create tarolo raidz da0 da1 da2
```

Megjegyzés: A Sun ajánlása szerint egy RAID-Z konfigurációban legalább három, legfeljebb kilenc lemezt javasolt alkalmazni. Ha egyetlen közös tárolóban esetleg tíznél több lemezt szeretnénk felhasználni, akkor érdemes inkább kisebb RAID-Z csoportokra felosztani ezeket. Ha viszont csak két lemezünk van, de továbbra is redundanciára lenne szükségünk, hozzunk helyette létre egy ZFS tükrözést. Ezzel kapcsolatban részletesebben a zpool(8) man oldalon keresztül tájékozódhatunk.

Ennek hatására tehát keletkezik egy tarolo nevű Z-tároló. Ez a korábbiakhoz hasonló módon ellenőrizhető is a mount(8) és df(1) parancsokon keresztül. Természetesen az iménti listába további lemezeszközök tetszőlegesen felvehetőek. Most hozzunk létre ezen a közös területen egy felhasználók nevű állományrendszert, ahová majd a felhasználók adatait fogjuk tenni:

```
# zfs create tarolo/felhasznalok
```

Miután ezzel megvagyunk, az imént létrehozott állományrendszerre nyugodtan beállíthatunk tömörítést és biztonsági másolatokat. Ebben az alábbi parancsok lesznek a segítségünkre:

```
# zfs set copies=2 tarolo/felhasznalok
# zfs set compression=gzip tarolo/felhasznalok
```

Ezt követően költöztessük át a felhasználókat, vagyis másoljuk át az adataikat ide és hozzuk létre a megfelelő szimbolikus linkeket:

```
# cp -rp /home/* /tarolo/felhasznalok
# rm -rf /home /usr/home
# ln -s /tarolo/felhasznalok /home
# ln -s /tarolo/felhasznalok /usr/home
```

A felhasználók adatai immáron a frissen létrehozott /tarolo/felhasznalok állományrendszeren tárolódnak. Próbáljuk ki, hozzunk létre egy új felhasználót és jelentkezünk be vele.

Készítsünk most egy pillanatképet is, amelyet aztán később szükség esetén vissza tudunk állítani:

```
# zfs snapshot tarolo/felhasznalok@08-08-30
```

A snapshot csak valós állományrendszerekkel működik, könyvtárakra vagy állományokra nem. A nevében a @ karakter választja el egymástól a hozzá tartozó címkét az állományrendszer vagy kötet nevétől. Ha netalán a felhasználói könyvtárak valamiért megsérültek volna, a következő paranccsal állíthatóak vissza:

```
# zfs rollback tarolo/felhasznalok@08-08-30
```

Az adott időpontban aktív pillanatképeket az adott állományrendszer `.zfs/snapshot` könyvtárában találhatjuk meg. Például az előbb készített pillanatkép az alábbi paranccsal nézhető meg:

```
# ls /tarolo/felhasznalok/.zfs/snapshot
```

Ha ebből elindulunk, akkor pillanatok alatt írható egy olyan szkript, amely a felhasználók adatairól havonta készít egy pillanatképet. Ilyenkor azonban fontos számításba vennünk, hogy az idővel felgyülemelő pillanatképek rengeteg helyet el tudnak foglalni. A korábbi pillanatkép így távolítható el:

```
# zfs destroy tarolo/felhasznalok@08-08-30
```

Miután alaposan kipróbáltuk a `/tarolo/felhasznalok` néven létrehozott állományrendszerünket, állítsuk be véglegesen ez eddigi `/home` állományrendszer helyére:

```
# zfs set mountpoint=/home tarolo/felhasznalok
```

Ekkor a `df` és `mount` parancsok használatával meggyőződhetünk róla, hogy ezt az állományrendszert innentől már valóban a `/home` könyvtárnak tekintjük:

```
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
tarolo on /tarolo (zfs, local)
tarolo/felhasznalok on /home (zfs, local)

# df
Filesystem            1K-blocks    Used   Avail Capacity  Mounted on
/dev/ad0s1a            2026030   235240  1628708    13%      /
devfs                   1           1         0    100%    /dev
/dev/ad0s1d            54098308 1032826 48737618     2%      /usr
tarolo                 26320512     0 26320512     0%      /tarolo
tarolo/felhasznalok    26320512     0 26320512     0%      /home
```

Ezzel lényegében befejeztük a RAID-Z tömb konfigurációját. Az állományrendszerek állapotára vonatkozóan a `periodic(8)` alkalmazásával akár naponta kérhetünk ellenőrzést:

```
# echo 'daily_status_zfs_enable="YES"' >> /etc/periodic.conf
```

20.2.2.3. A RAID-Z helyreállítása

Minden szoftveres RAID implementáció kínál valamilyen megoldást az állapotának ellenőrzésére, ez alól tulajdonképpen a ZFS sem kivétel. A RAID-Z eszközök állapota a következő paranccsal kérdezhető le:

```
# zpool status -x
```

Ezt az üzenetet láthatjuk, amikor minden tároló kifogástalanul működik és semmilyen probléma sincs:

```
all pools are healthy
```

Ha viszont valamilyen gond lenne valamilyik lemezzel, például leállt, akkor az előbbi parancs eredménye ehhez lesz hasonló:

```
pool: tarolo
state: DEGRADED
status: One or more devices has been taken offline by the administrator.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Online the device using 'zpool online' or replace the device with
        'zpool replace'.
scrub: none requested
config:
```

NAME	STATE	READ	WRITE	CKSUM
tarolo	DEGRADED	0	0	0
raidz1	DEGRADED	0	0	0
da0	ONLINE	0	0	0
da1	OFFLINE	0	0	0
da2	ONLINE	0	0	0

```
errors: No known data errors
```

A válasz szerint az eszközt az adminisztrátor állította le. Ez ennél a példánál valóban igaz. Lemezeket a következő módon lehet leállítani:

```
# zpool offline tarolo da1
```

Így miután leállítottuk a rendszert, a da1 eszköz cserélhető. A rendszer soron következő indításakor ezzel a paranccsal tudjuk jelezni logikailag is a lemez cseréjét:

```
# zpool replace tarolo da1
```

Nézzük meg újra a tömb állapotát, de ezúttal a -x kapcsoló megadása nélkül, mivel csak így fogjuk látni:

```
# zpool status tarolo
pool: tarolo
state: ONLINE
scrub: resilver completed with 0 errors on Sat Aug 30 19:44:11 2008
config:
```

NAME	STATE	READ	WRITE	CKSUM
tarolo	ONLINE	0	0	0
raidz1	ONLINE	0	0	0
da0	ONLINE	0	0	0
da1	ONLINE	0	0	0
da2	ONLINE	0	0	0

```
errors: No known data errors
```

A példa szerint minden megfelelően működik.

20.2.2.4. Az adatok ellenőrzése

Előzetesen már szó esett róla, hogy a ZFS képes a tárolt adatok sértetlenségének ellenőrzésére. Az új állományrendszerek létrehozásánál ez a lehetőség automatikusan aktiválódik, de tetszés szerint letiltható:

```
# zfs set checksum=off tarolo/felhasznalok
```

Ez a lépés viszont nem feltétlenül jó döntés, mivel az adatintegritás megtartásához felhasznált ellenőrző összegek nagyon kevés helyet foglalnak és meglehetősen hasznosak. Emellett semmilyen észlelhető lassulást nem okoznak az állományrendszer használata során. Ha engedélyezzük, a ZFS ilyen ellenőrző összegek segítségével folyamatosan figyelni tudja az adatok épségét. Ezt az ellenőrzést a `scrub` paranccsal válthatjuk ki. Nézzük meg például a `tarolo` esetében:

```
# zpool scrub tarolo
```

Ez a vizsgálat a tárolt adatok mennyiségétől függően nagyon sokáig is eltarthat, illetve rengeteg lemezműveletet foglal magában, ezért egyszerre csak egy ilyen futtatása javasolt. Miután befejeződött, a tároló állapota az eredményének megfelelően frissül, amelyet közvetlenül utána le is kérdezhetünk:

```
# zpool status tarolo
pool: tarolo
state: ONLINE
scrub: scrub completed with 0 errors on Sat Aug 30 19:57:37 2008
config:
```

NAME	STATE	READ	WRITE	CKSUM
tarolo	ONLINE	0	0	0
raidz1	ONLINE	0	0	0
da0	ONLINE	0	0	0
da1	ONLINE	0	0	0
da2	ONLINE	0	0	0

```
errors: No known data errors
```

A példában látható az utolsó ellenőrzés ideje. Ezen lehetőség használatával hosszú időn keresztül szavatolni tudjuk az adataink épségét.

A Z állományrendszerrel kapcsolatos további beállítási lehetőségekről a `zfs(8)` és `zpool(8)` man oldalakon olvashatunk.

Fejezet 21. A Vinum kötetkezelő

Az eredeti változatot írta: Greg Lehey.

21.1. Áttekintés

Nem számít, milyen lemezeink is vannak, ugyanis mindig adódnak velük kapcsolatban gondjaink:

- Kicsik.
- Lassúk.
- Nem elég megbízhatóak.

Ezekre a problémákra javasoltak és meg is valósítottak számos megoldást. A felhasználók egy része általában úgy védekezik ellenük, hogy több, gyakran redundánsan tároló lemezt használ. A különféle kártyák és hardveres RAID-vezérlők támogatása mellett a FreeBSD alaprendszerében megtalálható egy blokkos eszközmeghajtóként a Vinum kötetkezelő is, amellyel virtuális lemezmeghajtókat lehet létrehozni. Tehát a *Vinum* egy olyan ún. *kötetkezelő*, vagyis virtuális lemezkezelő, ami az említett három problémára próbál megoldást adni. A Vinum a hagyományos lemezes tárolásnál jóval nagyobb rugalmasságot, teljesítményt és megbízhatóságot biztosít, valamint ismeri a RAID-0, RAID-1 és RAID-5 modelleket külön-külön és kombinálva is.

Ebben a fejezetben összefoglaljuk a hagyományos lemezes tárolás jellegzetes problémáit és bemutatjuk a Vinum kötetkezelőt.

Megjegyzés: A FreeBSD 5-ös verziójától kezdve a Vinumot újraírták a GEOM-nak megfelelően (19 fejezet), megtartva az eredeti elgondolásokat, elnevezéseket és a lemezen tárolt metaadatok formátumát. Ezt az újraírt változatot nevezik *gvinumnak* (*GEOM vinum*). A szövegben a *Vinum*ra kizárólag csak általánosságban hivatkozunk, függetlenül az implementációjától. Most már az összes parancsot a *gvinum* használatával kell kiadni, illetve a hozzá tartozó modul neve *vinum.ko*-ról *geom_vinum.ko*-ra változott és a megfelelő eszközeiről a */dev/vinum* könyvtár helyett a */dev/gvinum* könyvtárban találhatók. A FreeBSD 6. verziójától pedig a régi Vinum implementáció többé már nem is része az alaprendszernek.

21.2. Kicsik a lemezeink

A lemezek kapacitása ugyan növekszik, de velük együtt a tárigények is. Ezért gyakran érezzük úgy, hogy a rendelkezésünkre álló lemezek tárkapacitását meghaladó állományrendszerre lenne szükségünk. Kétségtelen, hogy ez a probléma messze nem akkora jelentőségű, mint például tíz évvel ezelőtt, de még mindig fennáll. Egyes rendszerek ezt úgy hidalták át, hogy létrehoztak egy olyan absztrakt eszközt, amely az adatokat több lemezen tárolja el.

21.3. A hozzáférési idők szűk keresztmetszetei

Napjaink rendszerei szinte állandóan egyszerre több adathoz is hozzá akarnak férni. Például egy nagy forgalmú FTP vagy HTTP szerver több 100 Mbit/s sebességű kapcsolattal is csatlakozhat a világhálózathoz, amelyeken keresztül párhuzamosan többezer nyi tranzakciót is folytathat, ami jelentősen meghaladja a legtöbb lemez átlagos átviteli sebességét.

A jelenleg kapható lemezek soros adatátviteli sebessége egészen 70 MB/s-ig is terjedhet, de ennek az értéknek kevés a jelentősége olyan környezetben, ahol több, egymástól függetlenül futó program próbál egyszerre hozzáférni, hiszen ilyen esetekben csak a töredékét képesek elérni. Ilyenkor sokkal érdekesebb a lemezt kezelő alrendszer szempontjából nézni a problémát: így az egyes adatátviteli kérések terhelése lesz a meghatározó paraméter, vagyis az az idő, amit a kérés teljesítésében érintett meghajtók eltöltenek a feldolgozással.

Bármelyik kérést is vesszük, a kiszolgáláshoz a meghajtónak először a megfelelő helyre kell mozgatnia az író/olvasó fejeket, meg kell várni a fej alatt elhaladó első szektort, majd végrehajtani a megfelelő műveletet. Ezek a műveletek szétválaszthatatlanok: semmi értelme nincs megszakítani ezeket.

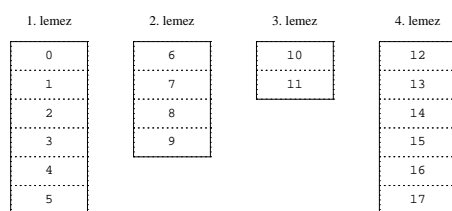
Tekintsünk egy átlagosnak mondható, nagyjából 10 kB méretű adatátvitelt: a legújabb nagyteljesítményű lemezek átlagosan 3,5 ms alatt képesek pozicionálni a fejeket. A leggyorsabb lemezek 15 000 fordulatot tesznek meg percenként (RPM), így az átlagos forgási késleltetés (egy fél fordulat ideje) 2 ms. 70 MB/s-os sebesség mellett az átvitel maga megközelítőleg 150 μ s, ami szinte elhanyagolható a pozicionálás idejéhez képest. Ilyen esetekben a tényleges adatátviteli sebesség 1 MB/s-nél alig valamivel többre esik vissza, és tisztán látszik, hogy erősen függ az átvitt adat mennyiségétől.

A hagyományos és kézenfekvő megoldása ennek a problémának “még több cséve” használata: egyetlen nagy lemez helyett alkalmazzunk több kisebb, de azonos tárhajótávolságú lemezt. Mindegyik lemez képes egymástól függetlenül mozgatni a fejeiket és az adatokat, aminek köszönhetően a tényleges adatátviteli mértéke nagyjából a lemezek számával arányosan növekszik.

Az adatátvitelben bekövetkező javulás pontos aránya természetesen kisebb, mint a lemezek száma: habár az egyes meghajtók képesek párhuzamosan mozgatni az adatokat, semmilyen módon garantálhatjuk, hogy a kérések egyenletesen oszlanak el köztük. Emiatt szinte elkerülhetetlen, hogy az egyik meghajtót nagyobb terhelés érje, mint a másikat.

A lemezekre eső terhelés egyenletessége erősen függ attól, hogyan osztjuk el az adatokat a meghajtók között. Az itt használt példában a lemezen tárolt adatokat egy könyv oldalaiként érdemes elképzelni, vagyis rengeteg szám szerint címezhető adatszektoroként. A virtuális lemezt ennek megfelelően a legegyszerűbben úgy tudjuk felosztani az egymás után következő független fizikai lemezek mérete szerint és így használni, mintha egy nagy könyvet kisebb részekre téptünk volna. Ezt a módszert nevezik *összefűzésnek*, és előnye, hogy a résztvevő lemezeknek nem kell azonos méretűeknek lenniük. Ez a megoldás remekül működik abban az esetben, amikor a virtuális lemez hozzáférései egyenletesen oszlanak el annak teljes területén. Amikor viszont az elérés csak egy kisebb területre korlátozódik, kevesebb javulás tapasztalható. A 21-1 Ábra mutatja be lemezek egy ilyen összefűzött konfigurációját.

Ábra 21-1. Az összefűzött szervezési mód



Feloszthatjuk a virtuális lemezüinket kisebb azonos méretű darabokra is, melyeket különböző eszközökön sorosan tárolunk el. Például az első 256 szektort eltároljuk az első lemezen, majd a következő 256 szektort a következő lemezen és így tovább. Az utolsó lemez kitöltése után az egész folyamat ismétlődik, egészen az összes lemez megtöltéséig. Ezt a leképezést *csíkozás*nak (“striping”) vagy RAID-0-nak nevezzük ¹. A csíkozás használata során valamivel bonyolultabbá válik az adatok megtalálása és többletmunkát is jelenthet olyan esetekben, amikor az adatátvitel több lemezt is érint, de ezzel egyidőben sokkal jobban szétosztja a terhelést a lemezek között. A 21-2 Ábra mutatja be a lemezek csíkozott szervezését.

Ábra 21-2. A csíkozott szervezési mód

1. lemez	2. lemez	3. lemez	4. lemez
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23

21.4. Adatintegritás

A modern lemezajtók utolsó fontos problémája, hogy nem eléggé megbízhatóak. Annak ellenére, hogy a lemezek ezen a téren meglehetősen sokat fejlődtek az utóbbi pár évben, egy szervernek még mindig ezek azok a központi részei, amelyek a leginkább hajlamosak a meghibásodásra. Amikor ez bekövetkezik, a hatása akár egy katasztrófával is felérhet: a sérült lemezmeghajtók cseréje és az adatok visszaállítása napokat is igénybe vehet.

Ennek a problémának a hagyományos megközelítése lenne a *tükrözés* (“mirroring”), vagyis amikor ugyanarról az adatról tartunk két példányt két eltérő fizikai hardveren. A RAID-szintek beköszöntével ezt a technikát RAID level 1-nak vagy RAID-1-nek is nevezik. Amikor írunk a kötetre, mindenhova írunk, az olvasás pedig bármelyik eszközzől elvégezhető. Így ha az egyik meghajtó tönkremenne, egy másikon még mindig megtalálható az összes adat.

A tükrözés két problémát vet fel:

- Ár. Legalább kétszer annyiba kerül, mint a nem redundánsan tároló megoldások.
- Teljesítménycsökkenés. Mivel az írást minden meghajtón végre kell hajtani, legalább kétszer annyi sávszélességet is felémeszt, mint a nem tükrözött kötetek esetén. Az olvasás viszont nem veszít a sebességéből: sőt, még gyorsabbnak is tűnhet.

Az adatintegritás megőrzésére egy másik megoldás a *paritás* használata, melyet a 2, 3, 4 és 5 RAID-szintek valósítanak meg. Ezek közül talán a RAID-5 a legérdekesebb. A Vinumban egy olyan csíkozott szervezési módként valósították meg, ahol minden csíkból egy blokk az összes többi paritási információját tartalmazza. A RAID-5 által megvalósított szervezés hasonlít a csíkozáshoz, azonban a RAID-5-ben mindegyik csík tartalmaz egy paritási információt is. Tehát a Vinumban, ahogy azt RAID-5 a megköveteli, a paritást tároló blokkok helye az egyik csíkról a másira változik. Az adatblokkokban található számok relatív blokkszámokat jelölnek.

Ábra 21-3. A RAID-5 szervezési mód

1. lemez	2. lemez	3. lemez	4. lemez
0	1	2	paritás
3	4	paritás	5
6	paritás	7	8
paritás	9	10	11
12	13	14	paritás
15	16	paritás	17

A RAID-5-nek a tükrözéshez képest megvan az az előnye, hogy jelentősen kevesebb tárhelyet igényel. Az olvasás hasonló a csíkozott szervezésekéhez, azonban az írás jóval lassabb, közel 25%-a az olvasás sebességének. Az egyik meghajtó meghibásodása esetén a tömb csökkentett módban még képes folytatni a működést: a fennmaradó meghajtókról továbbra is a megszokott módon lehet olvasni, viszont a sérült meghajtóról olvasott adatokat folyamatosan javítani kell a többiről származó segédinformációk szerint.

21.5. A Vinum objektumai

A tárgyalt problémák orvoslására a Vinumban egy négyszintű objektumhierarchiát alakítottak ki:

- A legjobban észlelhető objektum a virtuális lemez, amelyet *kötetnek* (volume) nevezünk. Ez a kötet lényegében ugyanazokkal a tulajdonságokkal rendelkezik, mint egy UNIX-os lemezmeghajtó, habár akadnak finomabb különbségek. Mérete korlátlan lehet.
- A kötetek *erekből* (plex) állnak, melyek a kötet teljes területét képviselik. Ennélfogva a hierarchia ezen szintje nyújtja a redundanciát. Az ereket legegyszerűbben a tükrözött tömbben helyet foglaló lemezekként tudjuk elképzelni, melyek ugyanazt az adatot tartalmazzák.
- Mivel a Vinum a UNIX lemezes tárolást megvalósító alrendszerében helyezkedik el, a többlemezes erek felépítéséhez használhatnánk a UNIX-os partíciókat, azonban ehhez a feladathoz nem eléggé rugalmasak, mivel a UNIX-os lemezek csak korlátozott számú partíciót tartalmazhatnak. A Vinum ehelyett *allemezeknek* (subdisk) nevezett folytonos területekre osztja fel az egyes UNIX-os partíciókat (a *meghajtókat*), melyeket aztán az erek létrehozására használ fel.
- A Vinum által létrehozott *meghajtókon* (drive) levő allemezek lesznek valódi UNIX-os partíciók. A Vinum-meghajtók tetszőleges számú allemezt tartalmazhatnak. Eltekintve a meghajtó elején található apró területtől, melyen a beállításokra és az állapotról vonatkozó információk tárolódnak, az egész meghajtó felhasználható adatok tárolására.

A most következő szakaszokban ismertetjük, hogy ezek az objektumok milyen módon szolgáltatják a Vinum részéről elvárt funkciókat.

21.5.1. A kötetek mérete

Az erek képesek a Vinum konfigurációjában található több különböző meghajtón elhelyezkedő allemezeket is nyalábba kötni. Ennek következményeképpen az egyes meghajtók mérete nem korlátozza az erek méretét, emiatt a

kötetét sem.

21.5.2. Redundáns adattárolás

A Vinum a tükrözést több ér egyetlen köteté olvasztásával hozza létre. Az erek mindegyike a kötetben található adatokat képviseli. Egy kötet legalább egy, legfeljebb nyolc eret tartalmazhat.

Habár egy ér egy kötet teljes adatát ábrázolja, előfordulhat olyan eset, hogy bizonyos részei hiányoznak fizikai, kialakítási (nem társítottunk allemezeket hozzájuk) okokból adódóan vagy véletlenül (a hozzá tartozó lemezterületek sérültek). Amíg legalább egy ér képes a kötet teljes tartalmát szolgáltatni, addig a kötet teljesen épnak tekinthető.

21.5.3. Teljesítmény

A Vinum az összefűzést és a csíkozást is egyaránt megvalósítja az erek szintjén:

- Az *összefűzött ér* allemezek területeiből építkezik.
- A *csíkozott ér* felosztja az adatokat az allemezek között. Az allemezek mindegyikének ugyanakkorának kell lennie, és legalább két allemeznek lennie kell, hogy eltérjen az összefűzött értől.

21.5.4. Hogyan szervezzük az ereket?

A FreeBSD 8.1 verziójában két fajta erezési megoldást találhatunk:

- Az összefűzött erek a legrugalmasabbak: tetszőleges számú allemezt tartalmazhatnak, az allemezek mérete pedig eltérhet. Az ér újabb allemezek hozzáadásával tovább bővíthető. Kevesebb processzoridőt igényel, mint egy csíkozott ér, habár a kettő többletköltsége közti eltérés nem mérhető. Másrésztől azonban nagyon érzékenyek a forgalmasabb pontokra, vagyis amikor az egyik lemez folyamatosan használatban van, miközben a többi üresen jár.
- A csíkozott (RAID-0) erek legnagyobb előnye, hogy csökkentik a forgalmasabb pontok kialakulását: a megfelelő méretű csíkszélesség (ami kb. 256 kB) választásával el tudjuk egyengetni a tömbben dolgozó meghajtók terhelését. Ennek a megközelítésnek a hátránya (részben) a sokkal összetettebb kód, valamint az allemezekre vonatkozó megszorítás, amely szerint meg kell egyezniük a méretüknek, illetve az érhez annyira bonyolult újabb allemezeket kapcsolni, hogy a Vinum jelenleg nem is képes rá. Ezeken kívül a Vinum még támaszt egy triviális igényt is: a csíkozott érben legalább két allemeznek lennie kell, mivel másképp nem tér el egy összefűzött értől.

A 21-1 Táblázat foglalja össze az egyes erezések előnyeit és hátrányait.

Táblázat 21-1. Vinum erezések

Erezés típusa	Legkevesebb allemez	Bővíthető	Megegyező méret	Alkalmazás
összefűzött	1	igen	nem	Sok adat tárolása, ahol a hangsúly a rugalmasságon és a mérsékelt teljesítményen van.

Erezés típusa	Legkevesebb allemez	Bővíthető	Megegyező méret	Alkalmazás
csíkozott	2	nem	igen	Nagy teljesítmény, nagy mennyiségű egyidejű hozzáférés mellett

21.6. Példák

A Vinum a rendszerben ismert objektumokkal kapcsolatos információkat egy *konfigurációs adatbázis*ban tartja fenn. Kezdetben a felhasználó egy vagy több konfigurációs állomány segítségével hozza létre ezt az adatbázist a `gvinum(8)` segédprogrammal. A Vinum ezt a konfigurációs adatbázist bemásolja mindegyik irányítása alatt álló slice-ba (melyek a Vinum *eszköz*nek hív). Az adatbázis minden egyes állapotváltás folyamán frissül, így egy újraindítás után minden egyes Vinum-objektum állapota pontosan helyreállítódik.

21.6.1. A konfigurációs állomány

A konfigurációs állomány írja le az egyes objektumokat. Egy egyszerűbb kötet definíciója így nézhet ki:

```
drive a device /dev/da3h
volume myvol
    plex org concat
    sd length 512m drive a
```

Ez az állomány négy Vinum-objektumot definiál:

- A *drive* kezdetű sor adja meg a lemez partícióját (*meghajtóját*) és a hardveren levő elhelyezkedését. Az *a* szimbolikus nevet kapta. A szimbolikus és a konkrét eszköznevek szétválasztásával lehetővé válik, hogy a lemezek félreértések nélkül átkerülhessenek egyik helyről a másikra.
- A *volume* kezdetű sor adja meg a kötetet. Itt az egyetlen szükséges jellemző a név, ami ebben az esetben a *myvol*.
- A *plex* kezdetű sor adja meg az eret. Itt az egyetlen szükséges paraméter a szervezési mód, ami ebben az esetben a *concat* (összefűzött). Nevet nem kell megadnunk, mivel a rendszer automatikusan létrehoz egy nevet a kötet nevéből a *.px* utótag hozzáadásával, ahol az *x* az ér száma lesz a kötetben belül. Emiatt a most definiált ér neve *myvol.p0* lesz.
- Az *sd* kezdetű sor adja meg az allemezt. Itt legalább meg kell adnunk a meghajtónak a nevét, ahol tárolni akarjuk, ill. a méretét. Ahogy már említettük az ereknél is, nevet nem kötelező megadnunk, mivel a rendszer magától rendel hozzá nevet, amit a hozzá tartozó ér nevéből származtat, hozzáadja a *.sx* utótagot, ahol az *x* az allemez éren belüli sorszáma lesz. Ennek következtében a Vinum ennek az allemeznek a *myvol.p0.s0* nevet adja.

Miután a `gvinum(8)` feldolgozta ezt az állományt, az alábbi kimenetet fogja adni:

```
# gvinum -> create config1
Configuration summary
Drives:      1 (4 configured)
Volumes:     1 (4 configured)
Plexes:      1 (8 configured)
Subdisks:    1 (16 configured)
```

```

D a                               State: up      Device /dev/da3h      Avail: 2061/2573 MB (80%)

V myvol                           State: up      Plexes:      1 Size:      512 MB

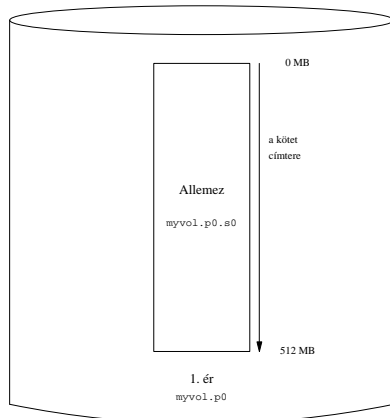
P myvol.p0                        C State: up      Subdisks:    1 Size:      512 MB

S myvol.p0.s0                     State: up      PO:          0 B Size:      512 MB

```

Ez a kimenet a gvinum(8) egyszerű listázási formátumát mutatja. Grafikusan a 21-4 Ábra mutatja be.

Ábra 21-4. Egyszerű Vinum-kötet



Ezen és az ezt követő ábrán egy kötetet láthatunk, amely ereket tartalmaz, amelyek pedig allemezeket. Ebben az alapvető példában a kötet egyetlen eret tartalmaz, amiben pedig egyetlen allemez van.

Az itt bemutatott kötetnek nincs semmilyen előnye a hagyományos lemezzartícionáláshoz képest. Egyetlen eret tartalmaz, tehát nem is redundáns. Az ér egyetlen allemezt tartalmaz, tehát nem tér el a megszokott lemezzartíciók helyfoglalásától sem. A következő szakaszokban sokkal érdekesebb konfigurációs módszereket is illusztrálunk.

21.6.2. Megnövelt rugalmasság: tükrözés

A kötetek rugalmassága tükrözéssel növelhető. Egy tükrözött kötet kiosztása során feltétlenül gondoskodnunk kell arról, hogy az egyes erkekhez tartozó allemezek eltérő meghajtókon találhatók, így az esetleges meghibásodások nem károsítják mind a két eret. Az alábbi konfigurációban egy kötetet tükrözünk:

```

drive b device /dev/da4h
volume mirror
plex org concat
sd length 512m drive a
plex org concat
sd length 512m drive b

```

Ebben a példában már nem kellett újra megadnunk az *a* meghajtót, mivel a Vinum figyelemmel kíséri az összes objektumot a saját konfigurációs adatbázisában. A definíció feldolgozása után a konfiguráció így fog kinézni:

```

Drives:      2 (4 configured)
Volumes:     2 (4 configured)
Plexes:      3 (8 configured)
Subdisks:    3 (16 configured)

D a          State: up      Device /dev/da3h      Avail: 1549/2573 MB (60%)
D b          State: up      Device /dev/da4h      Avail: 2061/2573 MB (80%)

V myvol      State: up      Plexes:      1 Size:      512 MB
V mirror     State: up      Plexes:      2 Size:      512 MB

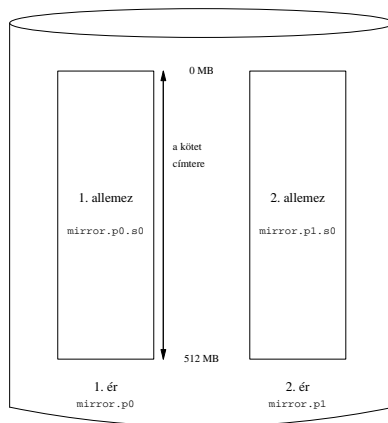
P myvol.p0   C State: up      Subdisks:    1 Size:      512 MB
P mirror.p0  C State: up      Subdisks:    1 Size:      512 MB
P mirror.pl  C State: initializing Subdisks:    1 Size:      512 MB

S myvol.p0.s0 State: up      PO:          0 B Size:      512 MB
S mirror.p0.s0 State: up      PO:          0 B Size:      512 MB
S mirror.pl.s0 State: empty PO:          0 B Size:      512 MB

```

A 21-5 Ábra ugyanezt a szerkezetet grafikusan is.

Ábra 21-5. Tükrözött Vinum-kötet



Ebben a példában minden ér tartalmazza a teljes 512 MB-os területet. Ahogy a korábbi példa esetén, itt is mindegyik ér csak egyetlen allemezt tartalmaz.

21.6.3. A teljesítmény javítása

Az előbbi példában szereplő tükrözött kötet egy tükrözetlen kötetnél már jobban ellenáll a hibáknak, azonban a teljesítménye is kisebb. A köteten minden egyes írás mind a két meghajtóra érvényesül, ezáltal a lemezek teljes sávszélességét nagyobb arányban használja. A teljesítményre vonatkozó megfontolásaink egy másik megközelítést

kívánnak meg: a tükrözés helyett inkább csíkozzuk szét az adatot a lehető legtöbb lemezen. Az alábbi konfiguráció egy olyan kötetet mutat be, ahol egy eret négy lemezmeghajtóan keresztül csíkozunk:

```
drive c device /dev/da5h
drive d device /dev/da6h
volume stripe
plex org striped 512k
  sd length 128m drive a
  sd length 128m drive b
  sd length 128m drive c
  sd length 128m drive d
```

Mint ahogy azt már korábban is említettük, nem szükséges még egyszer megadni azokat a meghajtókat, amiket a Vinum már ismer. A definíció feldolgozása után a konfigurációnk nagyjából így néz ki:

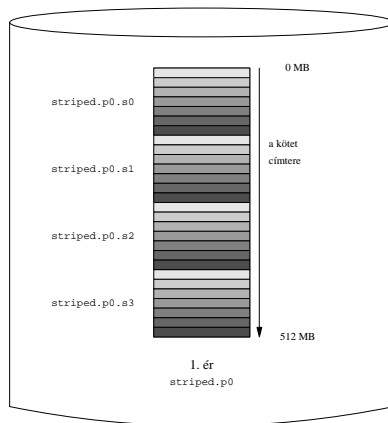
```
Drives:      4 (4 configured)
Volumes:     3 (4 configured)
Plexes:      4 (8 configured)
Subdisks:    7 (16 configured)

D a          State: up      Device /dev/da3h      Avail: 1421/2573 MB (55%)
D b          State: up      Device /dev/da4h      Avail: 1933/2573 MB (75%)
D c          State: up      Device /dev/da5h      Avail: 2445/2573 MB (95%)
D d          State: up      Device /dev/da6h      Avail: 2445/2573 MB (95%)

V myvol      State: up      Plexes:      1 Size:      512 MB
V mirror     State: up      Plexes:      2 Size:      512 MB
V striped    State: up      Plexes:      1 Size:      512 MB

P myvol.p0   C State: up      Subdisks:    1 Size:      512 MB
P mirror.p0  C State: up      Subdisks:    1 Size:      512 MB
P mirror.p1  C State: initializing Subdisks:    1 Size:      512 MB
P striped.p1 State: up      Subdisks:    1 Size:      512 MB

S myvol.p0.s0 State: up      PO:         0 B Size:      512 MB
S mirror.p0.s0 State: up      PO:         0 B Size:      512 MB
S mirror.p1.s0 State: empty   PO:         0 B Size:      512 MB
S striped.p0.s0 State: up      PO:         0 B Size:      128 MB
S striped.p0.s1 State: up      PO:        512 kB Size:      128 MB
S striped.p0.s2 State: up      PO:       1024 kB Size:      128 MB
S striped.p0.s3 State: up      PO:       1536 kB Size:      128 MB
```

Ábra 21-6. Csíkozott Vinum-kötet

Ez a kötet a 21-6 Ábrában látható. A csíkok sötétédése jelzi a helyüket az ér területében: a világosabbak elöl, a sötétebbek hátul szerepelnek.

21.6.4. Rugalmasság és teljesítmény

Megfelelő hardver birtokában lehet olyan köteteket is építeni, amelyek mind megnövelt rugalmasságot, mind pedig megnövelt teljesítményt mutatnak a szabványos UNIX-os partíciókhoz képest. Ennek a konfigurációs állománya így nézne ki:

```

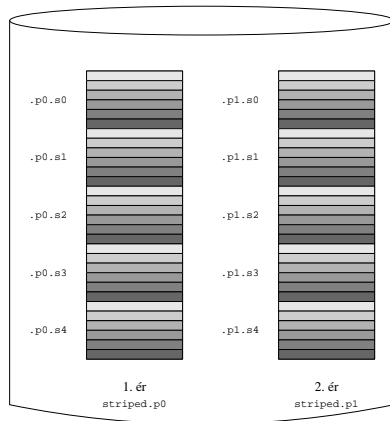
volume raid10
plex org striped 512k
  sd length 102480k drive a
  sd length 102480k drive b
  sd length 102480k drive c
  sd length 102480k drive d
  sd length 102480k drive e
plex org striped 512k
  sd length 102480k drive c
  sd length 102480k drive d
  sd length 102480k drive e
  sd length 102480k drive a
  sd length 102480k drive b

```

A második ér allemezei el vannak tolva az első ér allemezeitől két meghajtónyival. Ez segít megelőzni, hogy az írási műveletek ne ugyanarra az almezre vonatkozzanak, még akkor is, ha az adatátvitel két meghajtón is keresztülível.

A 21-7 Ábra illusztrálja ennek a kötetnek a szerkezetét.

Ábra 21-7. Tükrözött, csíkozott Vinum-kötet



21.7. Az objektumok elnevezése

Korábban már megismerhettük, hogy a Vinum alapértelmezett neveket társít az erekhez és az allemezekhez, habár ezek a nevek felülbírálnak. Ez viszont egyáltalán nem ajánlott, mivel már a VERITAS kötetkezelő, ahol tetszőleges neveket rendelhetünk az objektumokhoz, használata során kiderült, hogy akkora mértékű rugalmasságot nem kínál fel, mint amennyi zavart képes okozni.

A nevek tartalmazhatnak bármilyen nem üres karaktert, azonban érdemes inkább csak betűket, számjegyeket és az aláhúzást használni. A kötetek, erek és allemezek nevei akár 64 karakteresek is lehetnek, a meghajtók nevei pedig 32 karakteresek.

A Vinum objektumai a `/dev/gvinum` könyvtárban belüli hierarchiában helyezkednek el eszközeleíróként. Az imént említett példakonfiguráció hatására a következő eszközeleírók jönnek létre:

-

Megjegyzés: Ez a rész csak a Vinum korábbi, elavult implementációjára vonatkozik.

A `/dev/vinum/control` és `/dev/vinum/controld` nevű vezérlőeszközök, melyeket a `gvinum(8)` és a Vinum démon használ.

- Mindegyik kötethez egy eszközeleíró tartozik. Ezek a Vinum számára a központi eszközök, ezért az előbbi konfiguráció révén megjelennek a `/dev/gvinum/myvol`, `/dev/gvinum/mirror`, `/dev/gvinum/striped`, `/dev/gvinum/raid5` és `/dev/gvinum/raid10` eszközök.
-

Megjegyzés: Ez a rész csak a Vinum korábbi, elavult implementációjára vonatkozik.

Az egyes meghajtókhoz tartozó leírók a `/dev/vinum/drive` könyvtárban találhatóak. Ezek valójában szimbolikus linkek a megfelelő lemezes eszközökre.

- Minden kötethez közvetlen leírók tartoznak `/dev/gvinum` könyvtárban.
- Az egyes erek és allemezek eszközleírói a `/dev/gvinum/plex` és `/dev/gvinum/sd` könyvtárakban jelennek meg.

Például tekintsük most az alábbi konfigurációs állományt:

```
drive drive1 device /dev/sdlh
drive drive2 device /dev/sd2h
drive drive3 device /dev/sd3h
drive drive4 device /dev/sd4h
volume s64 setupstate
plex org striped 64k
  sd length 100m drive drive1
  sd length 100m drive drive2
  sd length 100m drive drive3
  sd length 100m drive drive4
```

Az állomány feldolgozása után az eszközleírókat a `gvinum(8)` az alábbi módon szervezi a `/dev/gvinum` könyvtárban:

```
drwxr-xr-x  2 root  wheel      512 Apr 13 16:46 plex
crwxr-xr--  1 root  wheel    91,   2 Apr 13 16:46 s64
drwxr-xr-x  2 root  wheel      512 Apr 13 16:46 sd

/dev/vinum/plex:
total 0
crwxr-xr--  1 root  wheel    25, 0x10000002 Apr 13 16:46 s64.p0

/dev/vinum/sd:
total 0
crwxr-xr--  1 root  wheel    91, 0x20000002 Apr 13 16:46 s64.p0.s0
crwxr-xr--  1 root  wheel    91, 0x20100002 Apr 13 16:46 s64.p0.s1
crwxr-xr--  1 root  wheel    91, 0x20200002 Apr 13 16:46 s64.p0.s2
crwxr-xr--  1 root  wheel    91, 0x20300002 Apr 13 16:46 s64.p0.s3
```

Jóllehet, az ereket és allemezeket nem ajánlott külön-külön elnevezni, a Vinum meghajtóknak nevet kell adni. Ezzel megoldhatóvá válik, hogy az egyes meghajtók automatikusan felismerhetőek legyenek abban az esetben is, amikor fizikailag áthelyezzük ezeket. A meghajtók nevei legfeljebb 32 karakteresek lehetnek.

21.7.1. Állományrendszerek létrehozása

A kötetek egyetlen kivétellel teljesen azonosak a lemezekkel a rendszer számára. Ugyanis a UNIX-os meghajtóktól eltérően a Vinum nem particionálja a köteteket, és ezért nem is tárolnak partíciós táblát. Ez megkövetelte néhány lemezkezelő segédprogram, leginkább a `newfs(8)` módosítását, mivel azok korábban megpróbálták a Vinum-kötetek

nevének utolsó betűit egy partíció azonosítójaként értelmezni. Például egy lemezes meghajtó neve `/dev/ad0a` vagy `/dev/da2h` alakú. Az előbbi az első (0) IDE lemez első (a) partícióját, míg az utóbbi a harmadik (2) SCSI lemez nyolcadik (h) partícióját jelöli. Ezzel szemben azonban a Vinum-kötetek neve `/dev/gvinum/concat` alakú lesz, ahol a név semmilyen kapcsolatban nem áll a partíció nevével.

Hétköznapi esetben a `newfs(8)` megpróbálja a lemez nevét értelmezni, és panaszkodik, ha nem sikerül. Például:

```
# newfs /dev/gvinum/concat
newfs: /dev/gvinum/concat: can't figure out file system partition
```

A köteten a `newfs(8)` parancs kiadásával tudunk állományrendszert létrehozni:

```
# newfs /dev/gvinum/concat
```

Megjegyzés: A FreeBSD 5.0 előtt verziókban a `newfs(8)` parancsnak a régi elnevezési séma használata mellett még át kell adni egy `-v` kapcsolót is:

```
# newfs -v /dev/vinum/concat
```

21.8. A Vinum beállítása

A `GENERIC` rendszermag nem tartalmazza a Vinumot. Habár készíteni lehet olyan rendszermagot, amelyik támogatja a Vinumot, mégsem ajánlott. A Vinumot a szabványos módon modulként (`kld`) indíthatjuk el. Még a `kldload(8)` használatára sincs szükség, mivel a `gvinum(8)` indulása során ellenőrzi a modul jelenlétét és betölti, ha még nem lenne jelen.

21.8.1. Indítás

A Vinum alapvetően ugyanúgy tárolja a konfigurációkat a `slice`-okban, mint maguk a konfigurációs állományok. A konfigurációs adatbázis beolvasása során a Vinum felismeri azokat a kulcsszavakat, amelyeknek nem szabad előfordulniuk az állományokban. Például a lemezek beállítása tartalmazhatja a következő szöveget:

```
volume myvol state up
volume bigraid state down
plex name myvol.p0 state up org concat vol myvol
plex name myvol.p1 state up org concat vol myvol
plex name myvol.p2 state init org striped 512b vol myvol
plex name bigraid.p0 state initializing org raid5 512b vol bigraid
sd name myvol.p0.s0 drive a plex myvol.p0 state up len 1048576b driveoffset 265b plexoffset 0b
sd name myvol.p0.s1 drive b plex myvol.p0 state up len 1048576b driveoffset 265b plexoffset 1048576b
sd name myvol.p1.s0 drive c plex myvol.p1 state up len 1048576b driveoffset 265b plexoffset 0b
sd name myvol.p1.s1 drive d plex myvol.p1 state up len 1048576b driveoffset 265b plexoffset 1048576b
sd name myvol.p2.s0 drive a plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 0b
sd name myvol.p2.s1 drive b plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 524288b
sd name myvol.p2.s2 drive c plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 1048576b
sd name myvol.p2.s3 drive d plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 1572864b
sd name bigraid.p0.s0 drive a plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 0b
sd name bigraid.p0.s1 drive b plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 4194304b
sd name bigraid.p0.s2 drive c plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 8388608b
sd name bigraid.p0.s3 drive d plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 12582912b
sd name bigraid.p0.s4 drive e plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 16777216b
```

Az előbbiektől nyilvánvalóan eltér abban, hogy itt már megjelennek a konkrét pozíciókra és elnevezésekre vonatkozó információk (melyeket a felhasználó is megadhat, azonban ezt nem tanácsoljuk), valamint az állapotok (ezeket nem

láthatja a felhasználó). A Vinum a konfigurációban nem tárolja a meghajtókat, helyette a beállított lemezes meghajtók partícióin fog Vinum-címkeket keresni. Ennek köszönhetően a Vinum még akkor is képes pontosan megtalálni a meghajtókat, amikor megváltoznak a hozzá tartozó UNIX-os meghajtók azonosítói.

21.8.1.1. Automatikus indítás

Megjegyzés: Ez a rész csak a Vinum elavult implementációjára vonatkozik. A `loader.conf(5)` közvetítésével a `Gvinum` mindig automatikusan elindul a hozzá tartozó modul betöltésével együtt. Ha a rendszerindításkor be akarjuk tölteni a `Gvinum` modult, akkor a `/boot/loader.conf` állományba vegyük fel a `geom_vinum_load="YES"` sort.

Az alábbi sort mindenképpen hozzá kell adnunk az `/etc/rc.conf` állományhoz, hogy a Vinum a rendszerindítás során automatikusan elinduljon:

```
start_vinum="YES"                # állítsuk YES-re az indításhoz
```

Hozzuk létre és írjuk bele, ha nem lenne `/etc/rc.conf` nevű állományunk. Ennek hatására a rendszer az indulás során betölti a Vinum `kld` modult, és a konfigurációban szereplő objektumokat elindítja. Ez még az állományrendszerek csatlakoztatása előtt történik meg, aminek révén a Vinum-köteteken található állományrendszereket a rendszer automatikusan át tudja vizsgálni az `fsck(8)` segítségével, majd csatlakoztatja ezeket.

Amikor a Vinumot a `vinum start` paranccsal indítjuk el, a Vinum beolvassa a konfigurációs adatbázist a Vinum-meghajtók egyikéről. Normál körülmények között mindegyik meghajtón megtalálható a konfigurációs adatbázis egy példánya, ezért szinte teljesen mindegy, melyik meghajtót is olvassa. Egy rendszer-összeomlás után azonban a Vinumnak meg kell tudnia állapítania, melyik meghajtón található meg az adatbázis legfrissebb példánya, és ezt kell beolvasnia. Ezután a lemaradt meghajtókon található adatbázispéldányokat szinkronizálja ehhez a változathoz.

21.9. Rendszerindítás Vinum-kötetről

Olyan számítógépeknél, ahol a teljesen tükrözött Vinum-alapú állományrendszereket használunk, kíváncsi lehet magát a rendszerindításhoz használt állományrendszert is tükrözni. Egy ilyen konfiguráció összeállítása már messze nem annyira egyszerű, mint egy tetszőleges állományrendszer esetén, mivel:

- Az indításhoz használt állományrendszernek már a folyamat nagyon korai szakaszában rendelkezésre kell állnia, ezért a Vinumnak már itt elérhetőnek kell lennie.
- A rendszerindító állományrendszert tartalmazó kötetben még ott kell lennie a rendszerindító kódnak és a rendszermagnak is, melyeket a rendszer saját eszközein (például ilyen a BIOS a PC-knél) keresztül kell tudnunk beolvasni, amiket viszont nem tudunk felkészíteni a Vinumra.

A soronkövetkező szakaszokban “rendszerindító kötetként” (root volume) fogunk általánosságban véve hivatkozni a rendszerindításhoz használt állományrendszert tartalmazó Vinum-kötetre. Ennek megfelelően valószínűleg jó ötlet a “root” névvel azonosítani ezt a kötetet, habár technikai szempontból ezt semmi nem követeli meg. Az itt felsorakozó példákban azonban ezt a nevet fogjuk használni.

21.9.1. A Vinum kellően korai indítása

Ennek kiváltásához számos lépést kell megtennünk:

- A rendszermagnak már el kell érnie a Vinumot a rendszerindítás során. Emiatt a 21.8.1.1 Szakaszban leírt automatikus indítási módszer nem alkalmazható erre a feladatra, és a `start_vinum` paramétert *nem* is szabad használni a most ismertetendő konfigurációban. A Vinumot statikusan bele is építhetjük a rendszermagba és így állandóan elérhető, de ez általában nem kielégítő megoldás. Megoldhatjuk úgy is, ha a `/boot/loader-re` (12.3.3 Szakasz) bízzuk a vinum modul betöltését, még a rendszermag előtt. Ezt az alábbi sorral válthatjuk ki a `/boot/loader.conf` állományban:

```
geom_vinum_load="YES"
```

•

Megjegyzés: A *Gvinum* használata során az összes többi beállítás automatikusan végrehajtható, amint a modul betöltődik, ezért ilyenkor csak a fentebb leírt eljárásra van szükség. Az itt felsoroltak csak az elavult Vinum implementációra vonatkoznak, csupán a régebbi típusú rendszerek kedvéért említjük meg.

A Vinumot nagyon korán életre kell keltenünk, hiszen a rendszerindításhoz használt állományrendszert tartalmazó kötetet kell élesítenünk. Alapértelmezés szerint a Vinum rendszerszinten futó része nem keres addig semmilyen Vinum-kötetinformációval rendelkező meghajtót, amíg a rendszergazda (vagy valamelyik rendszerindító szkript) ki nem adja a `vinum start` parancsot.

Megjegyzés: A most következő bekezdések mutatják be a szükséges lépéseket.

Ha hozzáadjuk a következő sort a `/boot/loader.conf` állományhoz, akkor azzal utasíthatjuk a Vinumot, hogy a rendszermag indítása során vizsgálja át az összes meghajtót:

```
vinum.autostart="YES"
```

Nem szükséges megmondani a rendszermagnak, merre keresse a rendszerindításhoz használt állományrendszert. A `/boot/loader` megkeresi a hozzá tartozó eszközt a `/etc/fstab` állományban és átadja ezt az információt a rendszermagnak. Amikor a csatlakoztatására kerül sor, a rendszermag az eszköz nevéből meg tudja állapítani, melyik eszközmeghajtót kérje meg a belső (fő- és al)eszközzazonosító leképzéséhez.

21.9.2. A Vinum-alapú rendszerindító kötet elérése a rendszertöltés során

Mivel a jelenlegi FreeBSD rendszertöltő csak 7,5 KB méretű és egyébként is csak az UFS állományrendszerről tud állományokat beolvasni (mint például a `/boot/loader`), teljesen lehetetlen még a Vinum belső szerkezetére is megtanítani, tehát a Vinum-konfigurációk értelmezésére és magának a rendszerindító kötet elemeinek kielemezésére. Ezért be kell vetnünk néhány trükköt ahhoz, hogy a rendszerindító kód számára a rendszerindításhoz használható szabványos "a" partíció képzetét keltsük.

Míndez csak akkor válik elérhetővé, ha az alábbi követelményeket teljesíti a rendszerindító kötet:

- Nem lehet csíkozott vagy RAID-5 típusú.

- Erenként nem tartalmazhat egynél több összefűzött allemezt.

Láthatjuk, hogy hasznos és lehetséges is több eret használni, melyek mindegyike a rendszerindító állományrendszer egy-egy másolatát tartalmazza. Az indulás folyamán azonban ezen példányok közül csak az egyiket fogja keresni a rendszer a rendszertöltőt és a többi állományt egészen addig, amíg a rendszermag magát az állományrendszert nem csatlakoztatja. A látszat kedvéért az ereken belül található allemekzek mindegyikének lennie kell egy saját "a" partíciójának, amivel lényegében alkalmassá válik a rendszerindításra. Ezeknek a hamis "a" partícióknak nem kell feltétlenül a többiekkel megegyező pozíciókon elhelyezkedniük, azonban a tévedések elkerülése érdekében valószínűleg hasznos olyan Vinum-köteteket létrehozni, ahol a keletkező tükrözött eszközök szimmetrikusak.

A rendszerindító kötet egyes eszközökön található "a" partícióit az alábbiak segítségével állíthatjuk be:

1. A rendszerindító kötet részeként megjelenő eszközön található allemek helyét (az eszköz elejétől számított eltolását) és méretét ellenőrizni kell az alábbi parancs segítségével:

```
# gvinum l -rv root
```

Ne felejtjük el, hogy a Vinum az eltolásokat és méreteket byte-okban méri. Ezekből tehát úgy nyerünk a `bsdlablel` használatához szükséges blokkszámokat, ha ezeket elosztjuk 512-vel.

2. Futassuk le a

```
# bsdlablel -e eszköznév
```

parancsot minden olyan eszközön, amelyik részt vesz a rendszerindító kötet kialakításában. Az *eszköznév* legyen a slice (fdisk)-táblát nem tartalmazó lemezek esetén a lemez neve (mint például `da0`), vagy ellenkező esetben a slice neve (például `ad0s1`).

Ha már lenne egy "a" partíció az eszközön (valószínűleg egy Vinum előtti rendszerindító állományrendszert tartalmaz), nevezzük át valami másra és így továbbra is elérhető marad (biztos, ami biztos), viszont többé már nem lesz a rendszer számára alapértelmezett rendszerindító eszköz. Az aktív partíciók (mint például az éppen csatlakoztatott rendszerindító állományrendszer) nem nevezhetők át, ezért ezt a lépést csak akkor tudjuk megtenni, ha a rendszerünket egy "Fixit" (Helyreállító) eszköztől indítjuk, vagy egy olyan kétlépéses folyamat során, ahol (tükrözés esetén) a lemeztől még nem indítottuk el a rendszert.

Ezt követően az eszközön található Vinum-partíciót (amennyiben létezik) az eszközön levő allemek eltolásához kell helyezni. Ennek eredménye lesz az új "a" partíció "offset" értéke. A partíció "size" (méret) értéke szó szerint átemelhető a fenti számításból. Az "fstype" legyen 4.2BSD. Az "fsize", "bsize" és "cpg" értékeket a jelenlegi állományrendszerhez mérten ajánlott megválasztani, azonban itt most egyáltalán nem bírnak jelentőséggel.

Ezzel a módszerrel létesítettünk egy olyan új "a" partíciót, amely lefedi az eszközön található Vinum-partíciót. Jegyezzük meg, hogy a `bsdlablel` kizárólag csak abban az esetben fogja megengedni ezt az átfedést, ha a Vinum-partíciónk "vinum" típussal van megjelölve.

3. Készen is vagyunk! Most már van minden eszközön egy hamisított "a" partíciónk, amelyeken megtalálható a rendszerindító kötet egy-egy másolata. Határozottan ajánlott még egyszer ellenőrizni a munkánkat az alábbi parancs kiadásával:

```
# fsck -n /dev/eszköznév
```

Figyeljünk arra, hogy az összes vezérlési információt tartalmazó állománynak a Vinum-köteten található rendszerindító állományrendszerre kell vonatkoznia, ami viszont egy új Vinum rendszerindító kötet beállítása után nem feltétlenül egyezik meg a jelenlegi aktív állományrendszerrel. Különösen az `/etc/fstab` és `/boot/loader.conf` állományokat kell ilyen szempontból ellenőriznünk.

A következő indítás során a rendszertöltő már az új Vinum-alapú rendszerindító állományrendszerrel fogja összeszedni a működéséhez szükséges adatokat és ezeknek megfelelően cselekedni. Végül, a rendszermag inicializálója után, mikor az összes eszközt felismerte, egy ehhez hasonló feltűnő üzenet fogja jelezni a beállítás sikerességét:

```
Mounting root from ufs:/dev/gvinum/root
```

21.9.3. Egy Vinum-alapú rendszerindító állományrendszer példája

Miután sikeresen konfiguráltuk a rendszerindító Vinum-kötetet, a `gvinum l -rv root` kimenete nagyjából így fog kinézni:

```
...
Subdisk root.p0.s0:
    Size:          125829120 bytes (120 MB)
    State: up
    Plex root.p0 at offset 0 (0 B)
    Drive disk0 (/dev/da0h) at offset 135680 (132 kB)

Subdisk root.p1.s0:
    Size:          125829120 bytes (120 MB)
    State: up
    Plex root.p1 at offset 0 (0 B)
    Drive disk1 (/dev/dal0h) at offset 135680 (132 kB)
```

Itt (a `/dev/da0h` partícióhoz képesti) 135680-as eltolás értékekre kell figyelniük. Ez képződik le a `bsdlabeled` fogalmi rendszerében aztán 265 darab 512 byte-os blokkra a lemezen. Ehhez hasonlóan a rendszerindító kötet mérete 245 760 darab 512 byte-os blokk lesz. A rendszerindító kötet másodpéldányát tartalmazó `/dev/dal0h` ugyanilyen beállításokkal rendelkezik.

Az említett eszközök valahogy így jelennek meg a `bsdlabeled` szerint:

```
...
8 partitions:
#      size  offset  fstype  [fsize bsize bps/cpg]
a:   245760    281  4.2BSD   2048 16384    0  # (Cyl.  0*- 15*)
c:  71771688     0  unused     0     0    0  # (Cyl.  0 - 4467*)
h:  71771672    16   vinum           # (Cyl.  0*- 4467*)
```

Megfigyelhető, hogy a hamis "a" partíció "size" paraméter értéke megegyezik a fentebb becsült értékkel, miközben az "offset" paraméter értéke egyenlő lesz a "h" Vinum-partíción belüli eltolás és az eszközön (vagy slice-on) belüli eltolás összegével. Ez jellemzően egy olyan beállítás, amivel szükségszerűen el tudjuk kerülni a 21.9.4.3 Szakaszban leírt hibajelenséget. Látható továbbá az is, hogy az egész "a" partíció végig az eszköz összes Vinum adatát tartalmazó "h" partíciójában foglal helyet.

A példával kapcsolatban megjegyezzük, hogy itt az egész eszközt a Vinum felügyelete alá bocsátottuk, tehát nem marad hátra semmilyen Vinum előtt használt rendszerindító partíció, hiszen ez egy olyan lemez, amelyet eleve egy Vinum-konfigurációba szántunk.

21.9.4. Hibakeresés

Fontos tudunk, hogy probléma esetén hogyan tudjuk helyreállítani a rendszerünket. A következő felsorolásban bemutatunk néhány ismert buktatót és a megoldásaikat.

21.9.4.1. A rendszertöltő elindul, de a rendszer viszont már nem

Ha valamilyen okból a rendszer nem indulna el, a 10 másodpercig tartó visszaszámlálás során a rendszertöltőt még meg tudjuk állítani a **szóköz** lenyomásával. Ekkor a betöltő által használt változók (mint például a `vinum.autostart`) a `show` segítségével megvizsgálhatóak és a `set` vagy `unset` parancsokkal módosíthatóak.

Ha mindössze az volt a probléma, hogy a Vinum modulja nem szerepelt az automatikusan betöltendő modulok között, a `load geom_vinum` parancs kiadásával betölthetjük azt.

Miután végeztünk, a rendszerindítás folyamata a `boot -as` paranccsal folytatható. A `-as` kapcsolók jelzik a rendszermag számára, hogy kérdezzen rá a rendszerindító állományrendszerre a csatlakoztatása előtt (`-a`) és csak egyfelhasználós módban indítsa a rendszert (`-s`), ahol a rendszerindító állományrendszer írásvédett. Így, ha csak egyetlen eret csatlakoztattunk egy többeres kötetből, az erek még véletlenül sem tudnak egymásnak ellentmondó állapotba kerülni.

Amikor megjelenik a csatlakoztatandó rendszerindító állományrendszert bekérése, bármelyik érvényes rendszerindításra alkalmas állományrendszer megadható. Amennyiben az `/etc/fstab` állományt jól beállítottuk, az alapértelmezett érték egy `ufs:/dev/gvinum/root` értékhez hasonló alakú lesz. Itt általában egy `ufs:da0d` formátumú értéket láthatunk, amely feltehetően egy Vinum használata előtti rendszerindító állományrendszert tartalmazó partíció. Legyünk óvatosak, ha itt egy olyan "a" partíciót adunk meg, amely valójában egy rendszerindító Vinum-eszköz allemezeire hivatkozik, mivel egy tükrözött konfiguráció esetén csak az eszköz egyik részét fogjuk csatlakoztatni. Ha a későbbiekben ezt az állományrendszert már nem csak írásvédett módban csatlakoztatjuk, mindenképpen el kell távolítanunk a rendszerindító Vinum-kötetből a többi eret, mivel máskülönben nagy valószínűséggel eltérő adatokat fognak tartalmazni.

21.9.4.2. Csak az elsődleges rendszertöltő indul el

Amikor az elsődleges rendszertöltő még elindul, viszont a `/boot/loader` már nem tud betöltődni (ezt rendszerindítás megkezdése után bal oldalt rögtön megjelenő forgó vonalból vehetjük észre), a **szóköz** lenyomásával itt még tehetünk egy kísérletet a betöltés megszakítására. Ennek hatására a rendszertöltés megáll a második fázisban, lásd 12.3.2 Szakasz. Itt a rendszerindításhoz megpróbálhatunk megadni egy másik partíciót, például egy olyat, amely a korábbi rendszerindító állományrendszert tartalmazza és amelyet az előbb átneveztünk az "a"-ról.

21.9.4.3. Semmi sem indul, a rendszertöltő hibákat ír

Ez a helyzet akkor állhat elő, ha a Vinum telepítése során tönkretettük volna a rendszertöltőt. Sajnos a Vinum minden esetben 4 KB helyet hagy szabadon a partíció elején, a saját fejléc információjának rögzítése előtt. Az ide kerülő első és második fázisú rendszertöltők, illetve a `bsdlabel` adatai azonban jelenleg 8 KB helyet kívánnak meg. Így ha a Vinum-partíció egy rendszerindításra szánt slice vagy lemez 0. eltolásánál kezdődik, a Vinum beállításai felül fogják írni a rendszertöltőt.

A rendszertöltő is ugyanígy felülírja a Vinum fejlécét és akkor a Vinum nem találja a lemezeit, ha a fenti problémát orvosolva, például egy "Fixit" (Helyreállító) lemez segítségével, újratelepítjük a rendszertöltőt a 12.3.2 Szakaszban bemutatott `bsdlabel -B` parancs segítségével. Noha a Vinum egyetlen konkrét konfigurációs beállítása vagy a kötetekben tárolt adat sem sérül meg és vissza tudjuk állítani az összes elveszett információt ugyanakkor a

Vinum-konfigurációnak az újbóli megadásával, a helyzetet magát nehéz megoldani. A Vinum-fejléc és a rendszertöltő ütközésének megszüntetéséhez ugyanis legalább 4 KB-tal arrébb kell mozgatnunk az egész Vinum-partíciót.

Megjegyzések

1. A RAID jelentése: Olcsó lemezek hibatűrő tömbje (Redundant Array of Inexpensive Disks). Különféle típusú hibatűrési megoldásokat vonultat fel, habár az eredeti elnevezés félrevezető lehet, mivel redundanciát nem tartalmaz.

Fejezet 22. Virtualizáció

Írta: Murray Stokely.

22.1. Áttekintés

A virtualizációs szoftverek lehetővé teszik, hogy ugyanazon a számítógépen egyszerre több operációs rendszert is futassunk. Ezeknek a programcsomagoknak gyakorta részük egy gazda operációs rendszer is, amely a virtualizációs szoftvert futattja és ismer bizonyos vendég operációs rendszereket.

A fejezet elolvasása során megismerjük:

- a gazda- és a vendég operációs rendszerek közti különbségeket;
- hogyan telepítsünk FreeBSD-t egy Intel-alapú Apple Macintosh számítógépre;
- hogyan telepítsünk a **Virtual PC** használatával FreeBSD-t Microsoft Windows-ra;
- hogyan hozzuk ki a legtöbbet FreeBSD rendszerünkől virtualizáció alatt.

A fejezet elolvasásához ajánlott:

- alapvető UNIX-os és FreeBSD-s ismeretek (3 fejezet);
- a FreeBSD telepítésének ismerete (2 fejezet);
- a hálózati kapcsolatok beállításának ismerete (31 fejezet);
- külsős alkalmazások telepítésének ismerete (4 fejezet).

22.2. A FreeBSD mint vendég

22.2.1. Parallels-szel Mac OS®-en

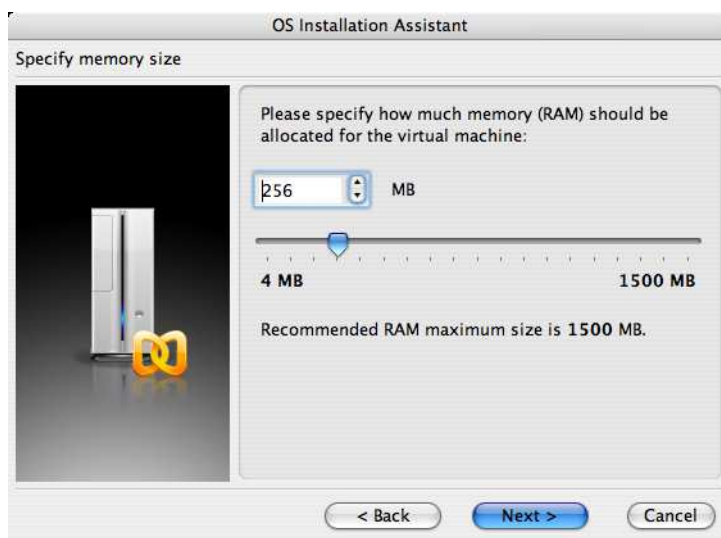
A **Parallels Desktop** a Mac OS 10.4.6, vagy afeletti verzióját futató, Intel-alapú Apple Mac személyi számítógépekre fejlesztett kereskedelmi alkalmazás. A FreeBSD-t teljes mértékben támogatja vendégként. Miután telepítettük a **Parallels**-t a Mac OS X-re, be kell állítanunk egy virtuális gépet, majd erre felraknunk a kívánt vendég operációs rendszert.

22.2.1.1. A FreeBSD telepítése Mac OS X/Parallelsre

A FreeBSD Mac OS X/**Parallels** párosra telepítéséhez első lépésként készítenünk kell egy új virtuális számítógépet. A létrehozás során válasszuk a **Guest OS Type**-nak (a vendég operációs rendszer típusának) a **FreeBSD**-t:

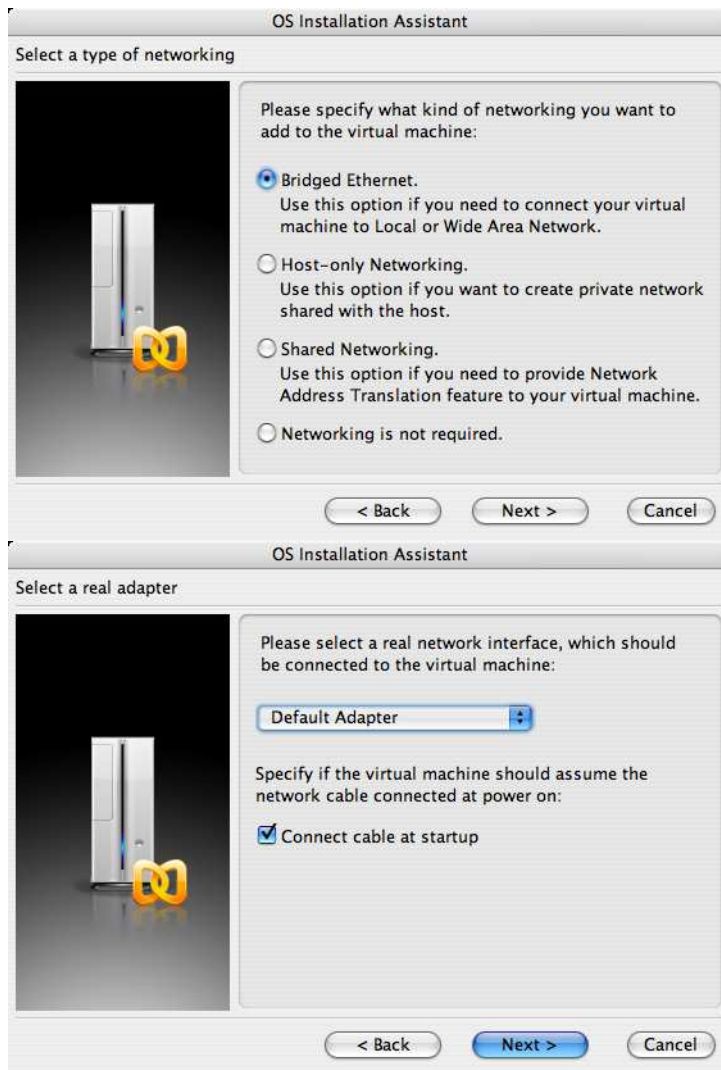


Ezután adjunk meg egy nagyjából elfogadható méretet a virtuális merevlemezünknek, valamint annyi memóriát, amennyire szükségünk lehet a virtuális FreeBSD-nk használata során. Egy 4 GB-os lemez és 512 MB rendszermemória a legtöbb esetben jó választásnak bizonyulhat a FreeBSD **Parallels** alatti használata során:

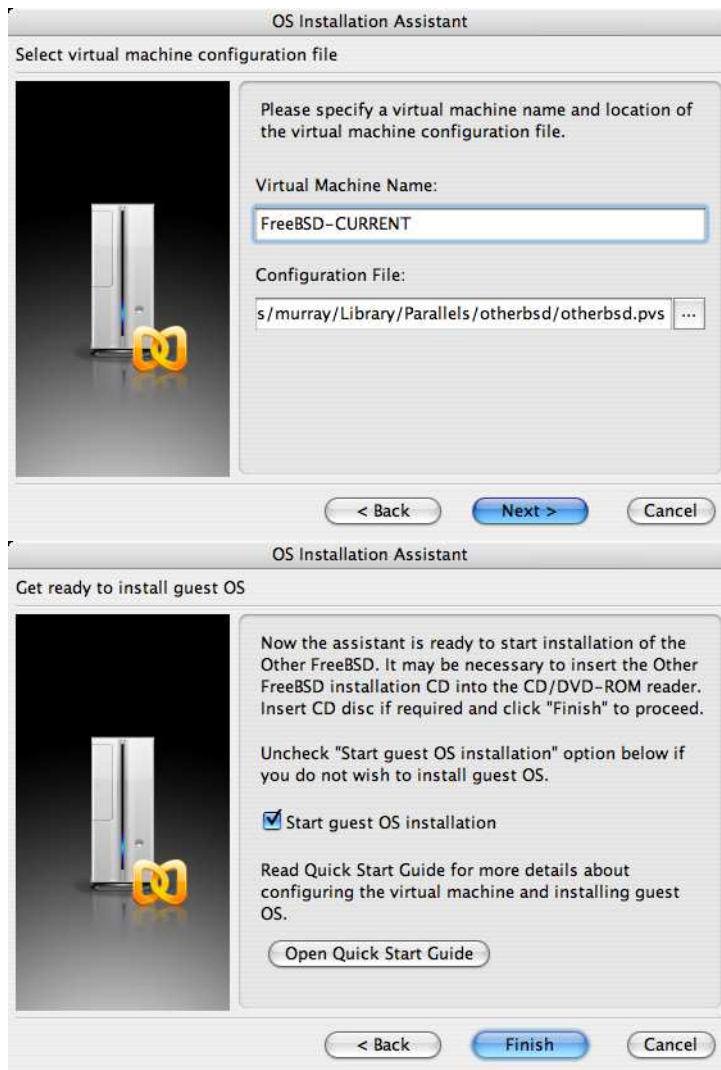




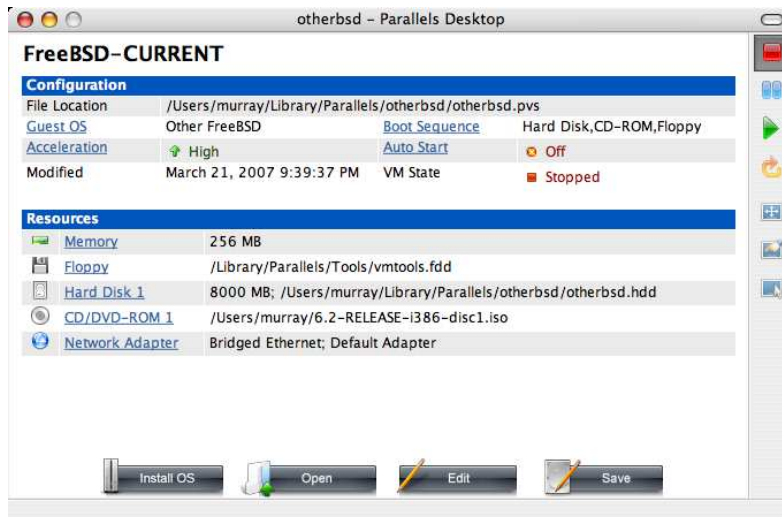
Válasszuk ki a hálózatkészlet típusát és a hálózati csatlót.



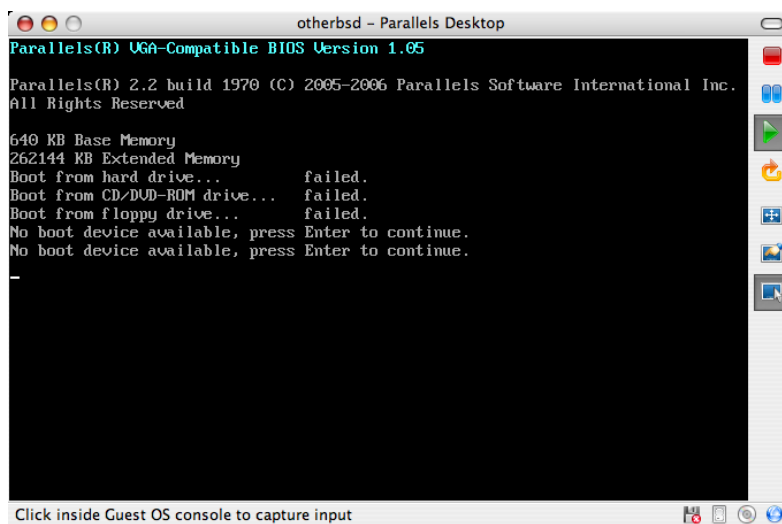
Mentsük el és fejezzük be a konfigurálást.



Miután a FreeBSD-s virtuális gépünk elkészült, telepítenünk kell rá magát az operációs rendszert is. Ezt a legegyszerűbben hivatalosan FreeBSD telepítő CD-ről, vagy a hivatalos FTP oldalról letölthető CD-képpel tehetjük meg. Ha lemásoltuk a megfelelő CD-képet a Mac helyi állományrendszerére, vagy behelyeztük a telepítő CD-t a CD-meghajtóba, kattintsunk a FreeBSD-s **Parallels** ablakunk jobb alsó sarkában található lemez ikonjára. Ekkor feljön egy párbeszédablak, ahol összerendelhetjük a virtuális gépünk CD-meghajtóját egy lemezen található képpel, vagy éppen a valódi CD-meghajtónkkal.



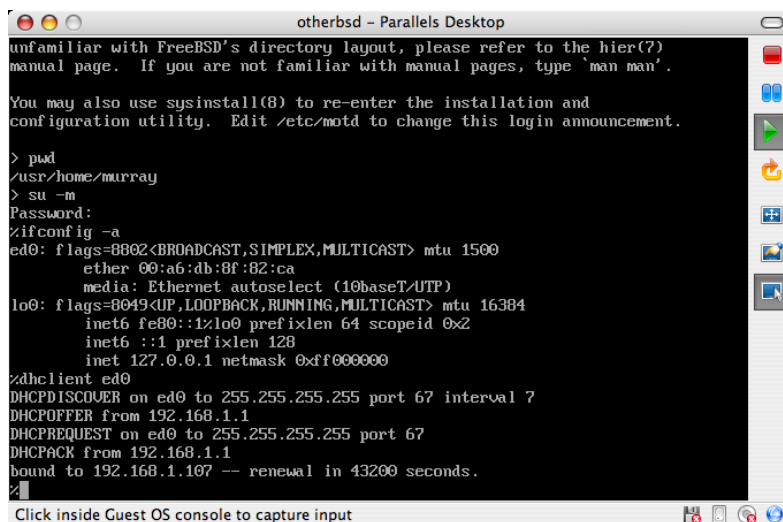
Ahogy megtettük az imént említett összerendelést, indítsuk is újra a FreeBSD-s virtuális gépünket a megszokott módon, az újraindítás ikonjára kattintva.



Ekkor a rendszer megtalálja a FreeBSD telepítőlemez és a **sysinstall** segítségével megkezd a telepítést a 2 fejezetben leírtak szerint. Ha szükségünk van rá, telepíthetjük az X11-et is, de egyelőre még ne próbáljuk beállítani.



A telepítés befejezését követően indítsuk újra a frissen telepített FreeBSD-s virtuális gépünket.



22.2.1.2. A FreeBSD beállítása Mac OS X/Parallels-en

Miután telepítettük a FreeBSD-t Mac OS X/Parallels-re, még vár ránk néhány konfigurációs lépés a rendszer virtualizált működésének optimalizálása érdekében.

1. A rendszerbetöltő változóinak beállítása

A legfontosabb lépés a `kern.hz` változó értékének csökkentése, amivel így a FreeBSD processzor-kihasználtságát is csökkentjük a **Parallels** alatt. Ezt a következő sor hozzáadásával tehetjük meg a `/boot/loader.conf` állományban:

```
kern.hz=100
```

Enélkül egy üresjáratban levő FreeBSD **Parallels**-vendég az iMac® egy processzorának durván 15%-át foglalja le. A változtatás életbe léptetése után azonban ez megközelítően 5%-ra redukálható.

2. Egy új konfigurációs állomány létrehozása a rendszermaghoz

Nyugodtan eltávolíthatjuk az összes SCSI, FireWire és USB eszközmeghajtót. A **Parallels** által felkínált virtuális hálózati csatolót az ed(4) meghajtón keresztül tudjuk elérni, ezért az ed(4) és miibus(4) meghajtókon kívül az összes többi elhagyható.

3. A hálózati kapcsolat beállítása

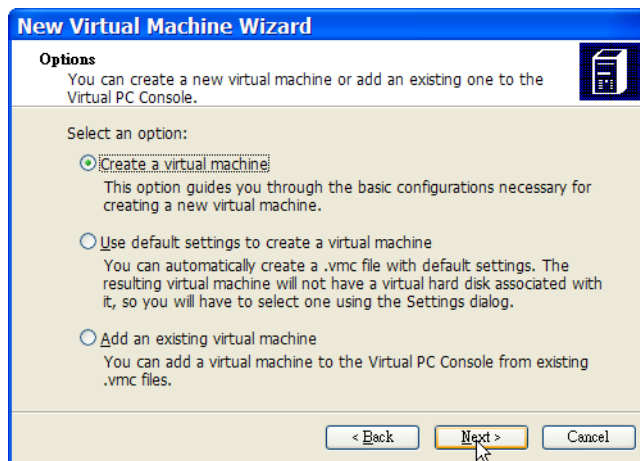
Az alapvető hálózati beállítás a virtuális gépünkön a DHCP aktiválása, aminek segítségével csatlakozni tudunk arra a helyi hálózatra, amelyen maga a gazda Mac is megtalálható. Ezt az alábbi sor felvételével tudjuk megoldani az `/etc/rc.conf` állományba: `ifconfig_ed0="DHCP"`. Bővebb információkért járuljunk a 31 fejezet fejezetéhez.

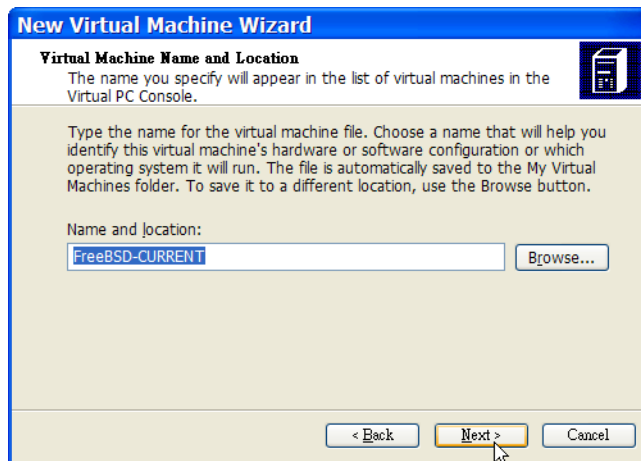
22.2.2. Virtual PC-vel Windows-on

A Windows-ra fejlesztett **Virtual PC** a Microsoft egyik szabadon letölthető szoftverterméke. A rendszerkövetelményeit bővebben lásd a <http://www.microsoft.com/windows/downloads/virtualpc/sysreq.mspx> linken. Miután telepítettük a Microsoft Windows-ra a **Virtual PC** alkalmazást, be kell állítanunk egy virtuális gépet, majd telepítenünk kell rá a kívánt vendég operációs rendszert.

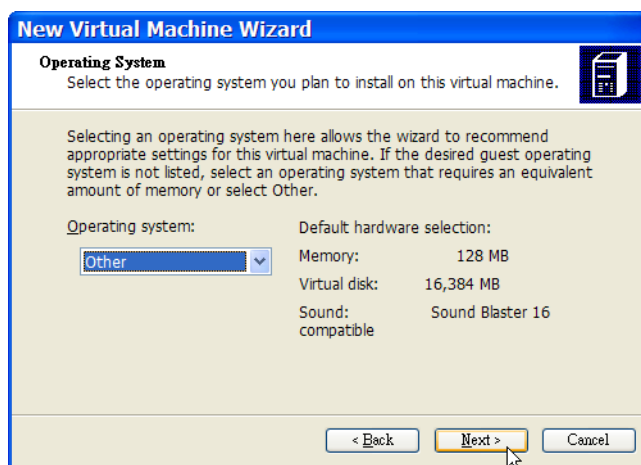
22.2.2.1. A FreeBSD telepítése Virtual PC/Microsoft® Windows-ra

Amikor a FreeBSD-t a Microsoft Windows és **Virtual PC** párosra akarjuk telepíteni, akkor kezdjünk egy új virtuális gép létrehozásával. Ehhez válasszuk ki a menüből a **Create a virtual machine** (Virtuális gép létrehozása) pontot.

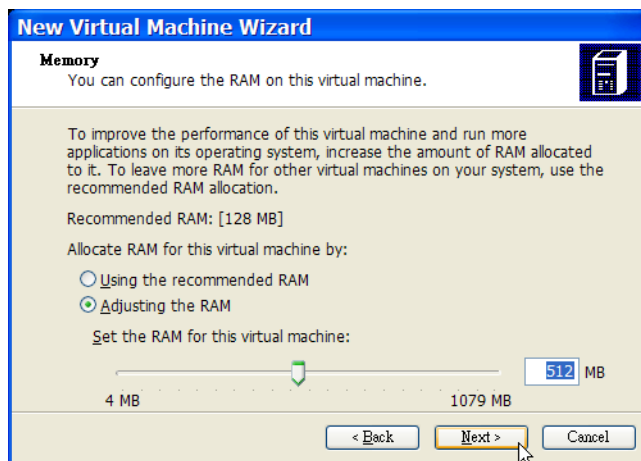


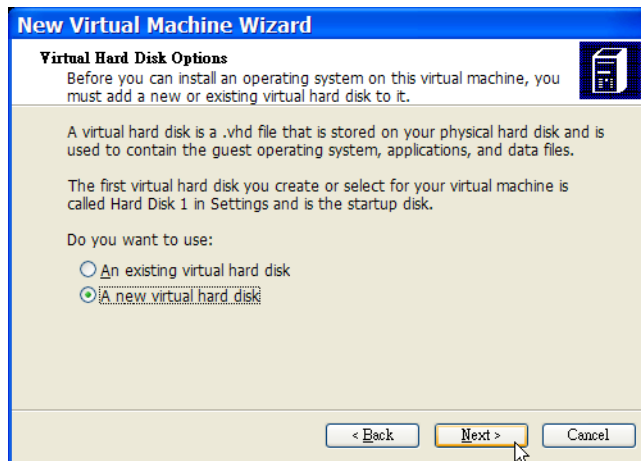


Majd válasszuk az Operating system (Operációs rendszer) beállításánál az Other (Egyéb) opciót.

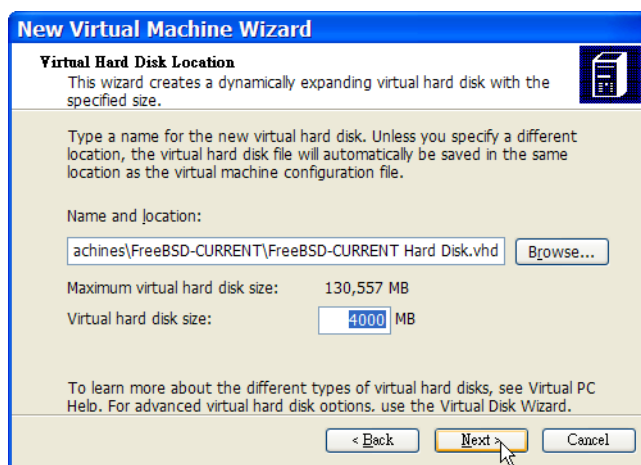


Ezután válasszuk ki a szándékainknak megfelelően a telepítendő FreeBSD példányhoz mért memória és lemezterület mennyiségét. Ahhoz, hogy a FreeBSD fusson **Virtual PC** alatt, 4 GB-nyi lemezterület és 512 MB RAM beállítása a legtöbb esetben kiválóan megfelelő.

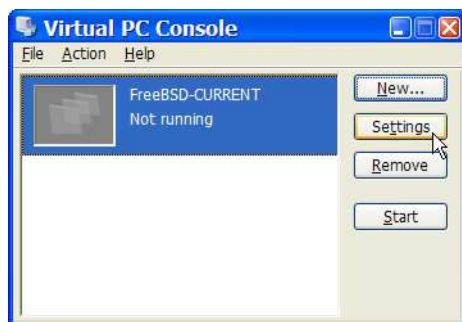


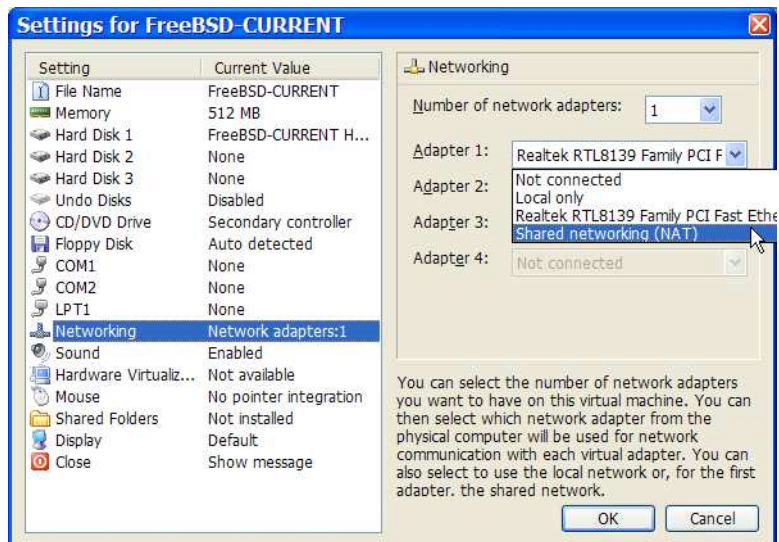


Mentsük el és fejezzük be a konfigurációt.



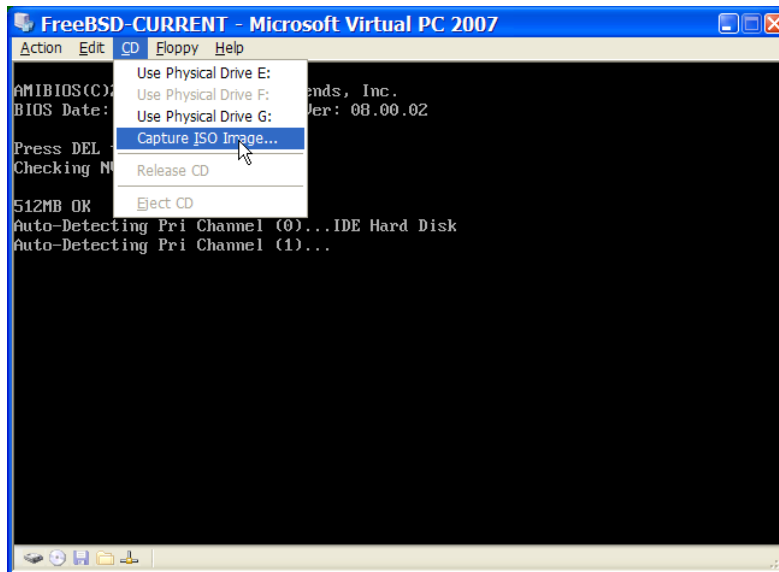
Válasszuk ki a FreeBSD-s virtuális gépünket, majd kattintsunk a **Settings** (Beállítások) menüre és állítsuk be hálózati csatoló és hálózatkezelés típusát.



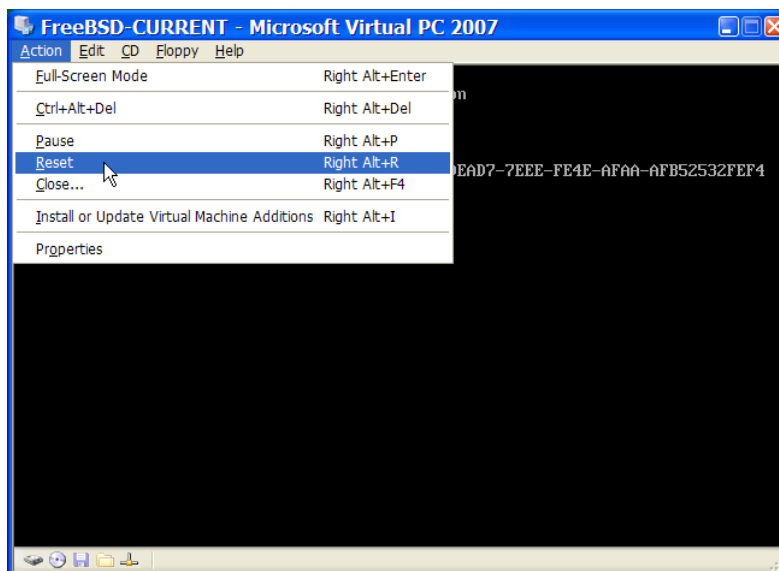


A FreeBSD-nek otthont adó virtuális gépünk létrehozása után telepítenünk is kell rá a rendszert. Ez legegyszerűbben a hivatalos FreeBSD telepítő CD-vel vagy a hivatalos FTP oldalról letölthető CD-képpel tehetjük meg. Amikor letöltöttük a megfelelő CD-képet a helyi Windows-os állományrendszerünkre vagy behelyeztük a telepítéshez használható CD-t a CD-meghajtónkba, a FreeBSD-s virtuális gépünk elindításához kattintsunk rá duplán. Ezt követően a **Virtual PC** ablakában kattintsunk a CD menüre és válasszuk ki belőle a **Capture ISO Image...** (Lemezkép használata...) pontot. Ennek hatására megjelenik egy ablak, amiben a virtuális gépünk CD-meghajtóihoz tudunk csatlakoztatni lemezképeket vagy akár létező CD-meghajtókat.

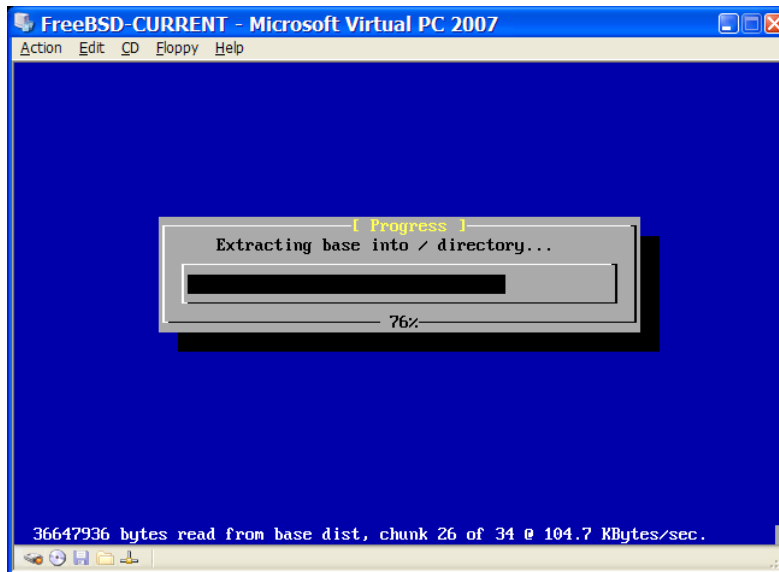




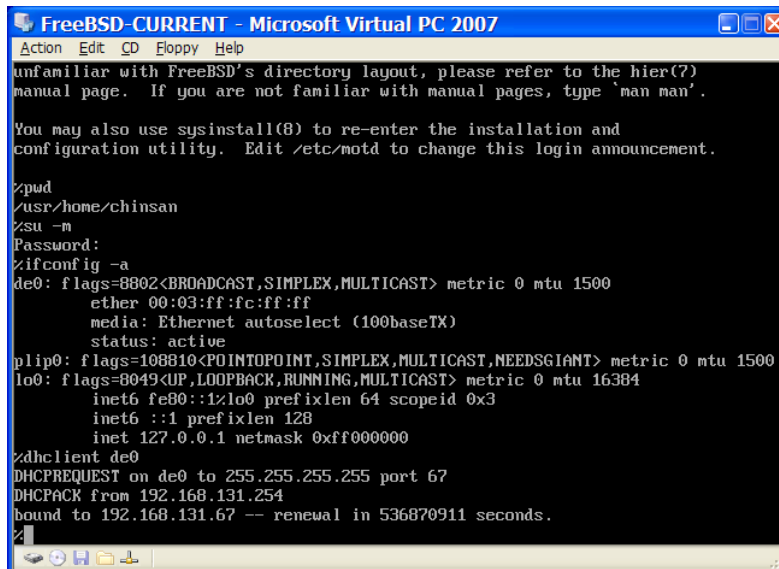
Miután sikeresen beállítottuk a telepítő CD forrását, indítsuk újra a virtuális gépet az Action (Művelet) menün belül a Reset (Újraindítás) pont kiválasztásával. Így a **Virtual PC** újraindítja a virtuális rendszert egy olyan speciális BIOS használatával, amely a normális BIOS-hoz hasonlóan először megkeresi az elérhető CD-meghajtókat.



Ebben az esetben a FreeBSD telepítőeszközét fogja megtalálni és megkezdni a 2 fejezetben ismertetett szokásos, **sysinstall** programra alapuló telepítési eljárást. Ennek során az X11-et is feltelepíthetjük, habár egyelőre még ne állítsuk be.



Ne felejtjük el kivenni a meghajtóból a telepítéshez használt CD-t vagy elengedni a megfelelő lemezképet, amikor befejeződött a telepítés. Végezetül indítsuk ismét újra a frissen telepített FreeBSD-s virtuális gépünket.



22.2.2.2. A FreeBSD beállítása a Microsoft Windows/Virtual PC-n

Miután a FreeBSD-t minden gond nélkül telepítettük a Microsoft Windows-on futó **Virtual PC**-re, még további beállítási lépéseket is meg kell tennünk a rendszer virtualizált működésének finomhangolásához.

1. A rendszertöltő változóinak beállítása

A legfontosabb teendőnk csökkenteni a `kern.hz` konfigurációs beállítás értékét, aminek köszönhetően vissza tudjuk fogni a **Virtual PC** alatt futó FreeBSD processzorhasználatát. Ezt úgy tudjuk megtenni, ha a `/boot/loader.conf` állományba felvesszük a következő sort:

```
kern.hz=100
```

Enélkül a **Virtual PC** alatt üresjáratban futó FreeBSD vendég operációs rendszer egy egyprocesszoros számítógép idejének durván 40%-át foglalja le. A változtatás után azonban ez az érték pusztán közel 3%-ra csökken le.

2. Új konfigurációs állomány létrehozása a rendszermaghoz

Nyugodtan eltávolíthatjuk a SCSI, FireWire és USB eszközmeghajtókat. A **Virtual PC** által felajánlott virtuális hálózati csatolót a de(4) meghajtón keresztül tudjuk használni, ezért a de(4) és miibus(4) eszközön kívül az összes többi hálózati eszköz támogatása kizsedhető a rendszermagból.

3. A hálózati kapcsolat beállítása

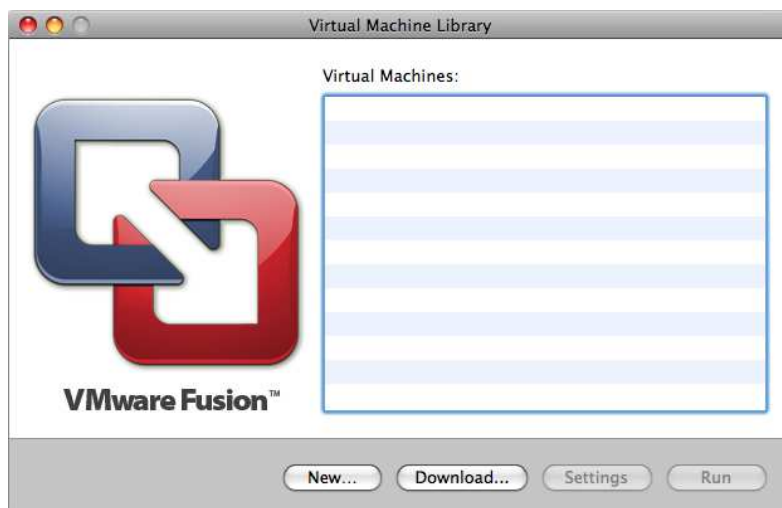
A legalapvetőbb hálózati beállítás csupán annyiból áll, hogy DHCP-n keresztül csatlakoztatjuk a virtuális gépünket ugyanahhoz a helyi hálózathoz, amiben a gazda Microsoft Windows-os gépünk is megtalálható. Ezt úgy tudjuk elérni, ha a `/etc/rc.conf` állományba megadjuk a `ifconfig_de0="DHCP"` sort. A komolyabb hálózati beállításokat a 31 fejezetben találhatjuk.

22.2.3. VMWare-rel MacOS-en

A Mac-ek számára fejlesztett **VMWare Fusion** egy olyan kereskedelmi termék, amit az Intel alapú Apple Mac gépekre tudunk telepíteni a Mac OS 10.4.9 és későbbi változatain. A FreeBSD itt egy teljesen támogatott vendég operációs rendszer. Miután a **VMWare Fusion** felkerült a Mac OS X rendszerünkre, be kell állítanunk a virtuális gépet és telepítenünk rá a vendég operációs rendszert.

22.2.3.1. A FreeBSD telepítése a Mac OS X/VMWare-re

Először indítsuk el a VMWare Fusion-t, aminek eredményeképpen betöltődik a Virtual Machine Library. Egy új virtuális gépre létrehozásához kattintsunk a "New" gombra:



Ekkor bejön az új gép összeállítását segítő New Virtual Machine Assistant, ahol a továbblépéshez kattintsunk a Continue gombra:



Az operációs rendszerek (Operating System) közül válasszuk az “egyéb” (Other) kategóriát, majd a Version fülön a FreeBSD vagy a FreeBSD 64-bit változatot attól függően, hogy 32 bites vagy 64 bites támogatásra van szükségünk:



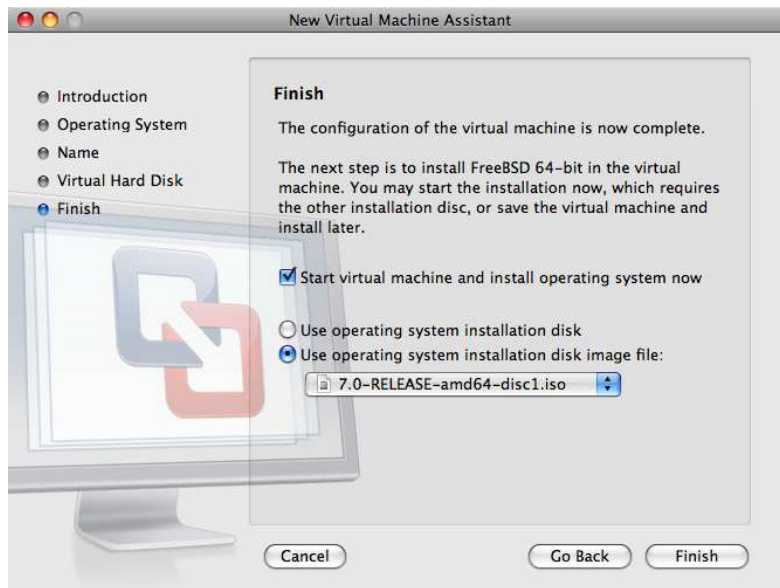
Adjuk meg a virtuális gép képének nevét és a könyvtárat, ahova el akarjuk menteni:



Válasszuk meg a virtuális géphez tartozó virtuális merevlemez méretét is:



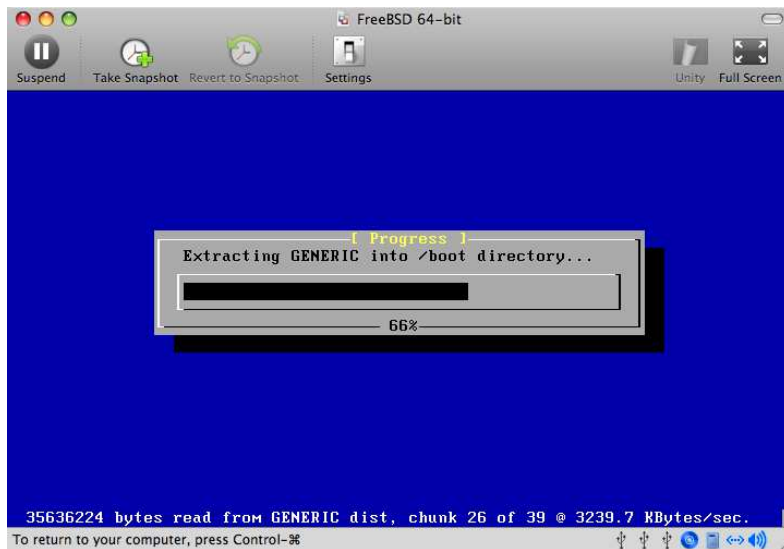
Mondjuk meg, hogy milyen módon szeretnénk telepíteni a virtuális gépre, ISO formátumú lemezképről vagy CD-ről:



Ahogy a Finish feliratú gombra kattintunk, a virtuális gép máris elindul:

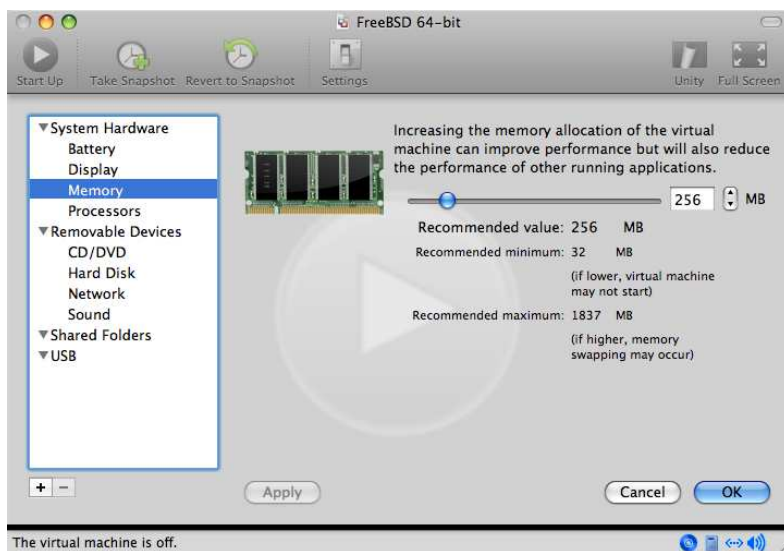


Telepítsük fel a FreeBSD-t a megszokott módon vagy a 2 fejezet utasításai mentén:



Miután befejeződött a telepítés, módosítsuk a virtuális gép beállításait, például a memória mennyiségét:

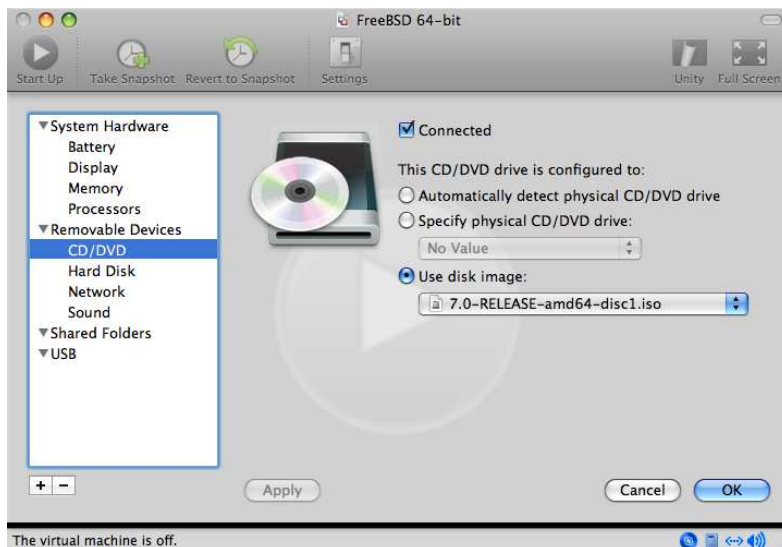
Megjegyzés: A virtuális gép hardveres beállításai a futása alatt nem változtathatóak meg.



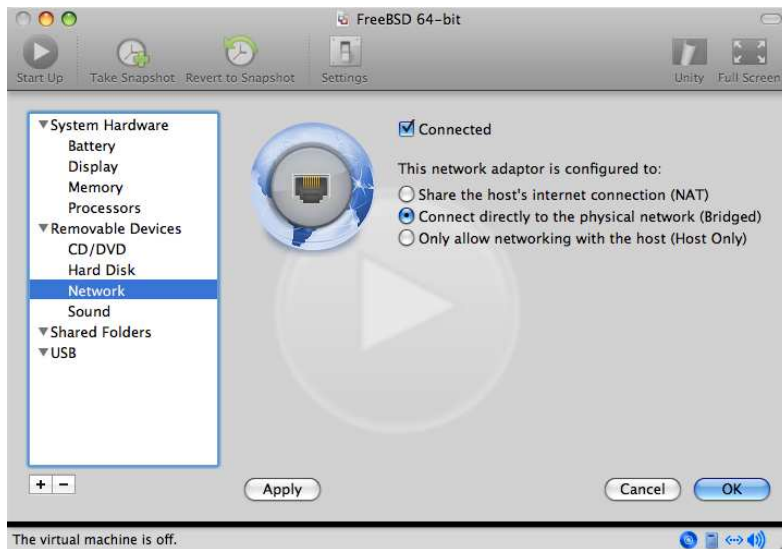
A virtuális gép által használható processzorok számát:



A CD-meghajtó állapotát. Általában lehetőségünk van a virtuális gépet leválasztani a CD-meghajtóról vagy ISO lemezképről, ha már nem használjuk.



A hálózati csatlakozás a virtuális géppel kapcsolatban utolsóként beállítandó tényező. Ha a befogadó gépen kívül még más gépeket is el akarunk érni a virtuális gépről, akkor ehhez mindenképpen a **Connect directly to the physical network (Bridged)** opciót válasszuk. Minden más esetben a **Share the host's internet connection (NAT)** az ajánlott, mivel így a virtuális gép eléri az internetet, de a hálózatról nem lehet azt elérni.



Miután befejeztük a beállítások finomhangolását, indítsuk is el a frissen telepített FreeBSD-s virtuális gépünket.

22.2.3.2. A FreeBSD beállítása a Mac OS X/VMWare-en

Ahogy a FreeBSD-t sikeresen telepítettük a Mac OS X alatt futó **VMWare**-re, néhány konfigurációs lépést még meg kell tennünk a virtualizált rendszer teljesítmények optimalizálása érdekében.

1. A rendszertöltő változóinak beállítása

A legfontosabb lépés talán a `kern.hz` változó értékének csökkentése, amivel a **VMWare** alatt futó FreeBSD processzorhasználatát szoríthatjuk vissza. Ezt a következő sor hozzáadásával érhetjük el a `/boot/loader.conf` állományban:

```
kern.hz=100
```

Enélkül az üresjáratban zakatoló FreeBSD-s **VMWare** vendég nagyjából az iMac egyik processzorának 15%-át emésztí fel. Ezzel a módosítással azonban ez lenyomható közel 5%-ra.

2. Új konfigurációs állomány létrehozása a rendszermaghoz

Nyugodtan törölhetjük az összes FireWire és USB eszköz meghajtóját. A **VMWare** egy `em(4)` meghajtón keresztül elérhető virtuális hálózati kártyát biztosít, így az `em(4)` kivételével az összes hálózati eszköz meghajtóját kivethetjük a rendszermagból.

3. A hálózat beállítása

A legegyszerűbb hálózati beállítás mindösszesen a DHCP használatát igényli, aminek révén a virtuális gépünk a befogadó Mac-kel egy helyi hálózatra kerül. Ezt úgy tudjuk engedélyezni, ha az `/etc/rc.conf` állományba felvesszük az `ifconfig_em0="DHCP"` sort. Ha ennél komolyabb hálózati beállítások is érdekelnek minket, akkor olvassuk el a 31 fejezetet.

22.3. A FreeBSD mint gazda

Gazda operációs rendszerként a FreeBSD évekig nem kapott hivatalosan támogatást egyetlen elterjedtebb virtualizációs megoldás részéről sem. Sokan erre a célra eddig a **VMWare** korábbi és inkább már elavult, a Linux kompatibilitási rétegre épülő változatait (mint például `emulators/vmware3`) használták. Nem sokkal azonban a FreeBSD 7.2 megjelenése után a Sun **VirtualBox™** OSE (Open Source Edition) natív FreeBSD alkalmazásként bukkant fel a Portgyűjteményben.

A **VirtualBox** egy folyamatos fejlesztés alatt álló, komplett virtualizációs csomag, amely immáron elérhető a legtöbb népszerű operációs rendszerre, mint a Windows, Mac OS, Linux és a FreeBSD. Egyaránt képes Windows és UNIX fajtájú vendégrendszerek futtatására. Nyílt- és zárt forráskódú változatban is elérhető. A felhasználók szempontjából a kettő közti talán legfontosabb eltérés, hogy a nyílt forráskódú változat nem tartalmaz USB támogatást. A különbségek teljes listája megtalálható a **VirtualBox** wiki "Editions" oldalán, a <http://www.virtualbox.org/wiki/Editions> címen. FreeBSD alatt jelenleg csak a nyílt forráskódú változat érhető el.

22.3.1. A VirtualBox™ telepítése

A **VirtualBox** a `emulators/virtualbox-ose` könyvtárból érhető el portként, és onnan a következő parancsokkal telepíthető:

```
# cd /usr/ports/emulators/virtualbox-ose
# make install clean
```

A beállítások közt az egyik leghasznosabb a GuestAdditions nevű programcsomag telepítése. A benne található programokon keresztül a vendégként futó operációs rendszer számos hasznos szolgáltatását el tudjuk érni, úgy mint az egérmutató integrációját (ekkor az egérkurzor zökkenőmentesen használható a gazda és a vendég rendszerben is) vagy a videomemória gyorsabb elérését (különösen Windows esetében). A vendégekhez telepíthető ilyen jellegű kiegészítések az adott rendszer telepítése után a **Devices** menüből érhetőek el.

A **VirtualBox** első indítása előtt el kell még végeznünk néhány további beállítást. Fontos tudnunk, hogy a port a telepítés során a `/boot/modules` könyvtárba tesz még egy rendszermagmodult is, amelyet még külön be kell töltenünk:

```
# kldload vboxdrv
```

Ehhez még vegyük fel a következő sort a `/boot/loader.conf` állományba, így a modul a rendszer minden egyes indításakor magától betöltődik:

```
vboxdrv_load="YES"
```

A **VirtualBox** 3.1.2 előtti változatai ezenkívül még igénylik a `proc` állományrendszer csatlakoztatását is. Az újabb változatokban erre már nincs szükség, mivel ezekben helyette már a `sysctl(3)` könyvtár függvényeit használják.

Ha viszont a port valamelyik korábbi változatát használjuk, akkor kövessük a lentebb szereplő utasításokat és csatlakoztassuk a `proc` állományrendszert:

```
# mount -t procfs proc /proc
```

Ha hozzáadjuk az alábbi sort a `/etc/fstab` állományhoz, akkor ez a beállítás is megmarad a rendszer újraindítása után:

```
proc      /proc    procfs   rw      0        0
```

Megjegyzés: Nagyon valószínű, hogy `proc` állományrendszerrel van gondunk, amikor a következő hibaüzenetet kapjuk a **VirtualBox** indításakor:

```
VirtualBox: supR3HardenedExecDir: couldn't read "", errno=2 cchLink=-1
```

Ilyenkor a `mount` parancs kiadásával ellenőrizzük az állományrendszer sikeres csatlakoztatását.

A **VirtualBox** telepítése során keletkezik még egy `vboxusers` nevű csoport. Ide azokat a felhasználókat vegyük fel, akik részére szeretnénk engedélyezni a **VirtualBox** használatát. A csoportba új tagokat például a `pw` paranccsal tudunk felvenni:

```
# pw groupmod vboxusers -m felhasználónév
```

Ezek után a **VirtualBox** indításához válasszuk a grafikus környezetünk menüjében található Sun VirtualBox menüpontot, vagy egy terminálban gépeljük be ezt a parancsot:

```
% VirtualBox
```

A **VirtualBox** beállításának további lehetőségeiről a <http://www.virtualbox.org/> címen elérhető hivatalos honlapon olvashatunk. Tekintettel arra, hogy a FreeBSD port még viszonylag friss és folyamatos fejlesztés alatt áll, ehhez még érdemes átolvasnunk a FreeBSD wikiben szereplő <http://wiki.FreeBSD.org/VirtualBox/> oldalt is, ahol a vele kapcsolatos legfrissebb információkat és egyéb tudnivalókat találhatjuk.

Fejezet 23. Honosítás: Az I18N/L10N használata és beállítása

Írta: Andrey Chernov. Átdolgozta: Michael C. Wu.

23.1. Áttekintés

A FreeBSD felhasználói földrajzi elhelyezkedésüket tekintve mindenhol megtalálhatóak a világon. Ebben a fejezetben ismertetjük a FreeBSD honosításához és idegennyelvre fordításához alkalmazható eszközöket, amelyek segítségével az angol nem, vagy csak kevésbé ismerő felhasználók is képesek lesznek komolyabban használni. Az i18n megvalósítása rengeteg szemszögből megközelíthető rendszer és alkalmazás szintjén egyaránt, ezért ahol szükséges, hivatkozni fogunk az odaillő forrásokra.

A fejezet elolvasása során megismerjük:

- milyen nyelveket és nyelvi beállításokat találhatunk napjaink operációs rendszereiben;
- hogyan használjuk a nyelvi beállításokat a saját parancsértelmezőnkben;
- hogyan állítsuk be a konzolt az angolon kívül más nyelvekhez;
- hogyan használjuk ténylegesen az X Window Systemet a különböző nyelvekkel;
- hol olvashatunk többet az I18N-kompatibilis alkalmazások fejlesztéséről.

A fejezet elolvasásához ajánlott:

- külső alkalmazások telepítésének ismerete (4 fejezet).

23.2. Az alapok

23.2.1. Mi az I18N/L10N?

A fejlesztők az I18N elnevezést az angol “internationalization” (idegennyelvűség) szóból származtatják, amiben a szám az első és utolsó betű (az “I” és “N”) közt állók mennyiségére utal. Ehhez hasonlóan keletkezett az L10N a “localization” (honosítás) kifejezésből. Ezek házasságából jöttek létre az I18N/L10N módszerei, protokolljai és mindazon alkalmazásai, melyekkel a felhasználók a választott nyelvüket használni tudják.

Az I18N alkalmazások céljak eléréséhez függvénykönyvtárakban implementált I18N készleteket használnak. Ezzel lehetővé válik a fejlesztők számára, hogy összegyűjtsék a programukban megjelenő összes szöveget egyetlen állományba, majd azt külön lefordítsák a különböző nyelvekre. Mi is ezen konvenció követésére szeretnénk biztatni minden programozót.

23.2.2. Miért használjuk az I18N/L10N-t?

Az I18N/L10N mindenhol jól jöhet, ahol idegennyelvű adatot akarunk megjeleníteni, bekérni vagy feldolgozni.

23.2.3. Milyen nyelveket támogat az I18N?

Az I18N és L10N nem korlátozódik a FreeBSD tudására. Jelenleg a világban beszélt legelterjedtebb nyelvek mindegyikét használhatjuk bennük. Csak hogy néhányat említsünk közülük: kínai, német, japán, koreai, francia, orosz, vietnámi és még sok más.

23.3. A honosítás használata

Az I18N minden adottságával együtt független a FreeBSD-től, egy egyezményes rendszer. Mindenkit bátorítunk arra, hogy segítse a FreeBSD-t ennek az egyezménynek a betartásában.

A honosítás beállításai három főbb részre tagolhatóak: a nyelv kódja, az ország kódja és a kódolás. A nyelvi beállítások nevei is ezekből állnak össze, az alábbi séma szerint:

NyelviKód_OrszágKód.Kódolás

23.3.1. A nyelv és az ország kódja

Ha a FreeBSD (vagy bármilyen más, az I18N-t ismerő) rendszert honosítani akarunk az adott nyelvre, akkor a felhasználónak ismernie kell az adott országra és nyelvre vonatkozó kódokat (az országcód fogja elárulni az alkalmazásnak, hogy a nyelv melyik változatát használja). Ezenkívül a böngészők, SMTP/POP szerverek és webszerverek stb. is ennek alapján fognak döntéseket hozni. Íme néhány nyelv/ország kódja:

Nyelv/ország kódja	Leírás
en_US	Angol - Egyesült Államok
ru_RU	Orosz - Oroszország
zh_TW	Hagyományos kínai - Tajvan

23.3.2. Kódolások

Bizonyos nyelvek 8 bites, széles vagy több byte-os, nem ASCII kódolású karaktereket használnak, melyekről a multibyte(3) man oldalán olvashatunk részletesebben. Ezeket régebbi alkalmazások egyáltalán nem ismerik fel, és hibásan vezérlőkaraktereknek tulajdonítják. Az újabbak általában már felismerik a 8 bites karaktereket. A felhasználóknak az alkalmazásokat a széles vagy a több byte-os karakterek használatához vagy újra kell fordítaniuk, vagy pedig megfelelően be kell állítaniuk, az implementációtól függően. A széles vagy több byte-os karakterek beolvasásához és feldolgozásához a FreeBSD Portgyűjtemény (<http://www.FreeBSD.org/ports/index.html>) nyelvenként tartalmaz különféle programokat. A konkrét részletek megértéséhez olvassuk el az érintett FreeBSD portok I18N dokumentációját.

Vagyis a felhasználóknak át kell nézniük az alkalmazáshoz tartozó dokumentációt, mivel ebből tudhatják meg, hogyan állítsák be ezeket megfelelően vagy milyen értékeket adjanak át a configure/Makefile/fordító hármasnak.

Amiket esetleg érdemes lehet ezzel kapcsolatban észben tartanunk:

- A nyelvfüggő egyszerű karakteres készletek (lásd multibyte(3)), például ISO8859-1, ISO8859-15, KOI8-R, CP437.

- A széles vagy több byte-os kódolások, például az EUC, Big5.

A karakterkészletek jelenleg elérhető listáját meg tudjuk tekinteni az IANA adatbázisában (<http://www.iana.org/assignments/characters-sets>).

Megjegyzés: A FreeBSD helyettük X11-kompatibilis nyelvi kódolásokat használ.

23.3.3. I18N alkalmazások

A FreeBSD port- és csomagrendszerében az I18N alkalmazások a könnyebb felismerhetőség érdekében a nevükben tartalmazzák az I18N megnevezést. Nem minden esetben támogatják a szükséges nyelvet.

23.3.4. A nyelvi beállítások megadása

Általában elegendő annyi, hogy a kívánt nyelvi beállítás nevét exportáljuk az általunk használt parancsértelmező LANG környezeti változójába. Ez megtehető a felhasználói könyvtárunkban található `~/.login_conf`, vagy a felhasználói parancsértelmező indító állományában (`~/.profile`, `~/.bashrc`, `~/.cshrc`). Nem szükséges a nyelvi beállítások részleteit, mint például az `LC_CTYPE`, `LC_CTIME` változókat, megadni. A pontosabb részleteket a FreeBSD adott nyelvre vonatkozó dokumentációjában találjuk meg.

A következő két környezeti változót kell megadnunk az említett konfigurációs állományokban:

- A LANG változót a POSIX setlocale(3) családjának
- A MM_CHARSET változót az alkalmazás MIME karakterkészletéhez

Ez magában foglalja a felhasználói parancsértelmezőt, az adott alkalmazás és az X11 beállítását.

23.3.4.1. A nyelvi beállítások megadásának módszerei

Két módszer létezik a nyelvi beállítások megadására, ezen kettőről fogunk a továbbiakban beszélni. Az első (és egyben ajánlott) ezek közül a bejelentkezési osztályban levő környezeti változók beállítása, a második pedig környezeti változók hozzáadása a parancsértelmező rendszerszintű indító állományához.

23.3.4.1.1. Beállítás a bejelentkezési osztályokkal

Ezzel a módszerrel a nyelvi beállítás nevéhez és a MIME karakterkészlethez kötődő környezeti változókat az összes létező parancsértelmező számára csak egyszer kell megadnunk ahelyett, hogy külön mindegyikük indítóállományában szerepeltetnénk. A felhasználó a saját részét maga is elvégezheti, míg a rendszer szintjén adminisztrátori jogosultságokat igényel.

23.3.4.1.1.1. Felhasználói szintű beállítás

Íme példa gyanánt a felhasználó könyvtárában egy egyszerű `.login_conf` állomány, amiben mind a két változót Latin-1 kódolásra állítottuk:

```
me:\
    : charset=ISO-8859-1:\
```

```
:lang=de_DE.ISO8859-1:
```

Ebben a `.login_conf` példában a változókat BIG-5 kódolású hagyományos kínai nyelvre állítjuk. Észrevehetjük, hogy itt sokkal több változó beállítására van szükségünk, mivel egyes szoftverek nem kezelik megfelelően a nyelvi beállításokat kínai, japán és koreai nyelvek esetén.

```
# Azok a felhasználók, akik nem kívánnak tajvani pénz- vagy idő formátumot
# használni, egyenként írják át a változókat
me:\
    :lang=zh_TW.Big5:\
    :setenv=LC_ALL=zh_TW.Big5:\
    :setenv=LC_COLLATE=zh_TW.Big5:\
    :setenv=LC_CTYPE=zh_TW.Big5:\
    :setenv=LC_MESSAGES=zh_TW.Big5:\
    :setenv=LC_MONETARY=zh_TW.Big5:\
    :setenv=LC_NUMERIC=zh_TW.Big5:\
    :setenv=LC_TIME=zh_TW.Big5:\
    :charset=big5:\
    :xmodifiers="@im=gcin": # a gcin beállítása XIM szervertől
```

A többit lásd a Rendszergazdai szintű beállítások résznél és a `login.conf(5)` man oldalon.

23.3.4.1.1.2. Rendszergazdai szintű beállítás

Ellenőrizzük, hogy a felhasználó `/etc/login.conf` állományban szereplő bejelentkezési osztálya a megfelelő nyelvet állítja be. Győződjünk meg róla, hogy az alábbi beállítások helyet kapnak az `/etc/login.conf` állományban:

```
nyelv_neve | A hozzáférés típusának leírása:\
    :charset=MIME_karakterkészlet:\
    :lang=nyelvi_beállítás_neve:\
    :tc=default:
```

Folytassuk tovább az előbbi Latin-1-es példánk szerint:

```
nemet | Nemet felhasználók hozzáferesei:\
    :charset=ISO-8859-1:\
    :lang=de_DE.ISO8859-1:\
    :tc=default:
```

Mielőtt megváltoztatnánk a felhasználók bejelentkezési osztályait, adjuk ki a következő parancsot:

```
# cap_mkdb /etc/login.conf
```

Ezzel a `/etc/login.conf` új tartalma láthatóvá válik a rendszer számára.

A bejelentkezési osztály megváltoztatása a `vipw(8)` programmal

A `vipw` segédprogramot új felhasználók hozzáadására használjuk, aminek eredményeképpen egy ehhez hasonló bejegyzést tudunk létrehozni:

```
felhasznalo:jelszo:1111:11:nyelv:0:0:Felhasznalo neve:/home/felhasznalo:/bin/sh
```

A bejelentkezési osztály megváltoztatása az adduser(8)-rel

Az adduser-rel az alábbiak szerint tudunk új felhasználókat felvenni a rendszerbe:

- Adjuk hozzá a `defaultclass = nyelv` sort az `/etc/adduser.conf`-hoz. Ne felejtjük el, hogy ezután minden olyan felhasználónál a `default` bejelentkezési osztályt meg kell adni, akik nem ezt a nyelvet használják.
- Egy másik megoldás lehet, hogy a `adduser(8)` használata során minden felhasználó esetén külön megadjuk a nyelvet az

```
Enter login class: default []:
```

rész megjelenésekor.

- Vagy használhatjuk az alábbi az egyes eltérő nyelvű felhasználók hozzáadásánál:

```
# adduser -class nyelv
```

A bejelentkezési osztály megváltoztatása a pw(8)-vel

Amennyiben a `pw(8)`-t használjuk új felhasználók hozzáadására, így érdemes meghívunk:

```
# pw useradd felhasználó_neve -L nyelv
```

23.3.4.1.2. Beállítás a parancsértelmező indító állományával

Megjegyzés: Ezt a módszert nem javasoljuk, mivel parancsértelmezőnként eltérő beállítást kíván. Használjuk helyette a bejelentkezési osztályokkal megvalósított módszert.

A nyelvi beállítás nevének és a MIME karakterkészlet beállításához egyszerűen csak adjuk meg a lenti `/etc/profile` és/vagy `/etc/csh.login` parancsértelmező indító állományokban bemutatott környezeti változót. Továbbra is a német nyelvet használjuk a példánkban:

Az `/etc/profile` esetén:

```
LANG=de_DE.ISO8859-1; export LANG
MM_CHARSET=ISO-8859-1; export MM_CHARSET
```

Vagy a `/etc/csh.login` esetén:

```
setenv LANG de_DE.ISO8859-1
setenv MM_CHARSET ISO-8859-1
```

Úgy is megoldhatjuk ezt a feladatot, ha fenti utasításokat a `/usr/share/skel/dot.profile` (hasonló a fentebb említett `/etc/profile` állományhoz) vagy `/usr/share/skel/dot.login` (hasonló a fentebb említett `/etc/csh.login` állományhoz) esetén hajtjuk végre.

X11 esetén:

Adjuk meg a `$HOME/.xinitrc` állományban:

```
LANG=de_DE.ISO8859-1; export LANG
```

Vagy:

```
setenv LANG de_DE.ISO8859-1
```

Attól függően, milyen parancsértelmezőt használunk (lásd fentebb).

23.3.5. A konzol beállítása

Az összes egyszerű karakteres készlet esetén a kérdéses nyelvhez megfelelő konzolos betűtípust az `/etc/rc.conf` állományban tudjuk beállítani:

```
font8x16=betűtípus_neve
font8x14=betűtípus_neve
font8x8=betűtípus_neve
```

Itt a `betűtípus_neve` az `.fnt` kiterjesztés elhagyásával a `/usr/share/syscons/fonts` könyvtárban található állományok nevéből adható meg.

Ha szükséges állítsuk még be a megfelelő billentyű- és betűkiosztást is a `sysinstall` segítségével. Ahogy sikerült elindítanunk a **sysinstall**-t, válasszuk a **Configure** (Beállítások) pontot, majd a **Console** (Konzol)-t! Vagy ehelyett beírhatjuk az alábbi sorokat a `/etc/rc.conf` állományba:

```
scrnmap=betűkiosztás_neve
keymap=billentyűkiosztás_neve
keychange="funkcióbillentyű_sorszáma szekvencia"
```

Itt a `betűkiosztás_neve` a `/usr/share/syscons/scrnmaps` könyvtárban található állományok nevéből származtatható az `.scm` kiterjesztés elhagyásával. A betűkiosztásokat általában a 9 bites karaktermátrixszal rendelkező VGA megjelenítők problémáinak megoldására lehet használni, mivel így az eredetileg 8 bittel ábrázolt betűket ki lehet tolni az ilyen típusú kártyák pszeudografikus területéről.

Ha aktiváltuk a **moused** egérkezelő démont az `/etc/rc.conf` állományban az alábbi sor megadásával:

```
moused_enable="YES"
```

akkor a következő bekezdésben rá is térhetünk az egérmutató adatainak vizsgálatára.

A `syscons(4)` meghajtóban található egérmutató alapértelmezés szerint a `0xd0 - 0xd3` karaktereket foglalja el a karakterkészletben. Ha a nyelv ezeket használja, arrébb kell költöztetnünk ezt az egérmutató által elfoglalt sávot. A FreeBSD-ben az `/etc/rc.conf` állományon keresztül érhetjük el:

```
mousechar_start=3
```

A `billentyűkiosztás_neve` a `/usr/share/syscons/keymaps` könyvtárból, a `.kbd` kiterjesztés elhagyásával keletkezik. Ha nem vagyunk benne biztosak, melyik kiosztást is kellene használnunk, a `kbdmap(1)` segítségével a rendszer újraindítása nélkül kipróbálhatjuk a rendelkezésre álló billentyűkiosztásokat.

A `keychange` használatára többnyire a funkcióbillentyűk adott termináltípushoz egyeztetéséhez van szükség, mert a funkcióbillentyűk szekvenciái nem adhatóak meg a billentyűkiosztásban.

Ezeket felül érdemes megbizonyosodnunk róla, hogy a `/etc/ttys` állományban jól állítjuk be a terminál típusát minden `ttv*` bejegyzés esetén. Az aktuálisan előre beállított kapcsolatok a következők:

Karakterkészlet	Terminálttípus
ISO8859-1 vagy ISO8859-15	cons25l1
ISO8859-2	cons25l2
ISO8859-7	cons25l7
KOI8-R	cons25r
KOI8-U	cons25u
CP437 (alapértelmezett VGA)	cons25
US-ASCII	cons25w

A széles és több byte-os karaktereket használó nyelvek esetén használjuk a `/usr/ports/nyelv` könyvtárban megfelelő FreeBSD portot. Egyes portok konzolosként jelennek meg, miközben a rendszer soros virtuális terminálként látja ezeket, ezért fenn kell tartanunk elegendő virtuális terminált mind az X11, mind pedig pseudo-soros konzol számára. Itt látható a konzolon más nyelvet használó alkalmazások részleges listája:

Nyelv	Hely
Hagyományos kínai (BIG-5)	chinese/big5con
Japán	japanese/kon2-16dot vagy japanese/mule-freewnn
Koreai	korean/han

23.3.6. Az X11 beállítása

Habár az X11 nem része a FreeBSD projektnek, megemlítnék vele kapcsolatban néhány hasznos információt a FreeBSD felhasználók számára is. Még több részletet a Xorg honlapjáról (<http://www.x.org/>) vagy az általunk használt X11 szerver dokumentációjából tudhatunk meg.

Az `~/Xresources` állományban további I18N beállításokat finomíthatunk alkalmazásonként (például betűtípusok, menük stb.).

23.3.6.1. Betűtípusok megjelenítése

Telepítsük fel az **Xorg** (`x11-servers/xorg-server`) vagy az **XFree86** (`x11-servers/XFree86-4-Server`) szerverek valamelyikét, majd telepítsük a nyelvhez tartozó TrueType betűtípusokat. Ezután a megfelelő nyelvi beállítása megadása révén már látni fogjuk a kiválasztott nyelven megjelenő menüket és egyéb szövegeket.

23.3.6.2. Idegennyelvű karakterek bevitele

Az X11 beviteli módszerének (X11 Input Method, XIM) protokollja egy új szabvány az összes X11 klienshez. Minden X11 alkalmazást olyan XIM-kliensként kell elkészíteni, amelyek a bemenő adatokat az XIM beviteli szerverektől kapják. Különböző XIM szerverek érhetőek el az eltérő nyelvekhez.

23.3.7. Nyomtatók beállítása

Egyes egyszerű karakteres készletek általában hardveresen beépítve megtalálhatóak a nyomtatókban. A széles és több byte-os karakterkészletek azonban külön beállítást igényelnek, amire az **apsfilter** használatát javasoljuk. A megfelelő nyelvhez szabott eszközökkel át is lehet konvertálni PostScript vagy PDF formátumba a nyomtatni kívánt dokumentumot.

23.3.8. A rendszermag és az állományrendszerek

A FreeBSD gyors állományrendszere (Fast File System, FFS) szabályosan kezeli a 8 bites karaktereket, tehát tetszőleges egyszerű karakteres készlet (lásd multibyte(3)) használható vele, viszont a karakterkészlet nevét nem tárolja el az állományrendszerben. Emiatt a neveket nyersen kezeli, semmit sem tud a kódolásukról. Az FFS hivatalosan még nem támogat semmilyen fajta széles vagy több byte-os karakterkészletet. Léteznek azonban független javítások az FFS-hez, amelyek lehetővé teszik ilyen széles vagy több byte-os karakterek használatát. Ezek csak átmeneti és nem hordozható megoldások, olyan módosítások, amelyekről úgy döntöttünk, nem vesszük fel ezeket a forrásfába. Az érintett nyelvek honlapjain elérhetjük ezeket a javításokat és többet megtudhatunk róluk.

A FreeBSD MS-DOS állományrendszere konfigurálható úgy, hogy képes legyen konvertálni az MS-DOS Unicode és a kiválasztott FreeBSD állományrendszerének karakterkészlete között. Erről bővebben a `mount_msdosfs(8)` man oldalon olvashatunk.

23.4. I18N programok fordítása

Számos FreeBSD port rendelkezik I18N támogatással. Ezek egy részének nevében szerepel az -I18N jelzés. Az ilyen és sok más hasonló program beépítetten ismeri az I18N-t, így nem igényelnek külön beállításokat.

Néhány alkalmazás azonban, mint például a **MySQL**, esetén az adott karakterkészletnek megfelelő módon kell beállítani a `Makefile` állományt. Ezt általában magában a `Makefile` állományban tudjuk megtenni, vagy pedig a **configure** megfelelő paraméterezésével.

23.5. A FreeBSD honosítása adott nyelvekre

23.5.1. Az orosz nyelv (KOI8-R kódolás)

Eredetileg írta: Andrey Chernov.

A KOI8-R kódolásról bővebben a KOI8-R oldalán (orosz hálózati karakterkészlet) (<http://koi8.pp.ru>) tájékozódhatunk.

23.5.1.1. A nyelvi beállítások megadása

Írjuk a következő sorokat a `~/login_conf` állományunkba:

```
me:Az en hozzáferesem:\
    :charset=KOI8-R:\
    :lang=ru_RU.KOI8-R:
```

Valamint lásd a fejezet korábbi részeiben említett példákat a nyelvi beállítások megadására.

23.5.1.2. A konzol beállítása

- Tegyük hozzá a következő sort az `/etc/rc.conf` állományunkhoz:

```
mousechar_start=3
```

- Illetve használjuk az `/etc/rc.conf` állományban még a következő beállításokat is:

```
keymap="ru.koi8-r"
scrnmap="koi8-r2cp866"
font8x16="cp866b-8x16"
font8x14="cp866-8x14"
font8x8="cp866-8x8"
```

- A `/etc/ttys` állományban szereplő mindegyik `ttv*` bejegyzésnél adjuk meg termináltípusnak a `cons25r-t`.

Valamint lásd a fejezet korábbi részében bemutatott példákat a konzol beállítására.

23.5.1.3. A nyomtatás beállítása

Mivel a legtöbb nyomtató hardveresen tartalmazza a CP866 kódlapot az orosz karakterek támogatásához, használunk kell egy kimeneti szűrőt a KOI8-R kódolású karakterek CP866 kódolásra konvertálásához. Egy ilyen szűrő alapértelmezés szerint telepítésre kerül a `/usr/libexec/lpr/ru/koi2alt` állományba. Az orosz nyomtatóhoz tartozó bejegyzés valahogy így néz ki az `/etc/printcap` állományban:

```
lp|Orosz helyi sornyomtato:\
    :sh:of=/usr/libexec/lpr/ru/koi2alt:\
    :lp=/dev/lpt0:sd=/var/spool/output/lpd:lf=/var/log/lpd-errs:
```

A bővebben magyarázathoz lásd a `printcap(5)` man oldalt.

23.5.1.4. Az MS-DOS állományrendszere és az orosz állománynevek

A most következő példa `fstab(5)` bejegyzés azt mutatja meg, hogy lehet bekapcsolni az orosz állománynevek támogatását a csatlakoztatandó MS-DOS állományrendszereken:

```
/dev/ad0s2      /dos/c  msdos    rw,-Wkoi2dos,-Lru_RU.KOI8-R 0 0
```

Az `-L` kapcsolóval kiválasztjuk a használni kívánt nyelvi beállítás nevét, és a `-w` kapcsolóval megadjuk a karakterek átváltásához szükséges táblázatot. A `-w` kapcsoló használata során mindenképpen csatlakoztassuk a `/usr` állományrendszert még az MS-DOS partíció előtt, mivel az átváltáshoz használt táblázatok a `/usr/libdata/msdosfs` könyvtárban találhatóak meg! A részleteket a `mount_msdosfs(8)` man oldalon találhatjuk meg.

23.5.1.5. Az X11 beállítása

1. Adjuk meg először a leírtak szerint a nem X-es nyelvi beállításokat.
2. Ha **Xorg**-ot használunk, telepítsük a `x11-fonts/xorg-fonts-cyrillic` csomagot.

Ellenőrizzük a `/etc/X11/xorg.conf` állományban a "Files" szakaszt. Az alábbi sort mindegyik más `FontPath` bejegyzés *előtt* kell szerepeltetnünk:

```
FontPath "/usr/X11R6/lib/X11/fonts/cyrillic"
```

Megjegyzés: A portok között találhatunk még további cirill betűtípusokat.

3. Az orosz billentyűzet életre keltéséhez írjuk be a következőket az `xorg.conf` állomány "Keyboard" szakaszába:

```
Option "XkbLayout"      "us,ru"  
Option "XkbOptions"     "grp:toggle"
```

Ellenőrizzük, hogy a `XkbDisable` ki van kapcsolva (ki van kommentezve) ebben a szakaszban.

A `grp:toggle` beállítás esetén az orosz/latin (RUS/LAT) átkapcsolás gombja a **jobb Alt** lesz, míg a `grp:ctrl_shift_toggle` beállításnál a **Ctrl+Shift**. A `grp:caps_toggle` esetén az orosz/latin váltás a **CapsLock** billentyűvel történik. Ilyenkor (de csak latin módban) a megszokott **CapsLock** funkció továbbra is elérhető a **Shift+CapsLock** kombinációval. A `grp:caps_toggle` valamiért nem működik az **Xorgban**.

Ha van "Windows" billentyűnk a billentyűzeten és azt tapasztaljuk, hogy egyes nem-alfabetikus billentyűk rosszul kerülnek kiosztásra orosz módban, adjuk hozzá a következő sort az `xorg.conf` állományhoz:

```
Option "XkbVariant"     ",winkeys"
```

Megjegyzés: Az orosz XKB billentyűzet egyes nem honosított alkalmazások esetén nem működik.

Megjegyzés: A kis mértékben honosított alkalmazások esetén javasolt meghívni a `XtSetLanguageProc(NULL, NULL, NULL)`; függvényt valahol a program elején.

Az X11 alkalmazások honosításához további útmutatásokat a KOI8-R X Window-ra (<http://koi8.pp.ru/xwin.html>) című leírásban találhatunk.

23.5.2. Hagyományos kínai honosítás tajvaniak számára

A FreeBSD-Taiwan projekt készített a FreeBSD-hez egy kínainak szóló hogyanant, amely elérhető a <http://netlab.cse.yzu.edu.tw/~statue/freebsd/zh-tut/> címen és számos kínai portot használ. A FreeBSD kínai hogyan jelenlegi szerkesztője Shen Chuan-Hsing (statue@freebsd.sinica.edu.tw).

Chuan-Hsing Shen (statue@freebsd.sinica.edu.tw) létrehozta a Kínai FreeBSD gyűjteményt (Chinese FreeBSD Collection, CFC) (<http://netlab.cse.yzu.edu.tw/~statue/cfc/>) a FreeBSD-Taiwan zh-L10N-tut munkáját felhasználva. A hozzá tartozó csomagok és szkriptek elérhetőek a <ftp://freebsd.csie.nctu.edu.tw/pub/taiwan/CFC/> címen.

23.5.3. Honosítás német (és minden más ISO 8859-1 kódolású) nyelvre

Slaven Rezic (<eserte@cs.tu-berlin.de>) készített egy írást, amely elmagyarázza, hogyan használjunk német nemzeti karaktereket a FreeBSD alatt. Ez a leírás németül készült és a <http://user.cs.tu-berlin.de/~eserte/FreeBSD/doc/umlaute/umlaute.html> címen érhető el.

23.5.4. Honosítás görög nyelvre

Nikos Kokkalis <nickkokkalis@gmail.com> egy teljes cikket írt a FreeBSD görög nyelvi támogatásáról. Ez elérhető a FreeBSD hivatalos görög nyelvű dokumentációjában, a http://www.FreeBSD.org/doc/el_GR.ISO8859-7/articles/greek-language-support/index.html címen. Felhívjuk a figyelmet, hogy az *csak* görög nyelven érhető el.

23.5.5. Honosítás japán és koreai nyelvekre

A japán honosításhoz lásd <http://www.jp.FreeBSD.org/>, a koreaihoz pedig lásd <http://www.kr.FreeBSD.org/>.

23.5.6. Idegennyelvű FreeBSD dokumentáció

Néhány FreeBSD felhasználó lefordította a FreeBSD dokumentációjának egyes részeit más nyelvekre is. Munkájuk elérhető a főoldalon (<http://www.FreeBSD.org/index.html>) található linkeken keresztül vagy a `/usr/share/doc` könyvtárban.

Fejezet 24. A FreeBSD frissítése és frissen tartása

Átdolgozta, áttrendezte és egyes részeit aktualizálta: Jim Mock. Eredetileg írta: Jordan Hubbard, Poul-Henning Kamp, John Polstra, és Nik Clayton.

24.1. Áttekintés

A FreeBSD a kiadások közt is állandó fejlődésben van. Vannak felhasználók, akik a hivatalosan kiadott változatokat használják, és vannak, akik szeretik folyamatosan nyomonkövetni a fejlesztéseket. Emellett viszont a hivatalos kiadások esetében szükség lehet bizonyos biztonsági frissítések és kritikus javítások alkalmazására. Függetlenül a pillanatnyilag használt változattól, a FreeBSD alaprendszerében megtalálható minden olyan eszköz, amellyel könnyedén frissíteni tudunk a különböző verziók között. Ebben a fejezetben segítünk dönteni a fejlesztői változat és a kiadások használata között. Továbbá megismerhetjük a rendszer frissítéséhez használható alapvető eszközöket.

A fejezet elolvasása során megismerjük:

- milyen segédprogramokkal tudjuk frissíteni az alaprendszert és a Portgyűjteményt;
- hogyan tartsuk naprakészen rendszerünket a **freebsd-update**, **CVSup**, **CVS** vagy **CTM** használatával;
- hogyan vessük össze a telepített rendszerünk aktuális állapotát egy ismert eredeti változattal;
- hogyan frissítjük a dokumentációt **CVSup** vagy dokumentációs portok segítségével.
- a két fejlesztői ág, a FreeBSD-STABLE és a FreeBSD-CURRENT közti különbséget;
- a `make buildworld (stb.)` segítségével hogyan fordítunk és telepítünk újra az egész alaprendszert.

A fejezet elolvasásához ajánlott:

- a hálózati kapcsolatunk helyes beállítása (31 fejezet);
- a külső szoftverek telepítésének ismerete (4 fejezet).

Megjegyzés: A fejezetben a FreeBSD forrásainak frissítését a `cvsup` parancs segítségével fogjuk elvégezni. Ehhez telepítsük a `net/cvsup` portot vagy csomagot (ha a `cvsup` parancsot nem akarjuk grafikus felületen keresztül használni, akkor elegendő csak a `net/cvsup-without-gui` portot). Ha a FreeBSD 6.2-RELEASE vagy későbbi változatával rendelkezünk, akkor elegendő csak az alaprendszer részeként elérhető `csup(1)` programot használnunk.

24.2. A FreeBSD frissítése

Írta: Tom Rhodes. A megíráshoz felhasznált jegyzeteket készítette: Colin Percival.

A biztonsági javítások telepítése minden számítógépes szoftver, különösen az operációs rendszerek számára lényeges mozzanat. Nagyon hosszú ideig ez a FreeBSD esetében nem volt könnyen megoldható: a javításokat közvetlenül a forráskódon kellett elvégezni, ezekből újrafordítani a rendszert, majd telepíteni.

Ez a nehézség mostanra viszont már elhárult, mivel a FreeBSD legfrissebb verziói már tartalmaznak egy `freebsd-update` nevű segédprogramot, amellyel mindez leegyszerűsödik. Ez a program két külön funkciót lát el. Először is, lehetővé teszi, hogy a FreeBSD alaprendszer újrafordítása és -telepítése nélkül javítsunk biztonsági és egyéb apró hibákat, valamint másodsorban támogatja a kisebb és nagyobb verziójú kiadások közti váltást.

Megjegyzés: Ezek a bináris frissítések azonban csak a FreeBSD biztonsági csapata által is felügyelt architektúrák és kiadások esetén érhetőek el. Emellett bizonyos lehetőségek használatához, például a FreeBSD verziói közti átállás támogatásához a `freebsd-update(8)` legújabb változata szükségeltetik. Ezért ne felejtjük el alaposan átolvasni a legújabb kiadásokról szóló bejelentéseket mielőtt frissítenénk rájuk, mivel ezzel kapcsolatban fontos információkat tartalmazhatnak. Az említett bejelentések a <http://www.FreeBSD.org/releases/> címen érhetőek el.

Ha a `crontab` már hivatkozik a `freebsd-update` programra, akkor a most következő művelet elkezdése előtt tiltsuk le.

24.2.1. A konfigurációs állományok

Ha változtatnánk szeretnénk a frissítési folyamaton, ekkor a programhoz tartozó, `/etc/freebsd-update.conf` nevű konfigurációs állományt kell módosítanunk. Az opciók részletes ismertetéssel rendelkeznek, habár némelyiknél még további magyarázat kellhet:

```
# Az alaprendszerben frissíteni kívánt komponensek
Components src world kernel
```

Ezzel a paraméterrel határozhatjuk meg, hogy a FreeBSD mely részei kerüljenek frissítésre. Alapértelmezés szerint a program frissíti a forrásokat, a teljes alaprendszert és a rendszermagot. Komponensként a telepítésnél választható elemeket adhatjuk meg, például "world/games" hozzáadásakor a games kategória elemei is folyamatosan frissülni fognak. Az "src/bin" megadásakor pedig az `src/bin` könyvtár tartalma frissül.

Ezt a beállítást a legjobb meghagyni az alapértelmezett értéken, mivel a további elemek megadásánál egyenként fel kell sorolni a frissítendő komponenseket. Ha itt viszont kifejejtünk valamit, akkor könnyen megeshet, hogy a források és a binárisok verziója elcsúszik egymástól.

```
# Az IgnorePaths beállítás után megadott szövegre illeszkedő összes
# bejegyzés frissítése kimarad
IgnorePaths
```

Ennél a beállításnál azokat a könyvtárakat kell megadnunk, amelyeket (és tartalmukat) ki szeretnénk hagyni a frissítés során. Ezek lehetnek például a `/bin` vagy az `/sbin`. Így meg tudjuk akadályozni, hogy `freebsd-update` esetleg felülírjon valamilyen helyi változtatást a rendszerünkben.

```
# Az UpdateIfUnmodified beállítás után megadott elérési útvonalakon csak
# a felhasználó által még nem módosított állományok fognak frissülni
# (hacsak a módosításokat össze nem fésüljük, lásd lentebb)
UpdateIfUnmodified /etc/ /var/ /root/ /.cshrc /.profile
```

A megadott könyvtárakban csak azokat a konfigurációs állományokat fogja frissíteni, amelyeket nem változtattuk meg. Amennyiben bármelyikük eltér az eredetileg frissítendő változattól, azt a program nem módosítja. Létezik egy másik hasonló beállítás, a `KeepModifiedMetadata`, amely hatására a `freebsd-update` az összefésülés során elmenti a változtatásokat.

```
# A MergeChanges beállításnál szereplő állományok helyi módosításait
# automatikusan összefésüljük a FreeBSD újabb verziójára frissítése közben
MergeChanges /etc/ /var/named/etc/
```

Itt azokat a könyvtárakat adhatjuk meg, amelyekben a `freebsd-update` számára engedélyezzük a konfigurációs állományok új verziójának összefésülését a jelenlegi állapottal. Az összefésülés lényegében a `mergemaster(8)` használatánál már megszokott módon, `diff(1)` formátumban érkező módosítások sorozata alapján történik. Ekkor egy szövegszerkesztő segítségével felügyelhetjük az összefésülés menetét vagy megállíthatjuk a `freebsd-update` futását. Ha kétségeink adódnak, akkor egyszerűen mentsük le az `/etc` könyvtárat és fogadjuk el mindegyik összefésülés eredményét. A `mergemaster` működéséről a 24.7.11.1 Szakasz ad részletesebb tájékoztatást.

```
# A FreeBSD frissítésekor ezt a könyvtárat fogja a program használni a
# letöltött módosítások és az egyéb ideiglenes állományok tárolására
# WorkDir /var/db/freebsd-update
```

Az itt megadott könyvtárba fognak kerülni az elvégzendő módosítások és az egyéb ideiglenesen keletkező állományok. A verziók közti váltás során ebben a könyvtárban ajánlott legalább 1 GB szabad tárterületnek lennie.

```
# A kiadások közti váltás során a Components beállításnál megadott
# elemek kerüljenek csak frissítésre (StrictComponents yes), vagy a
# program próbálja meg magától kitalálni, hogy milyen komponensek
# *lehetnek* fenn a rendszeren és azokat frissítse (StrictComponents
# no)?
# StrictComponents no
```

Ha ennél a beállításnál a `yes` értéket adjuk meg, akkor a `freebsd-update` feltételezni fogja, hogy a `Components` opcionál felsoroltunk minden frissítendő komponenst és nem próbál meg mást is megváltoztatni. Ilyenkor tehát a `freebsd-update` tulajdonképpen egyedül csak a `Components` által meghatározott elemekhez tartozó állományokat fogja frissíteni.

24.2.2. Biztonsági javítások

A biztonsági javítások mindig egy távoli gépen tárolódnak, a következő parancsok használatával tölthetők le és telepíthetők:

```
# freebsd-update fetch
# freebsd-update install
```

Amennyiben a rendszermagot is érintik javítások, úgy a rendszert a művelet befejeződésével újra kell indítanunk. Ha minden a megfelelő módon történt, akkor a rendszerünk már tartalmazni fogja a korábban letöltött és telepített javításokat, és a `freebsd-update` akár beállítható egy naponta végrehajtandó `cron(8)` feladatnak. Ehhez mindössze a következő bejegyzést kell elhelyeznünk az `/etc/crontab` állományban:

```
@daily                                root    freebsd-update cron
```

A bejegyzés szerint naponta egyszer le fog futni a `freebsd-update`. Ilyenkor, vagyis a `cron` paraméter megadásakor a `freebsd-update` csak ellenőrzi, hogy vannak-e telepítendő frissítések. Ha talál, akkor automatikusan letölti ezeket a lemezre, de nem telepíti. Helyette levélben értesíti a `root` felhasználót, aki ezután bármikor manuálisan kérheti a telepítést.

Probléma esetén az alábbi paranccsal megkérhetjük a `freebsd-update` programot a legutóbb telepített módosítások visszavonására:

```
# freebsd-update rollback
```

Ha ez a visszavonás a rendszermagra vagy annak moduljaira is vonatkozott, akkor a rendszert újra kell indítanunk a parancs futásának befejeződésével. A FreeBSD csak ilyenkor képes betölteni az új binárisokat betölteni a memóriába.

A `freebsd-update` önmagától csak a `GENERIC` típusú rendszermagokat képes frissíteni. Ha saját rendszermagot használunk, akkor azt a rendszer többi komponensének frissítését követően újra kell fordítanunk és telepítenünk. A `freebsd-update` azonban még akkor is érzekelni és frissíteni fogja a `GENERIC` rendszermagot (amennyiben az létezik), ha az éppen nem az aktuális(an futó) rendszermag.

Megjegyzés: Mindig érdemes tartani egy másolatot a `GENERIC` rendszermagról a `/boot/GENERIC` könyvtárban. Rengeteg különböző probléma felderítésében tud segíteni, illetve ez a 24.2.3 Szakasz szakaszban leírt `freebsd-update` programmal végzett frissítéseknél is hasznos lehet.

Hacsak nem változtatjuk meg az `/etc/freebsd-update.conf` állományt, a `freebsd-update` a rendszermag forrásait is frissíti a többivel együtt. A saját rendszermag újrafordítása és telepítése ezután a már a megszokott módon elvégezhető.

Megjegyzés: A `freebsd-update` által terjesztett frissítések nem mindig érintik a rendszermagot. Ha a rendszermag forrásai nem változnak egy `freebsd-update install` parancs kiadása során, akkor nem kötelező újrafordítani a saját rendszermagot. A `freebsd-update` viszont mindig módosítani fogja a `/usr/src/sys/conf/newvers.sh` állományt. Itt az aktuális hibajavítás sorszáma szerepel (amelyet a `-p` (mint "patch level" előtaggal kapcsolnak a rendszer verziójához, és a `uname -r` paranccsal lehet lekérdezni). Ennek megfelelően tehát a saját rendszermag újrafordítása után, még ha semmi más nem is változott, a `uname(1)` képes pontosan jelezni a rendszerhez készült hibajavítás sorszámát. Ez különösen fontos több rendszer karbantartása során, mivel így könnyen és gyorsan tájékozódhatunk azok naprakészségéről.

24.2.3. Váltás kisebb és nagyobb verziók között

Verziók közti váltás során a külső alkalmazások működését akadályozó régi tárgykódok és függvénykönyvtárak törlődni fognak. Ezért javasoljuk, hogy vagy töröljük le az összes portot és telepítsük újra, vagy az alaprendszer frissítése után hozzuk ezeket is naprakész állapotba a `ports-mgmt/portupgrade` segédprogram segítségével. Először minden bizonnyal szeretnék kipróbálni a frissítést, ezt a következő paranccsal tehetjük meg:

```
# portupgrade -af
```

Ezzel gondoskodunk róla, hogy a minden a megfelelően telepítődjön újra. Ha a `BATCH` környezeti változót a `yes` értékre állítjuk, akkor a folyamat során megjelenő összes kérdésre automatikusan a `yes` választ adjuk, ezáltal önállósítani tudjuk.

Ha saját rendszermagot használunk, akkor ennél valamivel azért több feladatunk van. Szükségünk lesz a `GENERIC` rendszermagot egy példányára, amelyet másoljunk a `/boot/GENERIC` könyvtárba. Amennyiben nincs `GENERIC` típusú rendszermag a rendszerünkön, a következő módok valamelyikén keresztül tudunk szerezni:

- Ha a saját rendszermagot még csak egyszer fordítottuk, akkor a `/boot/kernel.old` könyvtárban még megtalálható a `GENERIC`. Ezt nevezzük át egyszerűen `/boot/GENERIC` könyvtárra.
- Ha fizikailag hozzá tudunk férni az érintett géphez, akkor a `GENERIC` egy példányát akár CD-ről is átmásolhatjuk. Helyezzük be a telepítőlemezt és adjuk ki a következő parancsokat:

```
# mount /cdrom
# cd /cdrom/X.Y-RELEASE/kernels
# ./install.sh GENERIC
```

Itt a `X.Y-RELEASE` könyvtár nevében értelemsszerűen helyettesítsük be az általunk használt változatot. A `GENERIC` rendszermag ekkor alapértelmezés szerint a `/boot/GENERIC` könyvtárba kerül.

- Ha az előbbiek közül egyik sem lehetséges, akkor a `GENERIC` rendszermagot közvetlenül akár forrásból is lefordíthatjuk és telepíthetjük:

```
# cd /usr/src
# env DESTDIR=/boot/GENERIC make kernel
# mv /boot/GENERIC/boot/kernel/* /boot/GENERIC
# rm -rf /boot/GENERIC/boot
```

A `freebsd-update` akkor fogja ezt `GENERIC` rendszermagként felismerni, ha a hozzá tartozó konfigurációs állományt nem módosítjuk. Továbbá javasoljuk, hogy semmilyen speciális beállítást ne alkalmazzunk a fordítás során (érdemes üresen hagyni ehhez az `/etc/make.conf` állományt).

Nem kötelező újraindítani a rendszert a `GENERIC` rendszermaggal.

A `freebsd-update` képes frissíteni rendszerünket egy adott kiadásra. Például a következő paraméterek megadásával válthatunk a FreeBSD 6.4 használatára:

```
# freebsd-update -r 6.4-RELEASE upgrade
```

A parancs elindulása után nem sokkal, a váltáshoz szükséges információk összegyűjtéséhez a `freebsd-update` elemzi a konfigurációs állományában megadott beállításokat és a rendszer jelenleg használt verzióját. A képernyőn ekkor sorban megjelennek a program részéről érzékelt és nem érzékelt komponensek. Mint például ahogy itt látható:

```
Looking up update.FreeBSD.org mirrors... 1 mirrors found.
Fetching metadata signature for 6.3-RELEASE from update1.FreeBSD.org... done.
Fetching metadata index... done.
Inspecting system... done.
```

```
The following components of FreeBSD seem to be installed:
kernel/smp src/base src/bin src/contrib src/crypto src/etc src/games
src/gnu src/include src/krb5 src/lib src/libexec src/release src/rescue
src/sbin src/secure src/share src/sys src/tools src/ubin src/usbin
world/base world/info world/lib32 world/manpages
```

```
The following components of FreeBSD do not seem to be installed:
kernel/generic world/catpages world/dict world/doc world/games
world/proflibs
```

```
Does this look reasonable (y/n)? y
```

Ekkor a `freebsd-update` megpróbálja letölteni a verziók közti váltáshoz szükséges összes állományt. Bizonyos esetekben kérdésekkel fordul a felhasználó felé arra vonatkozóan, hogy miket telepítsen fel vagy mit csináljon.

A saját rendszermag használatakor az iménti lépés valamilyen ehhez hasonló figyelmeztetést fog adni:

```
WARNING: This system is running a "SAJÁT RENDSZERMAG" kernel, which is not a
kernel configuration distributed as part of FreeBSD 6.3-RELEASE.
This kernel will not be updated: you MUST update the kernel manually
before running "/usr/sbin/freebsd-update install"
```

Ez a figyelmeztetés most nyugodtan figyelmen kívül hagyható. A folyamat során a frissített GENERIC rendszermagot fogjuk használni.

A javítások letöltését követően megkezdődik a telepítésük. A váltás ezen lépése az adott gép aktuális terhelésétől és sebességétől függően változó hosszúságú lehet. Ezután a konfigurációs állományok összefésülése zajlik le — itt általában a emberi felügyeletre is szükség van az állományok összefésülésének irányításához, amelynek folyamatosan láthatóak az eredményei. A meghíúsult vagy kihagyott összefésülések a teljes frissítési folyamat leállítását vonják maguk után. Az `/etc` könyvtárban tárolt fontosabb állományokról, mint például a `master.passwd` vagy `group` javasolt előzetesen biztonsági mentést készíteni és később kézzel hozzájuk adni a változtatásaikat.

Megjegyzés: A rendszerben ekkor még nem lesz jelen semmilyen konkrét változás, az összes említett javítás és összefésülés egy külön könyvtárban történik. A telepített javításokat és az összefésült konfigurációs állományokat a folyamat végén magának a felhasználónak kell véglegesíteni.

A frissítési eljárás végén a következő parancs kiadásával tudjuk ténylegesen érvényesíteni az eddig elvégzett módosításokat:

```
# freebsd-update install
```

Először mindig a rendszermag és a hozzá tartozó modulok cserélődnek le. Ahogy ez végrehajtódott, újra kell indítanunk a rendszert. Ha saját rendszermagot használunk, akkor a `nextboot(8)` parancs segítségével állítsuk be a következő rendszerindítás során betöltendő rendszermagot a `/boot/GENERIC` könyvtárban levőre (ezt frissítettük):

```
# nextboot -k GENERIC
```

Figyelem Mielőtt újraindítanánk a gépünket a `GENERIC` rendszermaggal, győződjünk meg róla, hogy szerepel benne minden olyan meghajtó, amely elengedhetetlen a rendszer hiánytalan indításához (és képes lesz újra csatlakozni a hálózathoz, ha éppen távolról adminisztráljuk). Ez különösen olyan esetben fontos, amikor a saját rendszermagunkban beépítetten szerepeltek bizonyos modulok. Ilyenkor a `GENERIC` rendszermag használatakor ezeket a `/boot/loader.conf` állományon keresztül tölthetjük be ideiglenesen. A frissítés befejezéséig érdemes viszont minden nem létfontosságú szolgáltatást leállítani, leválasztani lemezeket és hálózati megosztásokat stb.

A rendszerünk most már újraindítható a frissített rendszermaggal:

```
# shutdown -r now
```

A rendszer sikeres újraindulása után ismét el kell indítanunk a `freebsd-update` programot, amely korábban már elmentette a frissítés állapotát, emiatt a legutóbbi pontról fog folytatódni, illetve törli az osztott könyvtárak és tárgykódok régebbi változatait. Innen az alábbi paranccsal léphetünk tovább:

```
# freebsd-update install
```

Megjegyzés: A függvénykönyvtárak verziói közti eltérések mértékétől függően elképzelhető, hogy a telepítés az említett három fázis helyett kettőben történik.

Most pedig újra kell fordítanunk vagy telepítenünk az összes általunk korábban használt külső alkalmazást. Erre azért van szükségünk, mert bizonyos alkalmazások a verziók közti váltás során törölt programkönyvtáraktól függtek. Ennek automatizálásában a `ports-mgmt/portupgrade` lesz segítségünkre. Az alkalmazások frissítésének elindításához a következő parancsokat használjuk:

```
# portupgrade -f ruby
# rm /var/db/pkg/pkgdb.db
# portupgrade -f ruby18-bdb
# rm /var/db/pkg/pkgdb.db /usr/ports/INDEX-*.db
# portupgrade -af
```

A parancsok lefutását követően a `freebsd-update` utolsó hívásával zárjuk le a frissítést. Ezzel a paranccsal tudunk tehát pontot tenni a frissítési procedúra végére:

```
# freebsd-update install
```

Ha a `GENERIC` rendszermagot csak átmenetileg használtuk, akkor most már a megszokott módon fordíthatunk és telepíthetünk magunk egy saját rendszermagot.

Indítsuk újra a rendszert a FreeBSD frissített változatával. A folyamat ezzel véget ért.

24.2.4. Rendszerek állapotainak összehasonlítása

A `freebsd-update` ragyogóan felhasználható a FreeBSD egy telepített változatának és egy általunk garantáltan megbízható példányának összevetésére. Ilyenkor a rendszerhez tartozó segédprogramokat, programkönyvtárakat és konfigurációs állományokat ellenőriztethetjük le. Az összehasonlítást ezzel a paranccsal kezdetjük meg:

```
# freebsd-update IDS >> eredmeny.idk
```

Figyelem Habár a parancs neve IDS (intrusion detection system), nem helyettesít semmilyen olyan behatolásjelző megoldást, mint amilyen például a `security/snort`. Mivel a `freebsd-update` adatokat tárol a lemezen, teljesen kézenfekvő a hamisítás lehetősége. Míg ennek eshetősége adott mértékben visszaszorítható a `kern.securelevel` csökkentésével és a `freebsd-update` által használt adatok írásvédett állományrendszerre helyezésével, erre a problémára az ideális megoldást mégis egy teljes biztonságban tudható referencia rendszer jelentheti. Ennek tárolására alkalmas lehet például egy DVD vagy egy külső USB-egység.

A parancs kiadása után megkezdődik a rendszer vizsgálata, és az ellenőrzés során folyamatosan jelennek meg az átvizsgált állományok a hozzájuk tartozó ismert és kiszámított sha256(1)-kódjukkal együtt. Mivel a képernyőn túlságosan gyorsan elúsznának az eredmények, ezért ezeket egy `eredmeny.idk` nevű állományba mentjük a későbbi elemzésekhez.

Az így keletkező állomány sorai ugyan meglehetősen hosszúak, de szerencsére viszonylag könnyen értelmezhetőek. Például az adott kiadásban szereplő állományoktól eltérőeket ezzel a paranccsal kérdezhetjük le:

```
# cat eredmeny.idk | awk '{ print $1 }' | more
/etc/master.passwd
```

```
/etc/motd
/etc/passwd
/etc/pf.conf
```

A példában most csak az első néhány állományt hagytuk meg, gyakran tapasztalhatunk viszont ennél többet. Ezek közül bizonyos állományok értelemszerűen eltérnek, mint itt például az `/etc/passwd`, mert időközben új felhasználókat adtunk a rendszerhez. Máskor egyéb állományok, például modulok nevei is felbukkanhatnak, mert tegyük fel, hogy a `freebsd-update` már frissítette ezeket. Ha ki szeretnénk zárni valamilyen állományokat vagy könyvtárakat az ellenőrzésből, egyszerűen csak soroljuk fel ezeket az `/etc/freebsd-update.conf` állományban megjelenő `IDIgnorePaths` beállításnál.

A korábban tárgyaltaktól függetlenül ez a rendszer alkalmas bonyolultabb frissítési folyamatok kíségetésére is.

24.3. A Portgyűjtemény frissítése a Portsnap használatával

Írta: Tom Rhodes. A megíráshoz felhasznált jegyzeteket készítette: Colin Percival.

A FreeBSD alaprendszer a Portgyűjtemény frissítéséhez is tartalmaz egy `portsnap(8)` elnevezésű segédprogramot. Ez a program elindítása után csatlakozik egy távoli géphez, ellenőrzi a biztonsági kulcsát és letölti a portok legfrissebb változatait. A biztonsági kulcs feladata a frissítés közben letöltött állományok sértetlenségének szavatolása, ezzel gondoskodik róla, hogy az adatok átvitelük közben nem változtak meg. A Portgyűjtemény legújabb változatát így érhetjük el:

```
# portsnap fetch
Looking up portsnap.FreeBSD.org mirrors... 3 mirrors found.
Fetching snapshot tag from portsnap1.FreeBSD.org... done.
Fetching snapshot metadata... done.
Updating from Wed Aug 6 18:00:22 EDT 2008 to Sat Aug 30 20:24:11 EDT 2008.
Fetching 3 metadata patches.. done.
Applying metadata patches... done.
Fetching 3 metadata files... done.
Fetching 90 patches.....10....20....30....40....50....60....70....80....90. done.
Applying patches... done.
Fetching 133 new ports or files... done.
```

A példában látható, hogy a `portsnap(8)` eltéréseket talált a helyi és a távoli rendszerekben fellelhető portok között, majd azokat ellenőrizte. Emellett az is megfigyelhető, hogy korábban már futtatuk a programot, mivel ha most indítottuk volna az első alkalommal, akkor egyszerűen letöltötte volna a teljes Portgyűjteményt.

Ahogy a `portsnap(8)` sikeresen befejezi az imént kiadott `fetch` művelet végrehajtását, a helyi rendszeren már telepítésre készen fognak várakozni a Portgyűjtemény és az hozzá tartozó ellenőrzött módosítások. A `portsnap` első használatakor az `extract` parancs segítségével telepíthetjük a frissített állományokat:

```
# portsnap extract
/usr/ports/.cvsignore
/usr/ports/CHANGES
/usr/ports/COPYRIGHT
/usr/ports/GIDs
/usr/ports/KNOBS
/usr/ports/LEGAL
/usr/ports/MOVED
```

```
/usr/ports/Makefile
/usr/ports/Mk/bsd.apache.mk
/usr/ports/Mk/bsd.autotools.mk
/usr/ports/Mk/bsd.cmake.mk
...
```

Egy korábban már telepített Portgyűjteményt a `portsnap update` paranccsal tudunk frissíteni:

```
# portsnap update
```

Ezzel lezárult a portok frissítése, inentől már az aktualizált Portgyűjtemény felhasználásával tetszőlegesen telepíthetők vagy frissíthetők az alkalmazások.

A `fetch`, `extract` vagy `update` műveletek egyetlen parancsba is összefűzhetők, ahogy ezt az alábbi példában is láthatjuk:

```
# portsnap fetch update
```

Ez a parancs letölti a Portgyűjtemény legfrissebb változatát, majd kitörmöríti azt a helyi `/usr/ports` könyvtárba.

24.4. A dokumentáció frissítése

Az alaprendszer és a Portgyűjtemény mellett a dokumentáció is a FreeBSD operációs rendszer szerves részét képezi. Noha a FreeBSD dokumentációjának legfrissebb változata folyamatosan elérhető a FreeBSD honlapjáról (<http://www.freebsd.org/doc>), egyes felhasználók ezt csak lassan vagy nem képesek folyamatosan elérni. Szerencsére egy helyi másolat megfelelő karbantartásával az egyes kiadásokhoz tartozó dokumentáció is frissíthető.

24.4.1. A dokumentáció frissítése CVSup használatával

A FreeBSD telepített dokumentációjának forrásai az alaprendszeréhez hasonlóan (lásd 24.7 Szakasz) a **CVSup** segítségével frissíthetők. Ebben a szakaszban megismerhetjük:

- hogyan telepítsük a dokumentáció előállításához szükséges eszközöket, amelyekkel a forrásokból újra tudjuk generálni a FreeBSD dokumentációját;
- hogyan töltsük le a dokumentáció forrását **CVSup** segítségével a `/usr/doc` könyvtárba;
- a dokumentáció előállításához alkalmazott rendszer milyen beállításokkal rendelkezik, vagyis hogyan korlátozzuk a generálást bizonyos nyelvekre vagy formátumokra.

24.4.2. A CVSup és a dokumentációs eszközök telepítése

Viszonylag sokféle eszközre lesz szükségünk, ha a FreeBSD dokumentációját a forrásokból akarjuk előállítani. Ezek az segédprogramok nem részei a FreeBSD alaprendszerének, mivel alapvetően nagyon sok helyet foglalnak el, és leginkább olyan FreeBSD felhasználók számára fontosak, akik folyamatosan a dokumentációval dolgoznak vagy gyakran frissítik azt forrásból.

A feladathoz szükséges összes eszköz elérhető a Portgyűjteményből. Ebben a FreeBSD Dokumentációs Projekt összeállított egy `textproc/docproj` nevű portot, amellyel az említett programok telepítését és frissítését igyekeztek megkönnyíteni.

Megjegyzés: Ha nem tartunk igényt a dokumentáció PostScript vagy PDF változatára, akkor ehelyett inkább érdemes megfontolnunk a `textproc/docproj-nojadetex` port telepítését. Ebben a változatban a **teTeX** betűszedő rendszer kivételével az összes segédprogram megtalálható. Mivel a **teTeX** önmagában nagyon sok segédeszköz telepítését jelenti, ezért amennyiben a PDF változat ténylegesen nem szükséges, érdemes eltekinteni a telepítésétől.

A **CVSup** telepítésével kapcsolatban pedig részletesebb információkat a CVSup használatával foglalkozó szakaszban olvashatunk.

24.4.3. A dokumentáció forrásának frissítése

A `/usr/share/examples/cvsup/doc-supfile` konfigurációs állomány segítségével a **CVSup** képes letölteni a dokumentáció forrásállományainak legfrissebb példányait. Itt a frissítést alapértelmezés szerint egy nem létező géptől fogjuk kérni (mivel ezt kötelező kitölteni), azonban a `cvsup(1)` programnak egy parancssori paraméter segítségével megadhatjuk melyik **CVSup** szerverről töltsse le a forrásokat:

```
# cvsup -h cvsup.FreeBSD.org -g -L 2 /usr/share/examples/cvsup/doc-supfile
```

Ne felejtjük el a `cvsup.FreeBSD.org` helyére beírni a hozzánk földrajzilag legközelebb elhelyezkedő **CVSup** szervert. Ezek teljes listáját a A.6.7 Szakasz tartalmazza.

Egy ideig eltarthat, amíg először letöltjük a forrásokat. Várjuk meg türelmesen, amíg befejeződik a művelet.

Később a forrásokat ugyanezzel a paranccsal tudjuk frissíteni. A **CVSup** ugyanis mindig csak a legutóbbi futtatása óta történt változásokat tölti le, ezért később már ez a lépés jelentősen felgyorsulhat.

A források letöltése után a dokumentációt például az ekkor keletkezett `/usr/doc` könyvtárban található `Makefile` használatával állíthatjuk elő. Tehát miután az `/etc/make.conf` állományban beállítottuk a `SUP_UPDATE`, `SUPHOST` és `DOCSUPFILE` változókat, le tudjuk futtatni a következő parancsot:

```
# cd /usr/doc
# make update
```

Az előbb említett `make(1)` változók jellemző értékei:

```
SUP_UPDATE= yes
SUPHOST?= cvsup.freebsd.org
DOCSUPFILE?= /usr/share/examples/cvsup/doc-supfile
```

Megjegyzés: Mivel a `SUPHOST` és a `DOCSUPFILE` változók értékét a `?=` szimbólummal állítottuk be, lehetőségünk van a parancssorból ezeknek más értékeket adni. Az `/etc/make.conf` állományba általában így érdemes felvenni a változókat, így nem kell minden alkalommal módosítani, amikor valamilyen új beállítást akarunk kipróbálni.

24.4.4. A dokumentáció különböző beállításai

A FreeBSD dokumentációjához tartozó, frissítést és előállítást végző rendszernek van néhány olyan beállítása, amelyekkel kérhetjük kizárólag csak a dokumentáció egyes részeinek frissítését vagy bizonyos kimeneti formátumok

használatát. Ezek vagy globálisan az `/etc/make.conf` állományban, vagy pedig a parancssorból, a `make(1)` program paramétereként adhatóak meg.

Ízelítőül néhány közülük:

DOC_LANG

Az előállítandó és telepítendő nyelvű dokumentáció felsorolása, tehát például csak az angol dokumentáció esetén ez `en_US.ISO8859-1`.

FORMATS

Az előállítandó dokumentáció kimeneti formátumainak felsorolása. Itt pillanatnyilag értékként a `html`, `html-split`, `txt`, `ps`, `pdf` és `rtf` jelenhet meg.

SUPHOST

A frissítéshez használt **CVSup** szerver hálózati neve.

DOCDIR

Az elkészült dokumentáció telepítésének helye. Ez alapértelmezés szerint a `/usr/share/doc`.

A folyamathoz kapcsolódóan további rendszerszintű `make(1)` változókról a `make.conf(5)` man oldalon olvashatunk.

A FreeBSD dokumentációjának előállításáért felelős rendszerben használható `make(1)` további változók bemutatásával kapcsolatban pedig olvassuk el az A FreeBSD Dokumentációs Projekt irányelvei kezdőknek (http://www.FreeBSD.org/doc/hu_HU.ISO8859-2/books/fdp-primer) című könyvet.

24.4.5. A FreeBSD dokumentációjának telepítése forrásból

Miután sikerült letöltenünk a `/usr/doc` könyvtárba a dokumentáció legfrissebb forrásait, készen állunk a rendszerünkön telepített példány frissítésére.

A `DOCLANG` értékeként megadott nyelven készült dokumentációkat a következő paranccsal tudjuk frissíteni:

```
# cd /usr/doc
# make install clean
```

Ha a `make.conf` állományban korábban már megadtuk a `DOCSUPFILE`, `SUPHOST` és `SUP_UPDATE` változók értékeit, akkor a telepítés fázisa könnyedén össze is vonatható a források frissítésével:

```
# cd /usr/doc
# make update install clean
```

Ha pedig csak bizonyos nyelvekhez tartozó dokumentációt szeretnénk frissíteni, akkor a `make(1)` akár a `/usr/doc` könyvtáron belül az egyes nyelvekhez tartozó alkönyvtárakon belül is meghívható, például:

```
# cd /usr/doc/en_US.ISO8859-1
# make update install clean
```

A dokumentáció formátumát a `FORMATS` változó felhasználásával tudjuk meghatározni:

```
# cd /usr/doc
# make FORMATS='html html-split' install clean
```

24.4.6. A dokumentációs portok használata

A szóbanforgó megoldást fejlesztette: Marc Fonvieille.

Ez előző szakaszban megmutattuk hogyan lehet a FreeBSD dokumentációját a források felhasználásával frissíteni. A források használatával végzett frissítés azonban nem minden FreeBSD rendszer esetében lehetséges vagy hatékony. Ha ugyanis a dokumentációs forrásból akarjuk előállítani, viszonylag sok eszköz és segédprogram, az ún. *dokumentációs eszközök* használatával kell tisztában lennünk, valamint bizonyos mértékig ismernünk kell a **CVS** használatát, tudunk kell kikérni a legfrissebb változatot és előállítattnunk belőle a végleges változatot. Ezért ebben a szakaszban most szót ejtünk egy olyan módszerről, ahol a FreeBSD dokumentációját a Portgyűjteményen keresztül tudjuk frissíteni, ezáltal:

- anélkül le tudjuk tölteni és telepíteni a dokumentáció adott pillanatban generált változatát, hogy a rendszerünkön bármi további teendőre szükség lenne (ennek köszönhetően nem kell telepítenünk a dokumentációs eszközöket);
- letölthetjük a dokumentáció forrását és a Portgyűjtemény eszközeivel előállíthatjuk belőle a megfelelő változatot (ez a források beszerzésében és feldolgozásában segít valamelyest).

A FreeBSD dokumentáció frissítésének fentebb említett módjait támogatják tehát a *dokumentációs portok*, amelyeket a Documentation Engineering Team <doceng@FreeBSD.org> havi rendszerességgel tart karban. Ezek a portok a FreeBSD Portgyűjteményén belül a docs (<http://www.freshports.org/docs/>) nevű virtuális kategóriában találhatóak meg.

24.4.6.1. A dokumentációs portok fordítása és telepítése

A dokumentáció könnyebb előállításához a dokumentációs portok a Portgyűjtemény lehetőségeit veszik igénybe. Segítségükkel automatikussá teszik a dokumentáció forrásának letöltését, a `make(1)` parancs meghívását a megfelelő környezetben, beállításokkal és parancssori paraméterekkel. Rajtuk keresztül a dokumentáció eltávolítása ugyanolyan egyszerűen megtehető, mint akármelyik másik FreeBSD port vagy csomag esetében.

Megjegyzés: Továbbá, amikor a dokumentációs portokat a saját rendszerünkön fordítjuk, a *dokumentációs eszközök* függőségként automatikusan települni fognak.

A dokumentációs portok a következő módon szerveződnek:

- Létezik egy ún. “főport”, a `misc/freebsd-doc-en`, ahol az összes fontosabb állomány megtalálható. Ez lényegében a dokumentációs portok közös őse. Alapértelmezés szerint kizárólag csak az angol nyelvű dokumentációt állítja elő.
- Létezik egy “mindenes port”, a `misc/freebsd-doc-all`, amely az összes elérhető nyelven és formátumban előállítja a dokumentációt.
- Végezetül minden nyelvhez létezik egy-egy “alport”, ilyen például a magyar dokumentáció esetén a `misc/freebsd-doc-hu` port. Mindegyikük a főporttól függ és az adott nyelvű dokumentációt telepítik.

Az eddigi összefoglaltaknak megfelelően a dokumentációs portokat forrásból a következő paranccsal lehet telepíteni (`root` felhasználóként):

```
# cd /usr/ports/misc/freebsd-doc-en
# make install clean
```

Ennek hatására előáll és telepítődik a `/usr/local/share/doc/freebsd` könyvtárba az angol nyelvű dokumentáció állományokra bontott HTML formátumban (hasnolán a <http://www.FreeBSD.org> tartalmához).

24.4.6.1.1. Gyakori beállítások

A dokumentációs portok alapértelmezett viselkedése több különböző opció segítségével is befolyásolható. Ezek közül most összefoglalunk néhányat:

WITH_HTML

Minden dokumentum egyetlen HTML állományba kerüljön. A végeredmény ekkor az adott dokumentum típusának megfelelően `article.html` (cikk) vagy `book.html` (könyv) néven keletkezik (képekkel együtt).

WITH_PDF

Minden dokumentum Adobe Portable Document Format típusú állományban jön létre. Ezek az állományok a **Ghostscript** vagy más egyéb PDF nézegetőkkel nyithatóak meg. Ekkor a dokumentáció konkrét típusától függően az állományok `article.pdf` (cikk) vagy `book.pdf` (könyv) néven állítódnak elő.

DOCBASE

A dokumentáció telepítésének helye. Alapértelmezés szerint ez a `/usr/local/share/doc/freebsd` könyvtár.

Megjegyzés: Ügyeljünk arra, hogy a telepítés alapértelmezett célkönyvtára eltér a **CVSup** módszerétől. Ugyanis mivel ilyenkor egy portot telepítünk, a tartalma alapértelmezés szerint a `/usr/local` könyvtáron belülre kerül. Ez azonban a `PREFIX` változó átállításával tetszőleges megváltoztatható.

Az előbbieket most egy rövid példán keresztül összefoglaljuk. A következő paranccsal tudjuk tehát a magyar nyelvű dokumentáció Portable Document Format változatát telepíteni:

```
# cd /usr/ports/misc/freebsd-doc-hu
# make -DWITH_PDF DOCBASE=share/doc/freebsd/hu install clean
```

24.4.6.2. A dokumentációs csomagok használata

A dokumentációs portok előző szakaszban bemutatott forrásból telepítésével kapcsolatban már említettük, hogy szükséges hozzá a dokumentációs eszközök telepítése, valamint némi szabad tárterület. Ha a dokumentációs eszközök telepítéséhez nem elengedőek a rendelkezésre álló erőforrásaink vagy a források feldolgozása túlságosan sokat foglalna a rendszerünkön, akkor lehetőségünk van a dokumentációs portok előre lefordított, csomagolt változatát használni.

A Documentation Engineering Team <doceng@FreeBSD.org> minden hónapban előkészíti a FreeBSD dokumentációs csomagok legfrissebb változatát. Az így karbantartott bináris csomagok azután tetszőlegesen használhatóak a szabványos csomagkezelő eszközökkel, mint amilyen például a `pkg_add(1)`, `pkg_delete(1)` és így tovább.

Megjegyzés: A bináris csomagok használata esetén a FreeBSD dokumentációja az adott nyelvhez az összes elérhető formátumban telepítésre kerül.

Például az alábbi paranccsal a magyar nyelvű dokumentációhoz tartozó legfrissebb bináris csomagot tudjuk telepíteni:

```
# pkg_add -r hu-freebsd-doc
```

Megjegyzés: A csomagok elnevezése eltér a hozzá tartozó port nevétől. Alakja a következő: `nyelv-freebsd-doc`, ahol a `nyelv` az adott nyelv rövid kódja, vagyis a magyar esetén a `hu`, illetve az egyszerűsített kínai esetén a `zh_ch`.

24.4.6.3. A dokumentációs portok frissítése

Az előzetesen telepített dokumentációs portok bármilyen portok frissítésére alkalmas eszközzel frissíthetők. Például a telepített magyar nyelvű dokumentáció a `ports-mgmt/portupgrade` eszközön keresztül így frissíthető csomagok használatával:

```
# portupgrade -PP hu-freebsd-doc
```

24.5. A fejlesztői ág követése

A FreeBSD-nek két fejlesztési ága van: a FreeBSD.current és a FreeBSD-STABLE. Ebben a szakaszban mindegyikükről mondunk pár szót, és megmutatjuk, miként lehet az adott ághoz igazítani a rendszerünk frissítését. Először a FreeBSD-CURRENT, majd a FreeBSD-STABLE változata kerül tárgyalásra.

24.5.1. A FreeBSD friss változatának használata

Ahogy arról már az imént is szó esett, nem szabad elfelejtenünk, hogy a FreeBSD-CURRENT a FreeBSD fejlesztésének “frontvonala”. Emiatt a FreeBSD-CURRENT használóinak szakmailag jólképzetteknek kell lenniük, és sosem szabad visszariadniuk a használat közben felmerülő rendszerszintű problémák önálló megoldásától. Ha korábban még nem foglalkoztunk FreeBSD-vel, kétszer is gondoljuk meg a telepítését!

24.5.1.1. Mi a FreeBSD-CURRENT?

A FreeBSD-CURRENT a FreeBSD mögött álló legfrissebb forráskódot képviseli. Itt találkozhatunk különféle olyan fejlesztés alatt álló részekkel, kísérletezésekkel és átmeneti megoldásokkal, amelyek nem feltétlenül kerülnek bele a szoftver következő hivatalos kiadásába. Noha a FreeBSD fejlesztői a FreeBSD-CURRENT forráskódját naponta fordítják, adódhatnak olyan időszakok, amikor a források mégsem használhatóak maradéktalanul. Az ilyen gondokat általában a lehető leggyorsabban igyekeznek megoldani, azonban attól függően, hogy éppen a forráskód melyik verzióját sikerült kifogni, a FreeBSD-CURRENT használata kész katasztrófa vagy akár a fejlődésben igazi továbblépés is lehet.

24.5.1.2. Kinek van szüksége a FreeBSD-CURRENT-re?

A FreeBSD-CURRENT használata elsősorban az alábbi 3 csoportot érinti:

1. A FreeBSD közösség azon tagjait, akik aktívan dolgoznak a forrásfa valamelyik részén, és mindazokat, akik számára a "legfrissebb" verzió használata feltétlen elvárás.
2. A FreeBSD közösség azon tagjait, akik aktívan tesztelnek, és a FreeBSD-CURRENT kordában tartásához hajlandók időt áldozni a menet közben felbukkanó problémák megoldására. Vannak olyanok is, akik a FreeBSD változásaival és fejlesztési irányával kapcsolatban kívánnak javaslatokat tenni, melyeket javítások és módosítások formájában tesznek közzé.
3. Mindazokat, akik pusztán kíváncsiak a fejlesztésben zajló eseményekre, vagy hivatkozási szándékkal töltik le a legfrissebb forrásokat (például csak *nézegetik*, de nem futtatják). Az ilyen emberek esetenként megjegyzéseket fűznek a fejlesztéshez vagy kódot küldenek be.

24.5.1.3. Mi *nem* a FreeBSD-CURRENT?

1. Az olyan kiadás előtt álló funkciók kipróbálásának egyszerű módja, amelyekről hallottunk, hogy milyen remek újításokat hoznak és mi akarunk lenni az elsők, akik ezt használni is fogják. Ne feledjük azonban, hogy amikor mindenki előtt kezdünk el használni egy újítást, mi leszünk egyben az elsők is, akik szembesülnek a benne rejlő hibákkal.
2. A gyors hibajavítások eszköze. A FreeBSD-CURRENT szinte bármelyik változata pontosan ugyanakkora valószínűséggel hoz magával új hibákat, mint ahogy eltünteti a régieket.
3. Akármilyen értelemben is "hivatalosan támogatott". Képességeinktől függően őszintén igyekszünk a lehető legtöbbet megtenni a 3 "törvényes" FreeBSD-CURRENT csoportba tartozó emberekért, azonban egyszerűen *nincs időnk* komolyabb segítségnyújtást adni. Ez viszont nem azt jelenti, hogy komisz és fukar emberek vagyunk, akik utálnak segíteni a másoknak (de máskülönben nem tudna fejlődni a FreeBSD). Csupán a FreeBSD fejlesztése *közben* fizikailag képtelenek vagyunk a naponta érkező ezernyi üzenetet rendre megválaszolni! A FreeBSD előremozdítása és a kísérleti stádiumban álló kóddal kapcsolatos kérdések megválaszolása közül a fejlesztők általában az elsőt részesítik előnyben.

24.5.1.4. A FreeBSD-CURRENT használata

1. Iratkozzunk fel az `freebsd-current` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) és `svn-src-head` (<http://lists.FreeBSD.org/mailman/listinfo/svn-src-head>) listákra. Ez nem egyszerűen hasznos, hanem *elengedhetetlen*. Ha nem vagyunk a `freebsd-current` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) listán, akkor nem fogjuk látni a rendszer aktuális állapotára vonatkozó megjegyzéseket, és így esetleg feleslegesen öljük az időnket olyan problémák megoldásába, amelyeket mások már korábban megoldottak. Ami viszont ennél is fontosabb, hogy így elszalasztjuk a rendszerünk folyamatos életbentartására vonatkozó létfontosságú bejelentéseket.

Az `svn-src-head` (<http://lists.FreeBSD.org/mailman/listinfo/svn-src-head>) listán láthatjuk az a forráskód egyes változtatásaihoz tartozó naplóbejegyzéseket, a hozzájuk tartozó esetleges mellékhatások ismertetésével együtt.

A listákra vagy a <http://lists.FreeBSD.org/mailman/listinfo> oldalon található többi lista valamelyikére úgy tudunk feliratkozni, ha rákattintunk a nevére. A további lépésekről ezt követően itt kapunk értesítést.

Amennyiben a teljes forrásfa változásai érdekelnek minket, javasoljuk az svn-src-all (<http://lists.FreeBSD.org/mailman/listinfo/svn-src-all>) lista olvasását.

2. A tükrözések egyikéről töltsük le a FreeBSD forrását. Erre két mód is kínálkozik:

a.

Használjuk a `cvsup` programot a `/usr/share/examples/cvsup` könyvtárban található `standard-supfile` állománnyal. Ez a leginkább ajánlott módszer, hiszen így csak egyszer kell letölteni az egész gyűjteményt, majd ezután már csak a változásokat. Sokan a `cvsup` parancsot a `cron` parancson keresztül adják ki, és ezzel mindig automatikusan frissítik a forrásaikat. A `cvsup` működését a fentebb említett minta `supfile` állomány megfelelő módosításával tudjuk a saját környezetünkhöz igazítani.

Megjegyzés: Az említett `standard-supfile` állomány eredetileg nem a FreeBSD-CURRENT, hanem inkább a FreeBSD biztonsági problémáit érintő javítások követésére használatos. A FreeBSD-CURRENT forrásainak eléréséhez a következő sort kell kicserélnünk ebben az állományban:

```
*default release=cvs tag=RELENG_X_Y
```

Erre:

```
*default release=cvs tag=.
```

A `tag` paramétereként megadható egyéb címkékről a kézikönyv CVS címkék szakaszában olvashatunk.

b.

Használjuk a **CTM** alkalmazás nyújtotta lehetőségeket. Amennyiben nagyon rossz netkapcsolattal rendelkezünk (drága vagy csak levelezésre használható) a **CTM** megoldást jelenthet számunkra. Legyünk azonban tekintettel arra, hogy helyenként zűrös lehet a használata és néha hibás állományokat gyárt. Emiatt viszont csak ritkán használják, így előfordulhat, hogy hosszabb ideig nem is működik. A 9600 bps vagy annál nagyobb sebességű kapcsolatok esetén ezért inkább a **CVSup** használatát javasoljuk.

3. Ha nem csak böngészésre, hanem fordításra is szedjük a forrásokat, mindig töltsük le a FreeBSD-CURRENT *egészét*, ne csak egyes részeit. Ez azzal magyarázandó, hogy a forráskód bizonyos részei más helyeken található részekről is függenek, és ezért az önálló fordításuk szinte garantáltan gondot fog okozni.

A FreeBSD-CURRENT lefordítása előtt figyelmesen olvassuk át a `/usr/src` könyvtárban található `Makefile` állományt. A frissítési folyamat részeként először mindenképpen érdemes telepíteni egy új rendszermagot és újrafordítani az alaprendszert. Olvassuk el a FreeBSD-CURRENT levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) üzeneteit és a `/usr/src/UPDATING` állományt, ahol megtalálhatjuk az ezzel kapcsolatos legújabb információkat, melyek egy-egy újabb kiadás közeledtével egyre fontosabbá válnak.

4. Foglalkozzunk vele! Ha már a FreeBSD-CURRENT változatát használjuk, ne legyünk restek véleményt formálni róla, különösen abban az esetben, ha továbbfejlesztésekről vagy hibákra van szó. Leginkább a forráskóddal együtt érkező javaslatoknak szoktak örülni a fejlesztők!

24.5.2. A FreeBSD stabil változatának használata

24.5.2.1. Mi a FreeBSD-STABLE?

A FreeBSD-STABLE az a fejlesztési ág, ahonnan az egyes kiadások származnak. Ebbe az ágba már más ütemben kerülnek a változások, mivel általánosan elfogadott, hogy ide a korábban már kipróbált módosítások vándorolnak át a FreeBSD-CURRENT ágból. Ez azonban *még mindig* csak egy fejlesztési ág, ami arra utal, hogy a FreeBSD-STABLE által adott pillanatban képviselt források nem feltétlenül felelnek meg bizonyos célokra. Ez csupán egy újabb fejlesztési nyomvonal, nem pedig a végfelhasználók kenyere.

24.5.2.2. Kinek van szüksége a FreeBSD-STABLE-re?

Ha szeretnénk figyelemmel kísérni vagy valamilyen módon kiegészíteni a FreeBSD fejlesztési folyamatát, különösen a FreeBSD következő "nagyobb" kiadását illetően, akkor érdemes követnünk a FreeBSD-STABLE forrásait.

Habár a FreeBSD-STABLE ágba is bekerülnek a biztonsági jellegű javítások, ettől még nem kell feltétlenül ezt követnünk. A FreeBSD-hez kiadott biztonsági figyelmeztetések mindig leírják, hogyan kell javítani a hibát az érintett kiadásokban ¹, azonban az egész fejlesztési ágot felesleges csak biztonsági okból kifolyólag követni, mivel így olyan változások is kerülhetnek a rendszerbe, amire nincs szükségünk.

Habár igyekszünk gondoskodni a FreeBSD-STABLE ágban található források lefordíthatóságáról és működőképességéről, nem minden esetben szavatolható. Ráadásul mivel a FreeBSD-STABLE ágba kerülő kódokat először a FreeBSD-CURRENT ágban fejlesztik ki, és mivel a FreeBSD-STABLE felhasználói többen vannak a FreeBSD-CURRENT változaténál, ezért szinte elkerülhetetlen, hogy ilyenkor a FreeBSD-STABLE változatban bizonyos hibák és szélsőséges esetek be ne következzenek, amelyek a FreeBSD-CURRENT használata során még nem buktak ki.

Ezért a FreeBSD-STABLE ág vakon követését senkinek *sem* ajánljuk, és különösen fontos, hogy éles szervereken előzetes kimerítő tesztek nélkül ne futassunk FreeBSD-STABLE rendszert.

Ha ehhez nem rendelkezünk elegendő erőforrással, akkor egyszerűen használjuk a FreeBSD legfrissebb kiadását, és az egyes kiadások között pedig bináris frissítéssel közlekedjünk.

24.5.2.3. A FreeBSD-STABLE használata

1. Iratkozzunk fel a [freebsd-stable](http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable) (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable>) listára. Ezen keresztül értesülhetünk a FreeBSD-STABLE használata során felmerülő fordítási függőségekről vagy más, külön figyelmet igénylő problémákról. Gyakran ezen a levelezési listán elmélkednek a fejlesztők a vitatott javításokról vagy frissítésekről, amibe a felhasználók is beleszólhatnak, ha a szóbanforgó változtatással kapcsolatban bármilyen problémájuk vagy ötletünk van.

Iratkozzunk fel a követni kívánt ághoz tartozó **SVN** levelezési listára. Például ha a 7-STABLE ág változásait követjük, akkor az [svn-src-stable-7](http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-7) (<http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-7>) listára érdemes feliratkoznunk. Ennek segítségével elolvashatjuk az egyes változtatásokhoz tartozó naplóbejegyzéseket, a rájuk vonatkozó esetleges mellékhatások ismertetésével együtt.

Ezekre, valamint a <http://lists.FreeBSD.org/mailman/listinfo> címen elérhető listák valamelyikére úgy tudunk feliratkozni, ha a nevükre kattintunk. A további teendők ezután itt jelennek meg.

2. Amennyiben egy új rendszert akarunk telepíteni és a FreeBSD-STABLE havonta készült pillanatképeit akarjuk rajta futtatni, akkor erről bővebb felvilágosítást a Pillanatképek (<http://www.FreeBSD.org/snapshots/>) honlapján

találhatunk (angolul). Emellett a legfrissebb FreeBSD-STABLE kiadást telepíthetjük a tükrözések valamelyikéről is, majd innen a lentebb található utasítások szerint tudunk hozzáférni a FreeBSD-STABLE forráskódjának legfrissebb változatához.

Ha már fut a gépünkön a FreeBSD egy korábbi kiadása, és ezt akarjuk forráson keresztül frissíteni, akkor ezt a FreeBSD tükrözéseivel könnyedén megtehetjük. Két módon is:

a.

Használjuk a `cvsup` programot a `/usr/share/examples/cvsup` könyvtárból származó `stable-supfile` állománnyal. Ez a leginkább ajánlott módszer, mivel így csak egyszer kell letölteni a teljes gyűjteményt, utána már csak a hozzá tartozó változtatásokra van szükségünk. A `cvsup` parancsot sokan a `cron` segítségével futtatják, és ezzel automatikusan frissülnek a forrásainak. A `cvsup` működését környezetünkhöz az előbb említett minta `supfile` megfelelő módosításával tudjuk behangolni.

b.

Használjuk a **CTM** programot. Ha nincs olcsó vagy gyors internetkapcsolatunk, akkor érdemes ezt a módszert választani.

3. Alapvetően azonban ha gyorsan szeretnénk hozzájutni a forrásokhoz és a sávszélesség nem meghatározó tényező, akkor helyette válasszuk a `cvsup` vagy az `ftp` használatát, és csak minden más esetben **CTM**-et.

4.

Mielőtt lefordítanánk a FreeBSD-STABLE változatát, figyelmesen olvassuk át a `/usr/src` könyvtárban levő `Makefile` állományt. Az átállási folyamat részeként először minden bizonnyal telepítenünk kell egy új rendszermagot és újra kell fordítanunk az alaprendszert. A FreeBSD-STABLE levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable>) valamint a `/usr/src/UPDATING` elolvasásából értesülhetünk azokról az egyéb, gyakran nagyon fontos változásokról, melyek elengedhetetlenek lesznek a következő kiadás használatához.

24.6. A forrás szinkronizálása

Az internet (vagy elektronikus levelek) használatán keresztül számos mód kínálkozik az FreeBSD Projekthez tartozó források frissen tartásához egy adott, vagy éppen az összes területen attól függően, hogy mik érdekelnek minket. Ehhez elsősorban az Anonim CVS, CVSup és CTM szolgáltatásokat ajánljuk fel.

Figyelem Habár lehetséges csupán a forrásfa egyes részeit letölteni, a támogatott frissítési eljárás során azonban szükségünk lesz az egész fa szinkronizálására és a rendszerhez tartozó felhasználói programok (vagyis minden olyan program, amely a felhasználói térben fut, ilyeneket találhatunk többek közt a `/bin` és `/sbin` könyvtárakban) valamint rendszermag újrafordítására is. Ha csak a felhasználói programok forrásait, vagy csak a rendszermagot, esetleg csupán a forrásfa egyes részeit frissítjük, akkor az gondokat okozhat. Az itt előforduló problémák fordítási hibáktól kezdve rendszerösszeomlásokon keresztül akár adatvesztésbe is torkollhatnak.

Az **Anonim CVS** és a **CVSup** alkalmazások ún. *lehúzással* frissítik a forrásokat. A **CVSup** használatakor a felhasználó (vagy a `cron` szkript) meghívja a `cvsup` programot, amely az állományok aktualizálásához felveszi a kapcsolatot egy máshol megtalálható `cvsupd` szerverrel. Az így nyert frissítések az adott pillanatig visszemenőleg

érkeznek meg, de csak akkor, ha igényeljük ezeket. A frissítést könnyedén le tudjuk szabályozni a számunkra érdekes egyes állományokra és könyvtárakra. A frissítéseket a szerver hozza létre menet közben annak megfelelően, hogy milyen verziókkal rendelkezünk, és mihez akarunk szinkronizálni. Az **Anonim CVS** a **CVSup**nál valamivel egyszerűbb abban a tekintetben, hogy ez a **CVS**-nek egy olyan kiterjesztése, amely lehetővé teszi a változtatások közvetlen lehúzását egy távoli CVS tárházból. Miközben a **CVSup** mindezt sokkal hatékonyabb valósítja meg, addig az **Anonim CVS** jóval könnyebben használható.

Velük szemben a **CTM** nem hasonlítja össze interaktívan a saját és a központi szerveren tárolt forrásokat és nem is húzza át ezeket. Ehelyett egy olyan szkriptől van szó, amely naponta többször megvizsgálja a központi **CTM** szerveren tárolt állományok a legutóbbi futtatás óta keletkezett változtatásait, majd az észlelt módosulásokat betömöríti, felcímkézi egy sorozatszámmal és (nyomtatható ASCII formátumban) előkészíti ezeket az e-mailen keresztüli küldésre. Az így létrehozott "CTM delták" megérkezésük után a `ctm_rmail(1)` segédprogrammal kerülnek feldolgozásra, amely magától visszaalakítja, ellenőrzi és alkalmazza a változtatásokat a forrásfa felhasználó birtokában levő másolatára. Ez a megoldás hatékonyabb a **CVSup** használatánál, mert kisebb terhelést jelent a szerverek számára, hiszen a frissítéshez nem a *lehúzást*, hanem a *küldést* alkalmazzák.

Természetesen minden említett eljárásnak megvannak a maga kompromisszumai. Ha véletlenül kitöröljük a forrásfánk egyes részeit, a **CVSup** képes ezt észrevenni és helyreállítani a sérült részeket. A **CTM** ezzel szemben ezt nem végzi el, szóval ha (biztonsági mentés nélkül) letöröljük a forrásainkat, akkor az egész szinkronizálást az elejéről kell kezdenünk (pontosabban a legfrissebb CVS-es "alapdeltától") és a **CTM**-mel újraépíteni az egészet, esetleg a **Anonim CVS**-sel letörölni a hibás adatokat és újraszinkronizálni.

24.7. Az alaprendszer újrarendezése

Miután sikerült a helyi forrásfánkat a FreeBSD egy nekünk szimpatikus (FreeBSD-STABLE, FreeBSD-CURRENT és így tovább) változatához igazítanunk, elérkezett az idő, hogy a segítségével újraparancsuk az egész rendszert.

Készítsünk biztonsági mentéstNem tudjuk eléggé nyomatékosítani, hogy *mielőtt* nekikezdenénk, készítsünk egy biztonsági mentést a rendszerünkről. Míg az alaprendszer újraparancsuk nem túlságosan bonyolult feladat (egészen addig, amíg a megadott utasításokat követjük), saját magunk vagy mások hibájából fakadóan kialakulhatnak olyan helyzetek, amikor a rendszer nem lesz képes elindulni.

Mindenképpen győződjünk meg róla, hogy tisztességesen elvégeztük a mentést és akad a kezünk ügyében egy javításra felhasználható rendszerindító floppy vagy CD. Valószínűleg soha nem lesz ténylegesen szükségünk rájuk, azonban jobb félni, mint megijedni!

Iratkozzunk fel a megfelelő levelezési listákraA FreeBSD-STABLE és FreeBSD-CURRENT ágak természetüknél fogva *fejlesztés alatt állnak*. A FreeBSD fejlesztését is emberek végzik, ezért előfordulhatnak benne tévedések.

Ezek a tévedések gyakran csak ártalmatlan apróságok, amelyek hatására kapunk például egy ismeretlen diagnosztikai hibát. De ezzel szemben létrejöhetnek pusztító erejű hibák is, amelyek hatására a rendszerünk nem lesz képes elindulni, károsodnak az állományrendszerek (vagy még rosszabb).

Ha ilyen történik, akkor egy "felszólítást" (egy "heads up" témájú üzenetet) küldenek az érintett változatokhoz tartozó listákra, amelyben igyekeznek kifejteni a probléma természetét és a rendszerre mért hatását. Miután "minden rendbejött", a probléma megoldásáról is küldenek egy értesítést.

Ha a FreeBSD-STABLE levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable>) vagy a FreeBSD-CURRENT levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) olvasása nélkül

próbáljuk meg használni a FreeBSD-STABLE és FreeBSD-CURRENT verziókat, akkor csak magunknak keressük a bajt.

Ne használjuk a `make world` parancsot Rengeteg régebben készült dokumentáció erre a feladatra a `make world` parancs kiadását javasolja. Ennek használatával azonban átlépünk olyan fontos lépéseket, amelyek valójában csak akkor lennének kihagyhatóak, ha pontosan tudjuk mit csinálunk. Ezért az esetek döntő többségében nem a `make world` használatára van szükségünk, hanem a most bemutatandó eljárásra.

24.7.1. A rendszer frissítése dióhéjban

A frissítés megkezdése előtt érdemes elolvasnunk a `/usr/src/UPDATING` állományt, ahol a letöltött források használatához elvégzendő előzetes intézkedésekről kaphatunk hírt. Ezután kövessük az alábbiakban körvonalazott módszer egyes lépéseit.

Ezek a lépések feltételezik, hogy egy korábbi FreeBSD verziót használunk, tehát a fordító, a rendszermag, az alaprendszer és a konfigurációs állományok valamelyik régebbi változatát. Alaprendszer alatt, amelyet sokszor csak a “world” néven hivatkozunk, a rendszer számára alapvető fontosságú binárisokat, programkönyvtárakat és programfejlesztéshez szükséges egyéb állományokat értjük. Maga a fordítóprogram is része ennek, azonban tartalmaz néhány speciális megszorítást.

Mindezek mellett továbbá feltételezzük, hogy előzetesen már valamilyen módon letöltöttük a friss forrásokat. Ha rendszerünkön ezt még nem tettük volna meg, akkor a 24.6 Szakasz segítségével tájékozódhatunk részletesen arról, hogyan tölthetjük le a legfrissebb verziót.

A rendszer forráskódon keresztüli frissítése egy kicsivel körülményesebb, mint amennyire elsőre látszik. A FreeBSD fejlesztők az évek során fontosnak találták, hogy a folyamatosan felszínre bukkanó, elkerülhetetlen függőségek tükrében meglehetősen drámai módon megváltoztassák az erre javasolt módszert. Ezért a szakasz további részében a pillanatnyilag javasolt frissítési megoldás nyomán fogunk haladni.

A sikeres frissítések során az alábbi akadályokkal kell mindenképpen szembenéznünk:

- A fordító régebbi változata nem feltétlenül lesz képes lefordítani az új rendszermagot. (Illetve a régebbi fordítóprogramok tartalmazhatnak hibákat.) Ezért az új rendszermagot már a fordító új változatával kell előállítanunk. Ebből következik, hogy az új rendszermag elkészítéséhez először a fordítóprogram újabb változatát kell lefordítanunk. Ez viszont nem feltétlenül jelenti azt, hogy az új rendszermag fordítása előtt az új fordítóprogramot *telepítenünk* is kellene.
- Az új alaprendszer esetenként bizonyos új funkciókat igényelhet a rendszermagtól. Ezért a frissebb alaprendszer telepítése előtt telepítenünk kell a frissebb rendszermagot.
- Ez az előbb említett két akadály képzi az okát a következő bekezdésekben bemutatott `buildworld`, `buildkernel`, `installkernel`, `installworld` sorozatnak. Természetesen léteznek további egyéb indokok is, amiért még érdemes az itt leírtak szerint frissíteni a rendszerünket. Ezek közül most vegyünk néhány kevésbé nyilvánvalóbbat:
- A régebbi alaprendszer nem minden esetben fog problémamentesen együttműködni az új rendszermaggal, ezért az alaprendszer újabb változatát szinte azonnal az új rendszermagot követően kell telepítenünk.

- Vannak olyan konfigurációs változtatások, amelyeket még az új alaprendszer telepítése előtt el kell végeznünk, a többi viszont veszélyes lehet a korábbi alaprendszerre. Ezért a konfigurációs állományokat általában két külön lépésben kell frissíteni.
- A frissítés során nagyrészt csak állományok cserélődnek el és újabbak érkeznek, a korábbiak nem törlődnek. Ez bizonyos esetekben azonban gondokat okozhat. Ennek eredményeképpen a frissítés során időnként előfordulhat, hogy magunknak kell manuálisan némely megadott állományokat törölnünk. Elképzelhető, hogy ezt a jövőben még majd automatizálni fogják.

Ezek a megfontolások vezettek tehát az ismertetendő eljárás kialakításához. Ettől függetlenül adódhatnak olyan helyzetek, amikor további lépéseket is be kell iktatnunk, viszont az itt bemutatott folyamat egy ideje már viszonylag elfogadottnak tekinthető:

1. `make buildworld`

Először lefordítja az új fordítóprogramot és néhány hozzá tartozó eszközt, majd ennek felhasználásával elkészíti az alaprendszer többi részét. Az eredmény a `/usr/obj` könyvtárban keletkezik.

2. `make buildkernel`

Eltérően a `config(8)` és `make(1)` programok korábban javasolt alkalmazásától, ezzel a paranccsal már a `/usr/obj` könyvtárban létrehozott új fordítót használjuk. Ez védelmet nyújt a fordító és rendszermag változatai közti eltérésekből fakadó problémák ellen.

3. `make installkernel`

Telepíti a lemezre az új rendszermagot és a hozzá tartozó modulokat, ezáltal lehetővé válik a frissített rendszermag betöltése.

4. Átváltás egyfelhasználós módba.

Egyfelhasználós módban a minimálisra csökkenthetjük a futó szoftverek frissítéséből adódó bonyodalmakat. Ezzel együtt minimálissá válik a régi alaprendszer és az új rendszermag eltéréseiből eredő problémák előfordulása is.

5. `mergemaster -p`

Az új alaprendszer telepítéséhez elvégzi a konfigurációs állományok részéről szükséges frissítéseket. Például felvesz még nem létező csoportokat vagy felhasználókat. Ez gyakran elengedhetetlennek bizonyulhat, mivel ha a rendszer legutóbbi frissítése óta újabb csoportok vagy felhasználók kerültek be az alaprendszerbe, a `installworld` csak akkor tud hibamentesen lefutni, ha ezek már a futásakor is elérhetőek.

6. `make installworld`

Átmásolja a `/usr/obj` könyvtárból a korábban elkészített új alaprendszert. Lefutása után már mind az új rendszermag és az új alaprendszer a megfelelő helyén található.

7. `mergemaster`

Feldolgozzuk a korábbi fázisból fennmaradó konfigurációs állományok frissítését, mivel most már elérhető az új alaprendszer.

8. A rendszer újraindítása.

Az új rendszermag és az új konfigurációs állományokkal futó alaprendszer használatához teljesen újra kell indítanunk a számítógépünket.

Ha a FreeBSD ugyanazon fejlesztési ágán belül frissítjük a rendszerünket, például a 7.0 kiadásról a 7.1 kiadásra, akkor értelemszerűen nem kell az iménti eljárás minden lépését szorosan követni, hiszen nagyon valószínűtlen,

hogy komoly eltérések lennének a fordítóprogram, a rendszermag, az alaprendszer és a konfigurációs állományok között. Ilyenkor akár nyugodtan kiadhatjuk a `make world` parancsot, majd kérhetjük a rendszermag fordítását és telepítését.

A fejlesztési ágak közti váltás során azonban könnyen érhetnek minket meglepetések, ha nem a megadottak szerint járunk el.

Egyes váltásokhoz (például 4.x és 5.0 között) további lépések megtétele is szükséges lehet (például adott állományok törlése vagy átnevezése még az `installworld` előtt). Ilyenkor mindig figyelmesen olvassuk át a `/usr/src/UPDATING` állományt, különös tekintettel a végére, mivel gyakran ott adják meg a konkrét verzióváltáshoz szükséges teendőket.

A szakaszban összefoglalt lépések egyfajta evolúciós folyamat eredményei, melynek során a fejlesztők felismerték, hogy nem tökéletesen kivédeni az összes frissítéssel járó problémát. A javasolt eljárás remélhetőleg viszont még sokáig érvényes marad.

Megjegyzés: A FreeBSD 3.x vagy annál is korábbi változatok frissítése még ennél is több ügyességet kíván. Ha ilyen verziót akarunk frissíteni, akkor feltétlenül olvassuk el az `UPDATING` állományt!

Röviden tehát a FreeBSD forráskódon keresztüli frissítését így foglalhatjuk össze:

```
# cd /usr/src
# make buildworld
# make buildkernel
# make installkernel
# shutdown -r now
```

Megjegyzés: Néhány ritka esetben a `buildworld` lépés előtt szükségünk lehet a `mergemaster -p` parancs lefuttatására is. Erről az `UPDATING` állományból tudakozódhatunk. Általában azonban nyugodt szívvel kihagyhatjuk ezt a lépést, kivéve, ha nem egy vagy több főbb FreeBSD változatot átívelő frissítést végzünk.

Miután az `installkernel` sikeresen befejezte a munkáját, indítsuk újra a számítógépet egyfelhasználós módban (a betöltő parancssorában adjuk ki `boot -s` parancsot). Itt futtassuk a következőket:

```
# adjkerntz -i
# mount -a -t ufs
# mergemaster -p
# cd /usr/src
# make installworld
# mergemaster
# reboot
```

Olvassuk el a magyarázatokat Az iménti leírt folyamat csupán rövid összefoglalás, amivel némi gyorstalpalást igyekeztünk adni. Az egyes lépések megértéséhez azonban javasolt átolvasni a most következő szakaszokat is, különösen abban az esetben, ha saját rendszermagot akarunk használni.

24.7.2. Nézzük meg a `/usr/src/UPDATING` állományt

Mielőtt bármihez is nekifognánk, keressük meg a `/usr/src/UPDATING` (vagy hasonló, a forráskód másolatunk tényleges helyétől függő) állományt. Ebben adják hírül az esetlegesen felmerülő problémákra vonatkozó fontosabb információkat, vagy határozzák meg az egyes lefuttatandó parancsok pontos sorrendjét. Amennyiben az `UPDATING` ellentmondana az itt olvasottaknak, az `UPDATING` tartalma a mérvadó.

Fontos: A korábban tárgyaltak szerint az `UPDATING` elolvasása nem helyettesíti a megfelelő levelezési listák figyelemmel kísérését. Ez a két elvárás nem kizárja, hanem kiegészíti egymást.

24.7.3. Ellenőrizzük az `/etc/make.conf` állományt

Vizsgáljuk át a `/usr/share/examples/etc/make.conf` és az `/etc/make.conf` állományokat. Az előbbi tartalmaz néhány alapértelmezett beállítást – ezek javarészt megjegyzésbe rakták. Ha használni akarjuk a rendszer lefordítása során, tegyük bele ezeket az `/etc/make.conf` állományba. Ne felejtjük el azonban, hogy minden, amit megadunk az `/etc/make.conf` állományba, a `make` minden egyes elindításakor felhasználásra kerül. Éppen ezért olyanokat érdemes itt beállítani, amik az egész rendszerünket érintik.

A legtöbb felhasználó számára az `/etc/make.conf` állományhoz a `/usr/share/examples/etc/make.conf` állományban található `CFLAGS` és `NO_PROFILE` sorokra lesz szüksége, melyeket kivehetünk a megjegyzésből.

A többi definíció (`COPTFLAGS`, `NOPORTDOCS` és így tovább) használatáról már mindenki maga dönt.

24.7.4. Frissítsük az `/etc` tartalmát

Az `/etc` könyvtár tartalmazza a rendszer beállításaival kapcsolatos információk jelentős részét, valamint a rendszer indítása során lefutó szkripteket. Egyes szkriptek a FreeBSD verzióiról verzióira változnak.

Némely konfigurációs állományok a rendszer hétköznapi működésében is szerepet játszanak. Ilyen például az `/etc/group`.

Alkalmanként a `make installworld` parancs futása során igényt tart adott nevű felhasználókra és csoportokra. A frissítéskor azonban ezek a felhasználók vagy csoportok nem feltétlenül állnak rendelkezésre, ami gondokat okozhat. Ezért bizonyos esetekben a `make buildworld` előzetesen ellenőrzi az igényelt felhasználók és csoportok meglétét.

Erre például szolgálhat a `smmsp` felhasználó esete. Nélküle a felhasználók nem tudták telepíteni az új rendszert, mert hiányában az `mtree(8)` nem volt képes létrehozni a `/var/spool/clientmqueue` könyvtárat.

Ezt úgy lehetett megoldani, hogy még az alaprendszer lefordítása (a `buildworld`) előtt meg kellett hívni a `mergemaster(8)` parancsot a `-p` paraméterrel. Így csak azokat az állományokat fogja összehasonlítani, amelyek feltétlenül szükségesek a `buildworld` vagy az `installworld` sikeres működéséhez. Amennyiben a `mergemaster` egy olyan verziójával rendelkezünk, amely nem ismeri a `-p` paramétert, akkor az első indításakor használjuk a forrásfáiban található újabb verzióját:

```
# cd /usr/src/usr.sbin/mergemaster
# ./mergemaster.sh -p
```

Tipp: Ha különösen paranoiásak vagyunk, akkor a csoport törlése vagy átnevezése előtt az alábbi paranccsal ellenőrizni tudjuk az általa birtokolt állományokat:

```
# find / -group GID -print
```

Ez megmutatja *GID* (mely megadható numerikus vagy név formájában is) jelzésű csoporthoz tartozó összes állományt a rendszerünkben.

24.7.5. Váltunk egyfelhasználós módba

A rendszert egyfelhasználós módban érdemes lefordítani. A nyilvánvalóan érezhető gyorsaság előnyei mellett azért is jobban járunk, mert az új rendszer telepítése során számos rendszerszintű állomány is módosításra kerül, beleértve a szabványos rendszerszintű binárisokat, függvénykönyvtárakat, include állományokat és így tovább. Ha üzemelő rendszeren végezzük el mindezen változtatásokat (különösen amikor rajtunk kívül még további felhasználók is tartózkodnak a rendszerben), az csak a bajt hozza ránk.

Másik lehetőség gyanánt a rendszert magát lefordíthatjuk többfelhasználós módban is, majd ezután csak a telepítést hajtjuk végre egyfelhasználós üzemmódban. Ha eszerint cselekszünk, egyszerűen várjunk addig, amíg az összes fordítás be nem fejeződik, és az egyfelhasználósra váltást halasszuk a *installkernel* vagy *installworld* idejére.

Egy működő rendszerben rendszeradminisztrátorként az alábbi parancs kiadásával válthatunk át egyfelhasználós módba:

```
# shutdown now
```

Ezt elérhetjük úgy is, ha újraindítjuk a rendszert és a rendszer indításakor a “single user” pontot választjuk a menüből. Ekkor a rendszer egyfelhasználós módban indul el. Miután ez megtörtént, adjuk ki a következő parancsokat:

```
# fsck -p
# mount -u /
# mount -a -t ufs
# swapon -a
```

Ezekkel a parancsokkal először ellenőrizzük az állományrendszereket, ezután újracsatlakoztatjuk a / állományrendszert írható módban, csatlakoztatjuk az */etc/fstab* állományban megadott összes többi UFS típusú állományrendszert, majd bekapcsoljuk a lapozóállomány használatát.

Megjegyzés: Ha a gépünk óráját nem a greenwich-i, hanem a helyi idő szerint állítottuk be (ez akkor áll fenn, ha a *date(1)* parancs nem a helyes időt és időzónát jelzi ki), akkor még erre is szükségünk lehet:

```
# adjkerntz -i
```

Ezzel a helyi időzóna beállításait tudjuk jól beállítani — nélküle később még gondjaink akadhatnak.

24.7.6. Töröljük a */usr/obj* könyvtárat

A rendszer egyes részei fordításuk során a */usr/obj* könyvtáron belülre kerülnek (alapértelmezés szerint). Az itt található könyvtárak a */usr/src* könyvtárszerkezetét követik.

Ha mindenestől töröljük ezt a könyvtárat, akkor növeli tudjuk a `make buildworld` folyamat sebességét és megmenekülünk néhány függőségekkel kapcsolatos fejfájástól is.

Egyes `/usr/obj` könyvtáron belüli állományoknál szerepelhet a “megváltoztathatatlan” (immutable) állományjelző (lásd `chflags(1)`), amelyet a művelet elvégzéséhez először el kell távolítanunk.

```
# cd /usr/obj
# chflags -R noschg *
# rm -rf *
```

24.7.7. Fordítsuk újra az alaprendszert

24.7.7.1. A kimenet elmentése

Jól járunk azzal, ha a `make(1)` futásának kimenetét elmentjük egy állományba, mivel így a hibák esetén lesz egy másolatunk a hibaüzenetről. Ha konkrétan nekünk nem is feltétlenül segít megtalálni a hiba tényleges okát, mások viszont többet tudnak róla mondani, ha beküldjük ezt a FreeBSD egyik levelezési listájára.

Ezt egyébként a legegyszerűbben a `script(1)` parancs segítségével oldhatjuk meg, amelynek paraméteréül azt az állományt kell megadni, ahova menteni akarjuk a kimenetet. Ezt közvetlenül a rendszer újrafordítása előtt kell kiadnunk, majd miután megállt, a `exit` paranccsal kiléphetünk belőle.

```
# script /var/tmp/mw.out
Script started, output file is /var/tmp/mw.out
# make TARGET
... fordít, fordít, fordít ...
# exit
Script done, ...
```

Ilyenkor *soha ne* a `/tmp` könyvtárba mentsük a kimenetet, mert ennek a tartalma a következő indítás során magától törlődik. Sokkal jobban tesszük, ha a `/var/tmp` könyvtárba (ahogy tettük azt az előbbi példában is) vagy a `root` felhasználó könyvtárába mentünk.

24.7.7.2. Az alaprendszer fordítása

A `/usr/src` könyvtárban kell állnunk:

```
# cd /usr/src
```

(kivéve természetesen, ha máshol van a forráskód, akkor abba a könyvtárba menjünk).

Az alaprendszert a `make(1)` paranccsal fordíthatjuk újra. Ez a `Makefile` nevű állományból olvassa be a FreeBSD programjainak újrafordítását leíró utasításokat, a fordításuk sorrendjét és így tovább.

A begépelendő parancsor általános alakja tehát a következőképpen néz ki:

```
# make -x -DVÁLTOZÓ target
```

A fenti példában a `-x` egy olyan a paraméter, amelyet a `make(1)` programnak adunk át. A `make(1)` man oldalán megtalálhatjuk az összes neki átadható ilyen beállítást.

A *-DVÁLTOZÓ* alakú paraméterek közvetlenül a *Makefile* állománynak adnak át olyan változókat, amelyek segítségével vezérelhető a viselkedése. Ezek ugyanazok a változók, mint amelyek az */etc/make.conf* állományban is szerepelnek, és itt a beállításuk egy másik módját kapjuk. Így a

```
# make -DNO_PROFILE target
```

paranccsal is megadhatjuk, hogy ne profilozott függőkönyvtárak jöjjenek létre, ami pontosan megfelel a

```
NO_PROFILE=      true      #      Avoid compiling profiled libraries
```

sornak az */etc/make.conf* állományban.

A *target* árulja el a *make(1)* programnak, hogy mi a teendője. Minden egyes *Makefile* különböző “targeteket” definiál, és a kiválasztott *target* mondja meg, pontosan mi is fog történni.

Egyes *target*ek ugyan megjelennek a *Makefile* állományban, azonban nem feltétlenül hivatkozhatunk rájuk közvetlenül. Ehelyett csupán arra valók, hogy a fordítás folyamatának lépéseit felbontsák még kisebb allépésekre.

A legtöbb esetben azonban semmilyen paramétert nem kell átadnunk a *make(1)* parancsnak, ezért a teljes formája így fog kinézni:

```
# make target
```

ahol a *target* az egyik fordítási lehetőséget képviseli. Az első ilyen *target*nek mindig a *buildworld*-nek kell lennie.

Ahogy a neve is mutatja, a *buildworld* lefordítja az összes forrást a */usr/obj* könyvtárba, majd a *installworld* mint másik *target*, telepíti az így létrehozott elemeket a számítógépre.

A *target*ek szétválasztása két okból is előnyös. Először is lehetővé teszi, hogy az új rendszert biztonságban lefordíthassuk, miközben az a jelenleg futó rendszert nem zavarja. A rendszer tehát képes “saját magát újrafordítani”. Emiatt a *buildworld* *target* akár többfelhasználós módban is mindenféle nem kívánatos hatás nélkül használható. Ennek ellenére azonban továbbra is azt javasoljuk, hogy a *installworld* részt egyfelhasználós módban futtassuk le.

Másodrészt ezzel lehetőségünk nyílik NFS állományrendszer alkalmazásával több számítógépre is telepíteni hálózaton keresztül. Ha például három frissítendő számítógépünk van, az A, B és C, akkor az A gépen először adjuk ki a *make buildworld*, majd a *make installworld* parancsot. A B és C gépek ezután NFS segítségével csatlakoztatják az A */usr/src* és */usr/obj* könyvtárait, amelyet követően a *make installworld* paranccsal telepíteni tudjuk a fordítás eredményét a B és C gépekre.

Noha a *world* mint *target* még mindig létezik, használata határozottan ellenjavalt.

A

```
# make buildworld
```

parancs kiadásakor a *make* parancsnak megadható egy *-j* paraméter is, amellyel párhuzamosíthatjuk a folyamat egyes részeit. Ez általában többprocesszoros számítógépeken nyer értelmet, azonban mivel a fordítás folyamatának haladását inkább az állományműveletek mintsem a processzor sebessége korlátozza, ezért alkalmazható akár egyprocesszoros gépeken is.

Tehát egy átlagos egyprocesszoros gépen így adható ki a parancs:

```
# make -j4 buildworld
```

Ennek hatására `make(1)` egyszerre 4 szálon igyekszik működni. A levelezési listákra beküldött tapasztalati jellegű bizonyítékok azt igazolják, hogy általában ez a beállítás adja a legjobb teljesítményt.

Ha többprocesszoros géppel rendelkezünk és rajta SMP támogatású rendszermagot indítottunk el, akkor érdemes 6 és 10 közötti értékekkel kísérleteznünk.

24.7.7.3. Időigény

Számos tényező befolyásolja a fordítás tényleges időbeli hosszát, de a FreeBSD-STABLE fa lefordítása mindenféle trükkök és rövidítések nélkül a legtöbb számítógépen olyan egy vagy két órára taksálható. A FreeBSD-CURRENT fához ennél valamivel több időre lesz szükségünk.

24.7.8. Fordítsunk és telepítsünk egy új rendszermagot

Az újdonsült rendszerünket csak akkor tudjuk igazán kihasználni, ha egy új rendszermagot is készítünk hozzá. Ez gyakorlati szinten tulajdonképpen elvárás, mivel könnyen előfordulhat, hogy bizonyos memóriabeli adatszerkezetek felépítése megváltozott, ezért némely programok, mint például a `ps(1)` és `top(1)`, egészen addig nem lesznek képesek normálisan működni, amíg a rendszer és a rendszermag forráskódja nem illeszkedik egymáshoz.

Ennek legegyszerűbb és egyben legbiztonságosabb módja, ha a `GENERIC` beállításai alapján gyártunk és telepítünk egy rendszermagot. Még ha a `GENERIC` beállításai nem is tartalmazzák a rendszerünkben fellelhető összes eszközt, minden megtalálható bennük ahhoz, hogy a rendszert sikeresen elindíthassuk legalább egyfelhasználós módban. Ez mellesleg remek próbája az új rendszer életképességének. Miután elindítottuk a rendszert a `GENERIC` típusú rendszermaggal és meggyőződöttünk róla, hogy a rendszer tényleg működőképes, a megszokott rendszermagunk konfigurációs állománya alapján nyugodtan elkészíthetjük ezután azt is.

FreeBSD alatt egy új rendszermag építése előtt fontos újrafordítani az alaprendszert.

Megjegyzés: Ha saját beállításaink szerint akarunk rendszermagot létrehozni és már van is ehhez egy konfigurációs állományunk, akkor erre használhatjuk a `KERNCONF=SAJÁTMAG` paramétert is, valahogy így:

```
# cd /usr/src
# make buildkernel KERNCONF=SAJÁTMAG
# make installkernel KERNCONF=SAJÁTMAG
```

Hozzátennénk, hogy ha a `kern.securelevel` rendszerváltozó értékét 1 felé állítottuk és a rendszermag állományának beállítottunk `noschg` vagy hozzá hasonló állományjelzőt, akkor az `installkernel` lefuttatásához mindenképpen egyfelhasználós módba kell váltanunk. Minden más esetben további bonyodalmak nélkül ki tudjuk adni az említett parancsokat. A `kern.securelevel` részleteiről az `init(8)` oldalán, a különböző állományjelzőkről pedig a `chflags(1)` oldalán olvashatunk.

24.7.9. Indítsuk újra a rendszert egyfelhasználós módban

Az új rendszermag működésének leteszteléséhez indítsuk újra a rendszert egyfelhasználós módban. Ennek pontos részleteit lásd 24.7.5 Szakasz.

24.7.10. Telepítsük az új rendszer binárisait

Ha a FreeBSD friss változatát nemrég fordítottuk le a `make buildworld` paranccsal, akkor utána az `installworld` segítségével tudjuk telepíteni a keletkezett programokat.

Tehát írjuk be ezeket:

```
# cd /usr/src
# make installworld
```

Megjegyzés: Amennyiben a parancssorban a `make buildworld` használata során adtunk meg változókat, akkor ne felejtjük el ugyanazokat megadni a `make installworld` kiadása során sem. Ez viszont a többi paraméterre már nem feltétlenül érvényes. Például a `-j` beállítást szigorúan tilos az `installworld` targettel együtt használni.

Ennek megfelelően tehát ha korábban ezt írtuk be:

```
# make -DNO_PROFILE buildworld
```

akkor így telepítsünk:

```
# make -DNO_PROFILE installworld
```

Máskülönben azokat a profilozott függvénykönyvtárakat próbáljuk meg telepíteni, amelyek a `make buildworld` futása során nem jöttek létre.

24.7.11. Frissítsük a `make installworld` által kihagyott állományokat

Az alaprendszer újrafordítása nem regisztrálja az új vagy megváltozott állományokat bizonyos könyvtárakban (különösen értendő ez az `/etc`, `/var` és `/usr` esetén).

Az ilyen állományokat a legegyszerűbben a `mergemaster(8)` használatával tarthatjuk karban, de igény szerint akár kézzel is elvégezhetjük a szükséges aktualizálásokat. Függetlenül attól, hogy mit is választunk, mindenképpen készítsünk biztonsági mentést az `/etc` könyvtárról arra az esetre, ha bármilyen szörnyűség történne.

24.7.11.1. A `mergemaster`

Írta: Tom Rhodes.

A `mergemaster(8)` segédprogram valójában egy Bourne szkript, amely segít az `/etc` könyvtárunkban és a forrásfáiban levő `/usr/src/etc` könyvtárban elhelyezkedő konfigurációs állományok közti eltérések megállapításában. Ezt a módszert ajánljuk arra, hogy összevessük a konfigurációs állományainkat a forrásfáiban található változataikkal.

A használatának megkezdéséhez egyszerűen írjuk be, hogy `mergemaster`, majd várjunk egy kicsit, amíg a `mergemaster` létrehoz magának egy átmeneti környezetet a `/` könyvtárból elindulva és megtölti azt a különböző rendszerszintű beállításokat tartalmazó állományokkal. Ezeket az állományokat aztán összehasonlítja a jelenleg érvényben levő változataikkal. Ilyenkor a köztük talált eltéréseket a `diff(1)` formátumának megfelelően módon mutatja meg, ahol a `+` jelöli a hozzáadott vagy módosított sorokat, a `-` pedig a teljesen eltávolítandó vagy cserélendő sorokat. Erről a formátumról bővebben a `diff(1)` man oldalán találhatunk felvilágosítást.

A `mergemaster(8)` ezt követően megmutatja az összes olyan állományt, ahol eltérést tapasztalt, és ezen a ponton van lehetőségünk letörölni (delete) az új állományokat (amelyekre itt most ideiglenes állományként hivatkozik),

telepíteni (install) a módosíthatatlan ideiglenes (új) állományt, valamint összefésülni (merge) az ideiglenes (új) és a jelenlegi állományokat, vagy ismét átnézni (view) a diff(1) által jelzett különbségeket.

Ha az ideiglenes állomány törlését választjuk, akkor a mergemaster(8) ezt úgy értelmezi, hogy változatlanul meg akarjuk tartani a jelenlegi változatot és törölni az újat. Ezt alapvetően nem javasoljuk, hacsak tényleg nem látunk valamilyen okot erre. A mergemaster(8) parancssorában a `?` begépelésével bármikor kérhetünk segítséget. Ha az állomány kihagyását (skip) választjuk, akkor majd ismét felajánlja, amikor végeztünk az összes többivel.

A módosíthatatlan ideiglenes állomány telepítésének választásával lecseréljük a jelenleg verziót az újra. Ha az aktuális verziót sem változtattuk meg, akkor számunkra ez a legjobb megoldás.

Az állományok összefésülésének kiválasztásakor kapunk egy szövegszerkesztőt, benne a két állomány tartalmával. Ilyenkor tudjuk a képernyőn soronként egyeztetni a két állományt, majd a belőlük a megfelelő részek összeválogatásával kialakítani az eredményt. Ebben a feldolgozási módban az `l` (mint left, vagyis bal) billentyű lenyomására a bal oldalon látható részt, az `r` (mint right, vagyis jobb) lenyomására pedig a jobb oldalon látható részt választjuk ki. Az így keletkező eredményt ezután egy állományba kerül, amelyet telepíteni tudunk. Ez a megoldás olyan állományok esetében használható, amikor a felhasználó módosított az alapértelmezett beállításokat.

Ha a diff(1) szerinti alakban akarjuk átnézni a különbségeket, akkor a mergemaster(8) ugyanúgy megmutatja ezeket, mint a parancssor megjelenítése előtt.

Miután a mergemaster(8) végigment a rendszerszintű állományokon, további opciókat mutat. Megkérdezheti, hogy újra létre akarjuk-e hozni a jelszavakat tároló állományt (rebuild), illetve a folyamat végén a megmaradt ideiglenes állományok törlésére (remove) vár választ.

24.7.11.2. Az állományok aktualizálása kézzel

Ha inkább manuálisan szeretnénk frissíteni, akkor nem másolhatjuk csak egyszerűen át az állományokat a `/usr/src/etc` könyvtárból a `/etc` könyvtárba és nem hagyhatjuk ezeket sorsukra. Egyes állományokat először “telepíteni” kell. Ez azért van így, mert a `/usr/src/etc` könyvtár *nem puszta*n az `/etc` könyvtár egyszerű másolata. Ráadásul az `/etc` könyvtárban vannak olyan állományok, amelyek a `/usr/src/etc` könyvtárban nem is találhatóak meg.

Ha (az ajánlottak szerint) a mergemaster(8) segítségével dolgozunk, nyugodtan átléphetünk a következő szakaszra.

Saját magunk a legegyszerűbben ezt úgy tudjuk megoldani, ha telepítjük az állományokat egy új könyvtárba és ezután nekiállunk változásokat keresni.

Az `/etc` meglevő tartalmának mentése Habár elméletileg magától semmi sem fogja bántani ezt a könyvtárat, azért ettől függetlenül mindig érdemes biztosra menni. Ezért másoljuk az `/etc` könyvtár tartalmát egy megbízható helyre. Például:

```
# cp -Rp /etc /etc.old
```

Az `-R` itt a rekurzív másolást jelenti, a `-p` pedig a dátumok, az állományok és egyéb tulajdoni viszonyainak megőrzését.

Az `/etc` új változatának telepítéséhez szükségünk lesz még további könyvtárakra is. Erre a feladatra a `/var/tmp/root` tökéletesen megfelel, ahol még létre kell hoznunk néhány alkönyvtárat.

```
# mkdir /var/tmp/root
# cd /usr/src/etc
```

```
# make DESTDIR=/var/tmp/root distrib-dirs distribution
```

Ezzel létrejön a szükséges könyvtárszerkezet és települnek az állományok. Sok üres alkönyvtár is keletkezik a /var/tmp/root könyvtáron belül, ezeket töröljük. Ezt a legkönnyebben így tehetjük meg:

```
# cd /var/tmp/root
# find -d . -type d | xargs rmdir 2>/dev/null
```

Ezzel törlődnek az üres könyvtárak. (A szabvány hibakimenetet átirányítottuk a /dev/null eszközre, és ezzel elnyomtuk a nem üres könyvtárak esetén keletkező hibüzeneteket.)

A /var/tmp/root most már tartalmazza az összes olyan állományt, amelyek normális esetben a / könyvtáron belül foglalnak helyet. Ezt követően nincs más dolgunk, csak végigmenni az itt található állományokon és megállapítani, miben térnek a meglévőektől.

Vegyük észre, hogy a /var/tmp/root könyvtárba telepített állományok némelyikének neve “.”-tal kezdődik. Az írás pillanatában ezek csak a /var/tmp/root/ és /var/tmp/root/root/ könyvtárakban található parancsértelmezőhöz tartozó indító állományok lehetnek, habár adódhatnak még ilyenek (attól függően, mikor olvassuk ezt). Ezért a feldolgozásukhoz ne felejtsük el a `ls -a` parancsot használni.

A `diff(1)` alkalmazásával legegyszerűbben így tudunk összehasonlítani két állományt:

```
# diff /etc/shells /var/tmp/root/etc/shells
```

Ennek hatására megjelennek az /etc/shells és az új /var/tmp/root/etc/shells állományok közti különbségek. A segítségével gyorsan el tudjuk dönteni, hogy összefésüljük-e a két állományt, vagy csak egyszerűen írjuk felül a régebbi verziót az újjal.

Az új könyvtár (/var/tmp/root) nevébe írjuk bele a dátumot is, így könnyedén össze tudunk hasonlítani több verziót is: A rendszer gyakori újrafordítása az /etc szintén gyakori aktualizálását is maga után vonja, ami viszont fárasztó lehet.

Az iménti folyamatot fel tudjuk gyorsítani, hogy ha az /etc legutoljára összefésült változatát megtartjuk. A most következő eljárás ennek mikéntjét vázolja fel.

1. A megszokottak szerint fordítsuk le a rendszert. Majd amikor az /etc könyvtárat és a többi is frissíteni akarjuk, a célként megadott könyvtár nevében adjuk meg a dátumot. Ha tehát például 1998. február 14. van, akkor írjuk ezt:

```
# mkdir /var/tmp/root-19980214
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root-19980214 \
    distrib-dirs distribution
```

2. Fésüljük össze a könyvtárban található az állományokat a fentiekben körvonalmazottak szerint.

Befejezés után *őrizzzük meg* a /var/tmp/root-19980214 könyvtárat.

3. Mikor újra letöltjük a legfrissebb forrásokat és megismételjük az előbbi lépéseket, haladjunk megint az első lépés szerint. Ekkor tehát létrejön egy újabb könyvtár, amelynek a neve ezúttal már /var/tmp/root-19980221 lesz (ha például hetente frissítünk).
4. Most már meg tudjuk vizsgálni a közbeeső héten született eltéréseket, ha a két könyvtárra kiadunk egy rekurzív `diff(1)` hívást:

```
# cd /var/tmp
# diff -r root-19980214 root-19980221
```

Általában így kevesebb eltérést kapunk, mint amennyi például a `/var/tmp/root-19980221/etc/` és az `/etc` összehasonlítása során elkerült volna. Mivel kisebb a keletkezett különbségek száma, ezért könnyebb lesz átvinnünk az `/etc` könyvtárunkba is a módosításokat.

5. Ezután törölhetjük a régebbi `/var/tmp/root-*` könyvtárat:

```
# rm -rf /var/tmp/root-19980214
```

6. Az `/etc` összefésülésékor mindig ismételjük meg ezeket a lépéseket.

A `date(1)` meghívásával akár automatikussá is tehetjük a könyvtárak névadását:

```
# mkdir /var/tmp/root-`date +%Y%m%d`
```

24.7.12. Újraindítás

Ezzel készen is vagyunk. Miután ellenőriztük, hogy minden a megfelelő helyére került, indítsuk újra a rendszert. Ehhez egy egyszerű `shutdown(8)` is elegendő:

```
# shutdown -r now
```

24.7.13. Befejeztük!

Gratulálunk, sikerült frissítenünk a FreeBSD rendszerünket.

Ha mégis valami balul ütne ki, könnyen újra tudjuk fordítani a rendszer egyes részeit. Például, ha véletlenül letöröltük az `/etc/magic` állományt az `/etc` frissítése vagy összefésülése során, a `file(1)` parancs nem fog tudni rendesen működni. Ilyenkor a következőket kell tennünk a hiba kijavításához:

```
# cd /usr/src/usr.bin/file
# make all install
```

24.7.14. Kérdések

1. Minden egyes változtatásnál újra kell fordítani a rendszert?

Nem könnyű választ adni erre a kérdésre, mivel ez alapvetően a változtatás jellegétől függ. Például, ha elindítjuk a **CVSup** programot és csak az alábbi állományok frissülnek:

```
src/games/cribbage/instr.c
src/games/sail/pl_main.c
src/release/sysinstall/config.c
src/release/sysinstall/media.c
src/share/mk/bsd.port.mk
```

Ekkor valószínűleg nem éri meg újrafordítani a teljes rendszert. Elegendő csupán belépni az érintett állományokat tartalmazó alkönyvtárakba és ott rendre kiadni a `make all install` parancsot. Ha viszont már valami komolyabb,

például az `src/lib/libc/stdlib` változott meg, akkor vagy az egész rendszert, vagy legalább azon részeit fordítsuk újra, amely statikusan linkelték (és minden más időközben még hozzáadott statikusan linkelt dolgot).

Hogy melyik megoldást választjuk, teljesen rajtunk áll. Újrafordíthatjuk az egész rendszert kéthetente, mondván, hadd gyűljenek fel szépen a módosítások, vagy a függőségek pontos kielemezésével csak azokat az elemeket fordítjuk újra, amelyek tényleg meg is változtak.

Természetesen az egész attól függ, hogy milyen gyakran és melyik rendszert, a FreeBSD-STABLE-t vagy a FreeBSD-CURRENT-et frissítjük.

2. A fordító rengeteg 11-es jelzést (signal 11) (vagy másfajta jelzéseket) dob hibával. Mi történhetett?

Ez általában hardveres meghibásodásra utal. A rendszer újrafordítása alapjaiban véve egy remek módszer számítógépünk alkatrészeinek terhelésére, ezért gyakorta előhozza a memória már meglevő hibáit. Ezek többnyire abban fogalmazódnak meg, hogy a fordító rejtélyes módon leáll mindenféle furcsa jelzések hatására.

Erről biztosan úgy tudunk meggyőződni, ha újraindítjuk a `make` programot és az a folyamat egy teljesen másik pontján vész el.

Ilyenkor nem tudunk mást tenni, mint egymás után kicserélgetjük, kivesszük az alkatrészeket és így próbáljuk megállapítani, pontosan melyikük is okozza a gondokat.

3. A fordítása befejezése után törölhetem a `/usr/obj` könyvtárat?

Röviden: Igen.

A `/usr/obj` tartalmazza a fordítás folyamata során keletkező összes tárgykódot. Ennek törlése általában a `make buildworld` első lépései között szerepel. Ezért tulajdonképpen a `/usr/obj` megtartásának nincs túlságosan sok értelme, viszont elég sok (jelenleg úgy kb. 340 MB) helyet fel tudunk így szabadítani.

Ha azonban értjük a dolgunkat, akkor megadhatjuk a `make buildworld` parancsnak, hogy hagyja ki ezt a lépést. Ennek hatására a fordítás sokkal hamarabb véget ér, mivel a legtöbb forrást így nem kell újrafordítani. Öröm az örömben, hogy ha netalán aprócska függőségi problémák merülnének fel, akkor az egész fordítás megfeneklik mindenféle különös módokon. Emiatt gyakran írnak feleslegesen leveleket a FreeBSD levelezési listáira, melyek a rendszer sikertelen újrafordításáról panaszkodnak, miközben kiderül, hogy az maguk az érintettek akarták lerövidíteni a folyamatot.

4. Lehetséges a megszakadt fordítás folytatása?

Ez attól függ, hogy a probléma bekövetkezése előtt mennyire sikerült eljutni a fordításban.

Általában (tehát nem feltétlenül minden esetben) a `make buildworld` lefordítja a fordításhoz szükséges eszközök (például a `gcc(1)` és `make(1)`) újabb változatait és a rendszer függvénykönyvtárait, majd ezeket telepíti. Ezután ezekkel az új eszközökkel lefordíttatja saját magukat és ismét telepíti. Ezt követően fordítja újra az új rendszerállományokkal az egész rendszert (így ezúttal már az olyan szokásos felhasználói programokat is, mint például az `ls(1)` és a `grep(1)`).

Ha tudjuk, hogy az utolsó fázisban álltunk le (mivel megnéztük a fordításhoz tartozó kimenetet), akkor (minden további nélkül) elég ennyi:

... kijavítjuk a hibát ...

```
# cd /usr/src
# make -DNO_CLEAN all
```

Ezzel megmarad a korábbi `make buildworld` munkájának eredménye.

Ha ezt az üzenetet látjuk a `make buildworld` kimenetében:

```
-----
Building everything..
-----
```

akkor különösebb gond nélkül megcsinálhatjuk.

Amennyiben viszont nem látunk ilyen üzenetet, vagy nem vagyunk benne biztosak, akkor még mindig jobb elővigyázatosnak lenni, ezért kénytelenek leszünk teljesen előlről kezdeni a fordítást.

5. Hogyan tudjuk felgyorsítani a fordítást?

- Futtassuk egyfelhasználós módban.
- Tegyük a `/usr/src` és `/usr/obj` könyvtárakat külön állományrendszerekre, külön lemezekre. Sőt, ha lehetséges, akkor ezeket a lemezeket tegyük külön lemezvezérlőkre.
- Még mindig jobb, ha ezeket az állományrendszereket a `ccd(4)` (lemezek összefűzését vezérlő meghajtó) segítségével kiterjesztjük több lemezes eszközre.
- Kapcsoljuk ki a profilozást (az `/etc/make.conf` állományban a `"NO_PROFILE=true"` megadásával). Többnyire úgy sem lesz rá szükségünk.
- Az `/etc/make.conf` állományban a `CFLAGS` változót állítsuk az `-O -pipe` értékre. Az `-O2` gyakran sokkal lassabb, az `-O` és `-O2` alig tér el az optimalizálás mértékében. A `-pipe` paraméter hatására pedig a fordítóprogram átmeneti állományok helyett csöveket használ a kommunikációra, és így megtakarít némi lemezhasználatot (a memóriahasználat terhére).
- Ha a `make(1)` parancsnak átadjuk a `-jn` paramétert, akkor képes több mindent párhuzamosan futtatni. Ez sok esetben segít attól függetlenül, hogy egy- vagy többprocesszoros gépünk van.
- A `/usr/src` könyvtárat tartalmazó állományrendszert csatlakoztathatjuk (vagy újracsatlakoztathatjuk) a `noatime` beállítással. Ilyenkor az állományrendszer nem rögzíti a hozzáférés idejét. Erre az információra sincs igazából szükségünk.

```
# mount -u -o noatime /usr/src
```

FigyelemA fenti példa azt feltételezi, hogy a `/usr/src` könyvtárnak saját állományrendszere van. Ha ez nem így lenne (tehát például a `/usr` része), akkor itt azt kell megadnunk, nem pedig a `/usr/src` nevét.

- A `/usr/obj` könyvtárat tartalmazó állományrendszert csatlakoztathatjuk (vagy újracsatlakoztathatjuk) az `async` beállítással. Ennek hatására a lemez írása aszinkron módon történik. Magyarul az írási műveletek azonnal befejeződnek, miközben az adat ténylegesen csak pár másodperccel később kerül ki a lemezre. Ezzel az írási kérések gyönyörűen összegyűjthetők, ami nagymértékű növekedést eredményez a teljesítményben.

FigyelemNe felejtjük el azonban, hogy ezzel együtt az állományrendszerünk is sérülékenyebbé válik. Ezen beállítás használatával megnő annak az esélye, hogy egy áramkimaradást követő indításnál az állományrendszer helyreállíthatatlan állapotba kerül.

Ha egyedül csak a `/usr/obj` található ezen az állományrendszeren, akkor ez nem jelent akkora veszélyt. Amikor viszont rajta kívül még értékes adat is található az állományrendszeren, a beállítás érvényesítése előtt mindenképpen készítsünk róla friss mentéseket.

```
# mount -u -o async /usr/obj
```

FigyelemAhogy arról az előbb is szó esett, ha a `/usr/obj` nem egy különálló állományrendszeren található, akkor a példában szereplő csatlakozási pontot cseréljük ki a megfelelőre.

6. Mi tegyünk, ha valami nem megy rendesen?

Egyértelműen bizonyosodjunk meg róla, hogy a korábbi fordításokból nem maradtak vissza semmiféle kóbor állományok. Ennyi sokszor pontosan elég.

```
# chflags -R noschg /usr/obj/usr
# rm -rf /usr/obj/usr
# cd /usr/src
# make cleandir
# make cleandir
```

Igen, a `make cleandir` parancsot tényleg kétszer kell kiadni.

Ezután a `make buildworld` parancstól indulva kezdjük újra a fordítást.

Ha még ezek után is fennáll a probléma, küldjük el a hibát tartalmazó kimenetet és a `uname -a` parancs eredményét a FreeBSD general questions levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>) címére.

Ne lepődjünk meg, ha a beállításainkra vonatkozóan még kapunk további kérdéseket is!

24.8. A források követése több géppel

Írta: Mike Meyer.

Ha egyszerre több számítógéppel is szeretnénk követni ugyanannak a forrásfáának a változásait és ezért mindegyikre letöltjük a forrásokat majd újrafordítjuk ezeket, akkor sok erőforrást, de leginkább lemezterületet, hálózati sávszélességet és processzoridőt, feleslegesen használunk. Ezekkel úgy tudunk spórolni, ha valójában csak egyetlen géppel végeztetjük el a munka legtöbb részét, miközben a többi NFS használatával dolgozik. Ez a szakasz ezt a módszert foglalja össze.

24.8.1. Előkészületek

Először is szedjük össze az egyező binárisokat futtató gépeket, melyekre a továbbiakban csak *fordítási csoport* néven hivatkozunk. Minden gépnek lehet saját rendszermagja, viszont a felhasználói programok mindegyikjük esetében ugyanazok. Ebből a csoportból válasszuk ki egy *fordító gépet*. Ez lesz az a gép, amelyen a rendszer és a rendszermag lefordításra kerül. Ideális esetben ez a leggyorsabb gép, amelynek elegendő a processzorkapacitása arra, hogy lefuttassa a `make buildworld` és `make buildkernel` parancsokat. Érdemes még rajta kívül kiválasztanunk egy *tesztelő gépet* is, ahol a véglegesítés előtt kipróbálhatjuk a szoftverfrissítéseket. Ennek egy olyan gépnek *kell* lennie, amely akár hosszabb ideig is nélkülözhető a csoportból. Lehet akár maga a fordítást végző gép is, de nem elvárás.

A fordítási csoportban levő összes gépnek ugyanarról a gépről és ugyanarra a pontra kell csatlakoztatnia a `/usr/obj` és `/usr/src` könyvtárakat. Ezek optimális esetben a fordítással foglalkozó gép két külön lemezmeghajtóján vannak, melyek egyaránt elérhetőek NFS-en keresztül. Ha több fordítási csoportunk is van, akkor az `/usr/src` könyvtárnak elegendő csak egyetlen fordító gépen meglennie, a többi pedig csatlakoztassa NFS-en keresztül.

Végül győződjünk meg róla, hogy az `/etc/make.conf` és a `/etc/src.conf` állományok tartalma a fordítási csoport mindegyik gépénél megegyezik a fordító gépével. Ez azt jelenti, hogy a fordító gépnek az alaprendszer ugyanazon részeit és ugyanúgy kell létrehozni, mint amelyet a fordítási csoport akármelyik gépére telepíteni is akarunk. Ezenkívül még a fordítási csoportban levő minden egyes gép `/etc/make.conf` állományában a `KERNCONF` értékének a saját rendszermagjára vonatkozó konfigurációt kell megadni, illetve a fordítással foglalkozó gép `KERNCONF` változójánál pedig az együtt összeset, a sajátjával kezdve. Ennek megfelelően a fordító gépnek a rendszermagok lefordításához rendelkeznie kell az egyes gépek `/usr/src/sys/arch/conf` könyvtárában meglevő állományaival.

24.8.2. Az alaprendszer

Most, miután mindent megfelelően előkészítettünk, készen állunk a munkára. A 24.7.7.2 Szakaszban leírtak szerint fordítsuk le a rendszermagokat és az alaprendszert a fordító gépen, de utána még nem telepítsünk semmit se. Ha befejeződött a fordítás, lépünk be a tesztelő gépre és telepítsük a frissen fordított rendszermagot. Ha ez a gép NFS-en keresztül éri a `/usr/src` és `/usr/obj` könyvtárakat, akkor az egyfelhasználós módban aktiválni kell a hálózatot, majd csatlakoztatni ezeket. Ezt legkönnyebben úgy tudjuk megcsinálni, ha a gépet először elindítjuk többfelhasználós módban, majd a `shutdown now` paranccsal egyfelhasználós módba váltunk. Ha eljuttunk ide, telepítsük az új rendszermagot és rendszert, illetve a megszokott módon futtassuk a `merge master` parancsot. Amikor ezt befejeztük, ezen a gépen térjünk vissza a hétköznapi többfelhasználós működési módba.

Miután a tesztelésre szánt gépen ellenőriztük, hogy minden a megfelelő módon működik, az előbb tárgyalt eljárással telepítsük fel a fordítási csoportban levő összes többi gépre is az új szoftvereket.

24.8.3. Portok

Ugyanezt a gondolatmenet alkalmazható a portfa esetében is. Az első és egyben legfontosabb lépés a `/usr/ports` csatlakoztatása ugyanarról a gépről a fordítási csoport minden gépére. Az `/etc/make.conf` megfelelő beállításával még a terjesztési állományokat is meg tudjuk osztani. A `DISTDIR` értékét egy olyan közösen használt könyvtárra állítsuk, amely írható az NFS-en keresztül megosztott állományrendszerünkben a `root` felhasználóként tevékenykedők számára. A `WRKDIRPREFIX` változót minden gépen egy helyi fordítási könyvtárra állítsuk. Zárásképpen még hozzátesszük, hogy ha csomagokat akarunk készíteni és mások számára is elérhetővé tenni, akkor ne felejtsük el a `PACKAGES` változót a `DISTDIR` változóhoz hasonlóan beállítani.

Megjegyzések

1. Ez azért nem teljesen igaz. A régebbi FreeBSD kiadásokat ugyan nem támogathatjuk a végtelenségig, de általában így is több évig foglalkozunk velük. A FreeBSD régebbi kiadásával kapcsolatos jelenleg érvényes biztonsági házirend részletes bemutatása a <http://www.FreeBSD.org/security/> oldalon olvasható (angolul).

Fejezet 25. DTrace

Írta: Tom Rhodes.

25.1. Áttekintés

A DTrace, vagy más néven Dynamic Tracing technológiát a Sun dolgozta ki szerverek teljesítményében jelentkező szűk keresztmetszetek felderítésének megkönnyítésére. Ez nem egy nyomkövetésre szolgáló megoldást takar, hanem inkább a rendszer valós idejű elemzését és teljesítményének vizsgálatát elősegítő eszközt.

A DTrace figyelemre méltó elemzőeszköz, rengeteg rendkívül hasznos képességgel rendelkezik a rendszerben felbukkanó problémák diagnosztizálására. Előre programozott szkriptek segítségével pedig ezen képességek további előnyeit tudjuk kihasználni, ugyanis a DTrace programozható egy ún. D nyelven, amelynek révén a különböző vizsgálatokat könnyen a saját igényeink szerint tudjuk alakítani.

A fejezet elolvasása során megismerjük:

- mi is az a DTrace és milyen lehetőségei vannak;
- a Solaris és FreeBSD operációs rendszereken megtalálható DTrace implementációk közti eltéréseket;
- a DTrace FreeBSD alatt hogyan engedélyezhető és használható.

A fejezet elolvasásához ajánlott:

- a UNIX és FreeBSD alapvető ismerete (3 fejezet);
- a rendszermag konfigurációjának és fordításának alapvető ismerete (8 fejezet);
- az operációs rendszerek és azon belül a FreeBSD biztonsági fogalmainak minimális ismerete (14 fejezet);
- a FreeBSD forrásainak megszerzésének és azok lefordításának ismerete (24 fejezet).

Figyelem Ez a funkció még folyamatos tesztelés alatt áll. Bizonyos részei még egyáltalán nem, vagy csak korlátozottan érhetőek el. A dokumentáció annak megfelelően fog majd változni, hogy ezek az elemek fokozatosan elérik az éles felhasználáshoz szükséges szintet.

25.2. Eltérések az implementációban

Noha a FreeBSD alatt megtalálható DTrace implementáció nagyon hasonló az eredeti, Solaris alatt futó változathoz, tartalmaz bizonyos különbségeket, amelyeket a továbblépés előtt mindenképpen érdemes megemlítenünk. Az egyik legfontosabb ilyen szembetűnő különbség, hogy a FreeBSD esetén a DTrace használatát külön engedélyezni kell. A DTrace megfelelő működéséhez tehát a rendszermag konfigurációs állományában meg kell adnunk bizonyos beállításokat és modulokat kell betöltenünk. Ezekről hamarosan szó lesz.

A rendszermag konfigurációs állományában a `DDB_CTF` opció segítségével tudjuk engedélyezni ún. CTF adatok betöltését mind a rendszermag modulaiból, mind pedig magából a rendszermagból egyaránt. A CTF a Solaris “Compact Type Format” elnevezésű formátumára utal, amellyel például a DWARF megoldásához hasonló módon tárolhatunk tömörített alakban különböző típusú nyomkövetési információkat. Ilyen CTF adatok többek közt a

`ctfconvert` és a `ctfmerge` használatával rendelkezhetőek hozzá bináris állományokhoz. A `ctfconvert` segédprogram a fordítóprogram által az ELF állományokban szereplő DWARF típusú szakaszokban tárolt információkat képes beolvasni, és a `ctfmerge` a tárgykódban található CTF típusú ELF szakaszokat tudja végrehajtható állományokká vagy osztott könyvtárakka összefűzni. Röviden beszélni fogunk arról, hogyan lehet mindezeket a FreeBSD alaprendszerébe és rendszermagjába is beépíteni.

FreeBSD és Solaris esetén előfordulhat, hogy más fajta providerek állnak rendelkezésünkre. Ezek közül talán a legfontosabb a `dtmalloc`, amely a FreeBSD rendszermagjában típus szerint teszi lehetővé a `malloc()` függvény követését.

FreeBSD alatt kizárólag csak a `root` tudja használni a DTrace-t. Ennek oka a két operációs rendszer biztonsági megoldásai közti különbségekben keresendő, mivel a Solaris esetén létezik néhány olyan alacsony szintű ellenőrzés, amely a FreeBSD-nél még nincs. Ezért például a `/dev/dtrace/dtrace` eszköz szigorúan csak a `root` számára érhető el.

Végezetül megemlítjük, hogy a DTrace felhasználására a Sun CDDL licence vonatkozik. A Common Development and Distribution License FreeBSD a `/usr/src/cddl/contrib/opensolaris/OPENSOLARIS.LICENSE` állományban található, vagy interneten keresztül a <http://www.opensolaris.org/os/licensing> címen.

Ezen licenc értelmében a DTrace támogatással készített FreeBSD rendszermagok továbbra is BSD licencűek maradnak, azonban a rendszerrel terjesztett binárisok futtatásakor vagy a modulok betöltésekor már a CDDL érvényesül.

25.3. A DTrace támogatásának engedélyezése

A DTrace által felkínált lehetőségeket a következő sorok hozzáadásával tudjuk engedélyezni a rendszermag konfigurációs állományában:

```
options      KDTRACE_HOOKS
options      DDB_CTF
```

Megjegyzés: AMD64 architektúrán ezeken kívül még az alábbi sor is kelleni fog:

```
options      KDTRACE_FRAME
```

Ezzel a beállítással az FBT ("function boundary tracing") részére nyújtunk támogatást. A DTrace ugyan enélkül is képes lesz működni, de akkor csak korlátozott mértékben tudunk ilyen típusú vizsgálatokat végezni.

Az egész rendszert újra kell fordítanunk a CTF használatával. Ennek elvégzéséhez a következő parancsokat kell kiadnunk:

```
# cd /usr/src
# make WITH_CTF=1 kernel
```

A fordítás befejeződése után indítsuk újra a rendszerünket.

A rendszer újraindulása és az új rendszermag betöltődése után szükségünk lesz egy Korn-féle parancsértelmezőre is, mivel a DTrace eszköztárában rengeteg, a `ksh` programra épülő eszközt fogunk találni. Ezért tehát telepítsük a `shells/ksh93` csomagot, de megjegyezzük, hogy ugyanezen eszközök számára a `shells/pdksh` vagy `shells/mksh` csomagok is megfelelnek.

Végül töltsük le a DTrace eszköztárának legfrissebb változatát. Az aktuális verzió a <http://www.opensolaris.org/os/community/dtrace/dtracetoolkkit/> címen érhető el. Képes önmagát telepíteni, de a benne található eszközök használatához nem kötelező ezt elvégezni.

25.4. A DTrace használata

A DTrace funkcióinak alkalmazásához léteznie kell egy DTrace eszköznek. Ennek létrehozásához be kell töltenünk a megfelelő modult:

```
# kldload dtraceall
```

Innentől már működésre kész a DTrace. Rendszeradminisztrátorként a következő módon kérdezhetjük le a rendelkezésre álló vizsgálatokat:

```
# dtrace -l | more
```

Mivel lekérdezés eredménye pillanatok alatt betöltené az egész képernyőt, ezért az egészet még átirányítjuk a `more` parancshoz. Ha ez rendesen lefut, akkor a DTrace ténylegesen használhatónak tekinthető. Ezt követően tekintsük át a hozzá tartozó eszközkészletet.

Ez a mellékelt eszközkészlet lényegében a rendszerrel kapcsolatos információk összegyűjtésére alkalmas szkripteket tartalmaz. Vannak szkriptek, amelyekkel a megnyitott állományokat, a memóriát, a processzorhasználatot és még sok minden mást kérdezhetünk le. A szkriptek a következő parancs segítségével tömöríthetők ki:

```
# gunzip -c DTraceToolkit* | tar xvf -
```

A `cd` parancs segítségével lépünk be az így keletkező könyvtárba, és a kisbetűs névvel rendelkező állományok engedélyeit állítsuk be a 755 módra.

Mindegyik szkriptben el kell végeznünk némi módosítást: a `/usr/bin/ksh` hivatkozásokat írjuk át mindenhol a `/usr/local/bin/ksh` névre, illetve a `/usr/bin/sh` hivatkozásokat `/bin/sh` névre, majd végezetül pedig a `/usr/bin/perl` hivatkozásokat a `/usr/local/bin/perl` névre.

Fontos: Itt még egyszer kiemelnénk, hogy a FreeBSD-ben jelenleg megtalálható DTrace támogatás *még nem teljes és kísérleti jelleggel* szerepel. Ezért bizonyos szkriptek nem fognak működni, vagy azért, mert túlságosan Solaris lehetőségeihez igazodnak, vagy pedig azért, mert a jelenlegi implementáció által még nem ismert vizsgálatokra támaszkodnak.

Jelenlegi ismereteink szerint a FreeBSD egyelőre csak két szkriptet támogat teljes mértékben, ezek a `hotkernel` és a `procsystime`. A szakasz további részében ezzel a kettővel fogunk részletesebben foglalkozni.

A `hotkernel` feladata segíteni beazonosítani azokat a függvényeket, amelyek a legtöbb időt veszik igénybe a rendszermagon belül. A szkript futtatásakor nagyjából a következőt csinálja:

```
# ./hotkernel
Sampling... Hit Ctrl-C to end.
```

A folyamat **Ctrl+C** billentyűkombináció hatására állítható meg. A szkript futásának befejeződésekor különböző rendszermagbeli függvények és a hozzájuk tartozó idők jelennek meg, az utóbbi szerint növekvő sorrendben:

kernel`_thread_lock_flags	2	0.0%
0xc1097063	2	0.0%
kernel`sched_userret	2	0.0%
kernel`kern_select	2	0.0%
kernel`generic_copyin	3	0.0%
kernel`_mtx_assert	3	0.0%
kernel`vm_fault	3	0.0%
kernel`sopoll_generic	3	0.0%
kernel`fixup_filename	4	0.0%
kernel`_isitmxx	4	0.0%
kernel`find_instance	4	0.0%
kernel`_mtx_unlock_flags	5	0.0%
kernel`syscall	5	0.0%
kernel`DELAY	5	0.0%
0xc108a253	6	0.0%
kernel`witness_lock	7	0.0%
kernel`read_aux_data_no_wait	7	0.0%
kernel`Xint0x80_syscall	7	0.0%
kernel`witness_checkorder	7	0.0%
kernel`sse2_pagezero	8	0.0%
kernel`strncmp	9	0.0%
kernel`spinlock_exit	10	0.0%
kernel`_mtx_lock_flags	11	0.0%
kernel`witness_unlock	15	0.0%
kernel`sched_idletd	137	0.3%
0xc10981a5	42139	99.3%

Ez a szkript modulok esetén is alkalmazható. Ezt a módját a `-m` kapcsoló megadásával aktiválhatjuk:

```
# ./hotkernel -m
Sampling... Hit Ctrl-C to end.
^C
```

MODULE	COUNT	PCNT
0xc107882e	1	0.0%
0xc10e6aa4	1	0.0%
0xc1076983	1	0.0%
0xc109708a	1	0.0%
0xc1075a5d	1	0.0%
0xc1077325	1	0.0%
0xc108a245	1	0.0%
0xc107730d	1	0.0%
0xc1097063	2	0.0%
0xc108a253	73	0.0%
kernel	874	0.4%
0xc10981a5	213781	99.6%

A `procsystime` szkript egy adott azonosítóval vagy névvel rendelkező programhoz tudja megadni az általa kezdeményezett rendszerhívások által felhasznált időt. A most következő példában elindítjuk a `/bin/csh` egy újabb példányát. A `procsystime` elindul, majd megvárja, amíg kiadunk néhány parancsot a `csh` frissen indított másolatában. A teszt eredményei tehát a következők lesznek:

```
# ./procsystime -n csh
Tracing... Hit Ctrl-C to end...
```

^C

Elapsed Times for processes csh,

SYSCALL	TIME (ns)
getpid	6131
sigreturn	8121
close	19127
fcntl	19959
dup	26955
setpgid	28070
stat	31899
setitimer	40938
wait4	62717
sigaction	67372
sigprocmask	119091
gettimeofday	183710
write	263242
execve	492547
ioctl	770073
vfork	3258923
sigsuspend	6985124
read	3988049784

Jól megfigyelhető, hogy (nanomásodpercekben mérve) a legtöbb időt a `read()`, a legkevesebb időt pedig a `getpid()` rendszerhívás vette igénybe.

25.5. A D nyelv

A DTrace eszköztárban megtalálható számos szkript a DTrace saját programozási nyelvén íródott. Ezt a nyelvet nevezik a Sun implementációjában “a D nyelvnek”. Ennek ismertetésére itt most külön nem térünk ki, azonban a <http://wikis.sun.com/display/DTrace/Documentation> címen igen részletesen olvashatunk róla.

IV. Hálózati kommunikáció

A FreeBSD az egyik legelterjedtebb operációs rendszer a legnagyobb hálózati teljesítményt nyújtó kiszolgálók körében. Az itt található fejezetek témái:

- Soros kommunikáció
- PPP és PPP Etherneten keresztül (PPPoE)
- Elektronikus levelezés
- Hálózati kiszolgálók futtatása
- Tűzfalak
- Egyéb haladó hálózati témák

Ezek a fejezetek nem állnak egymással szoros kapcsolatban, csupán egy adott témáról adnak ismereteket. Ennélfogva nem kötelező ezeket sorrendben elolvasni, valamint egyáltalán nem is kell mindegyikőjüket átolvasni ahhoz, hogy a FreeBSD-t hálózati környezetben is használni tudjuk.

Fejezet 26. Soros vonali kommunikáció

26.1. Áttekintés

A UNIX mindig is támogatta a sörös vonali kommunikációt. Tulajdonképpen az első UNIX-os gépek is sörös vonalon kapták a felhasználóktól a bemenetet és ugyanígy küldték vissza a kimenetet. Az idők azóta már sokat változtak, hogy egy átlagos “terminál” mindössze egy 10 karakter per másodperc sebességű sörös nyomtatóból és egy billentyűzetből állt. Ebben a fejezetben ismertetünk néhány olyan megoldást, amellyel a FreeBSD képes sörös vonalon keresztül kommunikálni.

A fejezet elolvasása során megismerjük:

- hogyan kapcsoljunk terminálokat a FreeBSD rendszerünkre;
- hogyan tárcsázzunk modem segítségével távoli számítógépeket;
- hogyan tegyük lehetővé gépünkre a bejelentkezést távoli felhasználók számára;
- hogyan indítsuk a rendszerünket sörös konzolról.

A fejezet elolvasásához ajánlott:

- egy új rendszermag beállításának és telepítésének ismerete (8 fejezet);
- a UNIX-os engedélyek és a UNIX alatt futtatott programok működtetésének megértése (3 fejezet);
- annak a sörös vonali hardvernek (modemnek vagy többportos kártyának a) kézikönyve, amelyet a FreeBSD-vel használni szeretnénk

26.2. Bevezetés

26.2.1. Alapfogalmak

bps

Bit per másodperc — az adatátvitel sebessége

DTE

Adatterminál eszköz (Data Terminal Equipment) — ez például a számítógépünk

DCE

Adatkommunikációs eszköz (Data Communications Equipment) — ez a modem

RS-232

a hardveres sörös vonali kommunikációhoz szükséges EIA szabványú kábel

Amikor ebben a fejezetben az adatátvitel sebességéről beszélünk, akkor szándékosan nem használjuk a “baud” fogalmát. A baud ugyanis a kommunikációs eszközben adott idő alatt lezajló jelváltások mennyiségét jelöli, miközben itt a “bps” (bit per másodperc) kifejezés használata a *helyes* (vagy legalább is a szörszálhasogatók egyelőre megnyugodhatnak).

26.2.2. Kábelek és portok

Ha a FreeBSD rendszerünkhöz egy modemet vagy egy terminált akarunk csatlakoztatni, akkor ahhoz a számítógépünkben szükség lesz egy szabad sörös portra és egy megfelelő típusú kábelre. Ha már tisztában vagyunk a rendelkezésre álló hardverrel és a hozzá tartozó kábelrel, akkor nyugodtan átléphetjük ezt a részt.

26.2.2.1. A kábelek fajtái

A sörös kábeleknek több különböző típusa van. Közülük a céljainknak leginkább megfelelő két legismertebb változatuk az ún. null-modem és a szabványos (“egyeses”) RS-232-es sörös kábelek. A hardverhez tartozó dokumentációban megtaláljuk, hogy pontosan melyik típus tartozik hozzá.

26.2.2.1.1. A null-modem kábelek

Egy null-modem kábel bizonyos jeleket, többek közt a “földet” (Signal Ground, SG), egyenesen küldi, másokat viszont felcserélten. Például az “átküldött adat” (Transmitted Data, TD) jelzésű tű a kábel másik végén a “fogadott adat” (Received Data, RD) tűhöz fut be.

A terminálokhoz akár saját magunk is le tudunk gyártani egy null-modem kábelt (például ha a boltiakkal nem lennénk megelégedve). A következő táblázatban az RS-232C jeleit és érintkezőinek számozását láthatjuk egy DB-25-ös csatlakozó esetében. A szabvány a kábel két 1-es tűjét összekapcsoló vonalat *védőföldnek* (Protective Ground, PD) nevezi, de ezt gyakran el is hagyják. Némely terminál remekül működik mindössze a 2-es, 3-as és 7-es tűk használatával, miközben mások az iménti példától eltérő kiosztást igényelnek.

Táblázat 26-1. A DB-25 DB-25 közti null-modem kábel

Jel	Tű		Tű	Jel
SG	7	párja:	7	SG
TD	2	párja:	3	RD
RD	3	párja:	2	TD
RTS	4	párja:	5	CTS
CTS	5	párja:	4	RTS
DTR	20	párja:	6	DSR
DTR	20	párja:	8	DCD
DSR	6	párja:	20	DTR
DCD	8	párja:	20	DTR

Íme a mostanság elterjedt másik két séma.

Táblázat 26-2. A DB-9 DB-9 közti null-modem kábel

Jel	Tű		Tű	Jel
RD	2	párja:	3	TD
TD	3	párja:	2	RD
DTR	4	párja:	6	DSR
DTR	4	párja:	1	DCD
SG	5	párja:	5	SG
DSR	6	párja:	4	DTR
DCD	1	párja:	4	DTR
RTS	7	párja:	8	CTS
CTS	8	párja:	7	RTS

Táblázat 26-3. DB-9 DB-25 közti null-modem kábel

Jel	Tű		Tű	Jel
RD	2	párja:	2	TD
TD	3	párja:	3	RD
DTR	4	párja:	6	DSR
DTR	4	párja:	8	DCD
SG	5	párja:	7	SG
DSR	6	párja:	20	DTR
DCD	1	párja:	20	DTR
RTS	7	párja:	5	CTS
CTS	8	párja:	4	RTS

Megjegyzés: Amikor egy tű az átellenes oldalon két másik tűhöz csatlakozik, akkor azt általában úgy valósítják meg, hogy a két tűt a saját oldalukon összekötik, majd ezt kapcsolják hozzá a harmadik tűhöz.

Ezek a megoldások a legnépszerűbbek. Természetesen a tűk összekötésének több más variációja is létezik (ezekről az *RS-232 Made Easy* c. könyvben olvashatunk bővebben), ahol az SG párja az SG, a TD párja az RD, az RTS és a CTS párja az DCD, a DTR párja a DSR és ugyanezek fordítva.

26.2.2.1.2. Szabványos RS-232C kábelek

A szabványos soros kábel az összes RS-232C jelet közvetlenül átküldi. Vagyis a kábel egyik végén levő "átküldött adat" tű a másik végén is az "átküldött adat" tűhöz csatlakozik. Az ilyen típusú kábeleket többnyire a számítógépek és a modemek között alkalmazzák, de egyes termináltípusok esetében is szükségünk lehet rá.

26.2.2.2. A portok

A soros port olyan eszköz, amelyen keresztül a FreeBSD-s gép és a terminál között adatokat tudunk közvetíteni. Ebben a szakaszban az ilyen portok különféle típusait és ezek használatát ismertetjük FreeBSD alatt.

26.2.2.2.1. A portok típusai

A soros portoknak több típusa létezik. Mielőtt vásárolnánk egy készítenénk egy soros kábelt, mindenképpen győződjünk meg róla, hogy csatlakoztatni tudjuk majd a FreeBSD-s rendszerünkhöz és a terminálhoz egyaránt.

A legtöbb terminálon DB-25-ös portot találunk. A személyi számítógépek, köztük azok, amelyeken FreeBSD fut, DB-25-ös és DB-9-es portokkal rendelkeznek. Ha a gépünkben egy többportos soros kártya van, akkor ezeken kívül még RJ-12-es és RJ-45-ös portjaink is lehetnek.

A hardverhez tartozó dokumentációból tudjuk kideríteni az adott port konkrét fajtáját, de gyakran a port vizuális vizsgálata is segíthet eldönteni a kérdést.

26.2.2.2.2. A portok nevei

FreeBSD alatt az egyes soros portokat a `/dev` könyvtárban található eszközeleírókon keresztül tudjuk elérni. Ezeknek két típusa van:

- A behíváshoz használt portok nevei `/dev/ttydN` alakúak, ahol az *N* a port sorszáma, ami nullától indul. A behívó portok alapvetően a terminál esetében használatosak. A behívó portok használatához a soros vonalon az “vonal észlelése” (Data Carrier Detect, DCD) jelnek kell megbízhatóan működnie.
- A híváshoz használt portok nevei `/dev/cuaN` alakúak. A hívó portokat terminálok esetében ritkán alkalmazzák, helyettük inkább csak modemekhez használják. A hívó portokat akkor érdemes használni, ha a soros kábel vagy a terminál nem ismeri a DCD jelet.

Ha a terminált az első soros portra (ami MS-DOS-ban a COM1) csatlakoztattuk, akkor a `/dev/ttyd0` segítségével fogunk rá hivatkozni. Ha viszont a második soros porton (más néven COM2) található, akkor a `/dev/ttyd1` eszközt használjuk, és így tovább.

26.2.3. A rendszermag beállítása

A FreeBSD alapból négy soros portot támogat. Az MS-DOS világban ezeket rendre COM1, COM2, COM3 és COM4 portoknak nevezik. A FreeBSD jelen pillanatban ismeri még a “butább” többportos soros csatlakozókártyákat is, például a BocaBoard 1008 és 2016 típusokat, valamint több intelligensebb többportos kártyát, például a Digiboard és a Stallion Technologies gyártmányait. Az alap rendszermag azonban csak a szabványos COM portokat keresi.

Ha ellenőrizni akarjuk, hogy a rendszermag rendben megtalálta a soros portokat, akkor figyelmesen olvassuk el a rendszerindítás során megjelenő üzeneteket, vagy az `/sbin/dmesg` parancs kiadásával kérdezzük vissza a rendszermag üzeneteit. Különösen a `sio` kezdetű sorokra kell figyelnünk.

Tipp: Az alábbi paranccsal tudjuk leszűrni a `sio` szövegrészt tartalmazó sorokat:

```
# /sbin/dmesg | grep 'sio'
```

Például, ha négy soros port található a rendszerünkben, akkor a rájuk vonatkozó rendszerüzenetek a következők lesznek:

```
sio0 at 0x3f8-0x3ff irq 4 on isa
```

```
sio0: type 16550A
sio1 at 0x2f8-0x2ff irq 3 on isa
sio1: type 16550A
sio2 at 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
sio3 at 0x2e8-0x2ef irq 9 on isa
sio3: type 16550A
```

Ha a rendszermagunk nem ismerte volna fel az összes soros portot, akkor valószínűleg a `/boot/device.hints` állományt kell módosítanunk. Tegyük megjegyzésbe vagy akár teljesen távolítsuk is el azokat az eszközöket, amelyekkel nem rendelkezünk.

A soros portok és a többportos kártyák beállításával kapcsolatban a `sio(4)` man oldalát olvassuk el. Óvatosan bánjunk a FreeBSD megelőző változataiból származó konfigurációs állományokkal, mert az eszközök vonatkozó beállításokat és azok formátuma megváltozhatott azóta.

Megjegyzés: Az `port IO_COM1` a `port 0x3f8`, az `IO_COM2` a `0x2f8`, az `IO_COM3` a `0x3e8` és az `IO_COM4` a `0x2e8` beállítást helyettesíti. Ezek az adott porthoz tartozó gyakori címeket képviselik. A 4-es, 3-as, 5-ös és 9 megszakítások is igen általánosak ezeknél. A hagyományos soros portok viszont az ISA buszos PC-k esetében *nem képesek* a megszakításokon osztozni. (A többportos kártyák azonban lehetővé teszik az 16550A számára, hogy mindössze egy vagy két megszakítást használjon.)

26.2.4. Speciális eszközállományok

A rendszermagban található legtöbb eszköz az ún. “speciális eszközállományokon” keresztül érhető el, melyek a `/dev` könyvtárban találhatóak. A `sio` eszközök a `/dev/ttydN` (behívó portok) és `/dev/cuadN` (hívó portok) állományok használatával érhetőek el. A FreeBSD ezenkívül még külön eszközállományokat biztosít az inicializációhoz (`/dev/cuadN.init`) és a zároláshoz (`/dev/cuadN.lock`). Az inicializációs állományok a port megnyitásakor használhatóak a hozzá tartozó paraméterek beállítására, például így tudjuk elküldeni a `crtscs` utasítást az olyan modemeknek, amelyek a forgalom irányítását `RTS/CTS` jelzéseken keresztül valósítják meg. A zároló állományokkal a portokra vonatkozó zárolásokat állíthatjuk be, így a felhasználók vagy a programok nem lesznek képesek bizonyos paramétereket megváltoztatni. A `termios(4)`, `sio(4)` és `stty(1)` man oldalakon olvashatunk részletesebben a terminálok beállításairól, valamint az eszközök zárolásáról és inicializálásáról.

26.2.5. A soros port beállítása

A `tydN` (vagy `cuadN`) lesz az az eszköz, amit majd az alkalmazásainkból el akarunk érni. Amikor egy futó program megnyit egy ilyen eszközt, mindig tartoznak hozzá alapértelmezett terminál I/O beállítások. Ezeket a következő paranccsal tudjuk lekérdezni:

```
# stty -a -f /dev/ttyd1
```

Ha megváltoztatjuk az eszköz beállításait, akkor azok egészen addig érvényben is maradnak, amíg le nem zárjuk. Ha tehát ezután újra megnyitjuk, akkor minden visszaáll az alapértelmezett állapotra. Az alapértelmezett beállítások megváltoztatásához a “kezdeti állapotot” szimbolizáló eszközt kell megnyitnunk és átállítanunk. Például, ha alaphoz engedélyezni akarjuk a `CLOCAL` módot, a 8 bites kommunikációt és a `XON/XOFF` típusú forgalomirányítást a `tyd5` eszközön, akkor a következőt gépeljük be:

```
# stty -f /dev/ttyd5.init clocal cs8 ixon ixoff
```

A soros eszközök rendszerszintű inicializálását az `/etc/rc.d/serial` állomány vezérli. Lényegében ez határozza meg az összes soros eszköz alapértelmezett beállítását.

Ha bizonyos beállítások megváltoztatását tiltani szeretnénk az alkalmazások felé, akkor azt a “zárolt állapotot” tartalmazó eszközben kell rögzítenünk. Például, ha a `ttyd5` eszköz sebességét fixen 57600 bps-ra akarjuk beállítani, akkor írjuk be ezt:

```
# stty -f /dev/ttyd5.lock 57600
```

Ezután ha egy alkalmazás megnyitja a `ttyd5` eszközt és megpróbálja a port sebességét átállítani, akkor az továbbra is 57600 bps marad.

A kezdeti és a zárolt állapotot képező eszközöket általában csak a `root` felhasználó számára szabad írhatóvá tenni.

26.3. Terminálok

Készítette: Sean Kelly.

A terminálok olyankor kínálnak kényelmes és költséghatékony hozzáférést a FreeBSD rendszerünkhöz, amikor sem a gép konzolját, sem pedig a hozzá tartozó hálózatot nem érjük el. Ebben a szakaszban olvashatjuk, miként kell terminálokat használni FreeBSD alatt.

26.3.1. A terminálok alkalmazásai és típusai

Az eredeti UNIX rendszereknek nem voltak konzoljaik. Ehelyett az emberek a soros portokra csatlakoztatott terminálokon keresztül jelentkeztek be és így futtattak rajtuk programokat. Ez nagyon hasonlít ahhoz, mint amikor egy modem és egy terminálprogram felhasználásával betárcsázunk egy távoli gépre és vele szöveges módban dolgozunk.

Napjaink személyi számítógépein azonban találhatunk már akár nagy felbontású megjelenítéssel megáldott konzolokat is, habár a soros porton keresztüli bejelentkezés lehetősége még mind a mai napig elérhető a legtöbb UNIX-alapú rendszerben. Ez alól a FreeBSD sem kivétel. Ha rákötünk egy terminált a gépünk egyik üres soros portjára, akkor a megszokott módon képesek vagyunk bejelentkezni a rendszerbe és futtatni bármilyen szöveges programot, hasonlóan ahhoz, ahogy azt a konzolban vagy az X Window Systemben egy `xterm` ablakban megtehetjük.

Ha egy irodában vagyunk, akkor egy FreeBSD rendszerre több terminált is kapcsolhatunk, melyek az alkalmazottak asztalain foglalnak helyet. Otthoni használat esetén egy kiöregedett számítógép, például egy régi IBM PC vagy egy Macintosh is ráköthető egy gyorsabb FreeBSD rendszerre. Ennek segítségével az egyébként egyfelhasználós számítógépünket egy valódi többfelhasználós rendszerré alakíthatjuk.

A FreeBSD esetén háromféle terminálról beszélhetünk:

- A buta (dumb) terminálok
- A terminálként funkcionáló személyi számítógépek
- Az X terminálok

A most következő alszakaszokban ezeket fejtjük ki részletesebben.

26.3.1.1. A buta terminálok

A buta terminál alatt olyan speciálizált eszközt értünk, amellyel soros vonalon keresztül csatlakozunk számítógépekhez. Azért nevezik ezeket “butának”, mert csupán annyi számítási teljesítményt zsúfoltak beléjük, hogy szöveget legyenek képesek küldeni, fogadni és megjeleníteni. Semmilyen program nem képes rajtuk futni. Helyette az a számítógép fogja a szövegszerkesztőt, fordítóprogramot, levelező klienst, játékot és a többi futtatni, amelyre vele kapcsolódtunk.

A buta termináloknek többszáz, különböző gyártmányú fajtája létezik. Ilyenek például a Digital Equipment VT-100 vagy a Wyse WY-75 típusú termináljai. A FreeBSD szinte mindegyiküket ismeri. Egyes drágább terminálok még grafikus megjelenítésre is képesek, de ezeket a lehetőségeket csak bizonyos szoftverek tudják ténylegesen kihasználni.

A buta terminálok leginkább olyan munkahelyeken terjedtek el, ahol az alkalmazottaknak nincs szükségük grafikus alkalmazások, tehát például az X Window System használatára.

26.3.1.2. Személyi számítógépek mint terminálok

Ha egy buta terminál csupán szöveg küldésére, fogadására és megjelenítésére képes, akkor bármelyik személyi számítógép utána tudja mindezt csinálni. Ehhez mindössze egy megfelelő kábelre és az adott gépen futó *terminál emulációs* szoftverre van szükségünk.

Az ilyen fajta megoldás nagyon elterjedt az otthoni használat esetén. Például, ha valamelyik családtagunk éppen szorgalmasan dolgozik a FreeBSD rendszerkonzolján, akkor a rákapcsolt terminálon keresztül még mi magunk is el tudunk végezni valamennyi szöveges felületet igénylő munkát.

Az alap FreeBSD rendszerben legalább két segédprogram használható a soros vonali kapcsolaton keresztüli munkára: a `cu(1)` és a `tip(1)`.

Egy FreeBSD rendszerű kliensről így tudunk csatlakozni egy másik rendszerre:

```
# cu -l soros-vonali-eszköz
```

Ahol a “soros-vonali-eszköz” a rendszerünkben a soros portot jelölő speciális eszköz neve. Az ilyen eszközök neve `/dev/cuaN`.

Az eszköz nevében az “N”-es rész a soros port sorszámát adja meg.

Megjegyzés: A FreeBSD-ben az eszközök sorszámozása nullától kezdődik, nem pedig egytől (ellentétben tehát azzal, ahogy azt az MS-DOS rendszerekben és leszármazottaikban már megszokhattuk). Ez azt jelenti, hogy amit az MS-DOS alapú rendszerekben `COM1`-nek hívnak, az a FreeBSD-ben általában a `/dev/cua0`.

Megjegyzés: Egyes emberek más, többnyire a Portgyűjteményből is elérhető programokat szeretnek inkább használni. A portok között találhatunk elég sok olyan szoftvert, amely a `cu(1)` és a `tip(1)` programokhoz hasonlóan működik. Ilyen például a `comms/minicom`.

26.3.1.3. Az X terminálok

Az X terminálok a terminálok közül a legfejlettebbek. Általában nem is soros porton, hanem hálózaton, például Etherneten keresztül csatlakoznak. Természetesen nem csak szöveges alkalmazásokat, hanem lényegében bármilyen X alkalmazást képesek megjeleníteni.

Az X terminálokról itt most csak a teljesség kedvéért szólunk, de ebben a fejezetben *nem* szándékozunk tárgyalni az X terminálok csatlakoztatását, beállítását és használatát.

26.3.2. Beállítás

Ebben a fejezetben ismertetjük mindazt, ami ahhoz kell, hogy a FreeBSD rendszerünkön engedélyezni tudjuk a terminálon keresztüli bejelentkezéseket. Feltételezzük, hogy a rendszermagunk támogatja a terminálok által használt soros portokat, illetve, hogy ezeket már csatlakoztattuk is.

Ha visszagondolunk a 12 fejezetre, akkor eszünkbe juthat, hogy a rendszer indításakor az `init` nevű program felelős az összes futó program irányításáért és inicializálódásáért. Az `init` egyik feladata, hogy beolvassa az `/etc/ttys` állományt és neki megfelelően az elérhető terminálokon elindítsa a `getty` programot. A `getty` felelős a bejelentkezéshez szükséges azonosító beolvasásáért és a `login` program elindításáért.

Ennek megfelelően tehát, ha a FreeBSD rendszerünkön terminálokat akarunk beállítani, akkor ehhez a következő lépéseket kell megtennünk `root` felhasználóként:

1. Az `/etc/ttys` állományba vegyünk fel egy bejegyzést a soros porthoz tartozó `/dev` könyvtárbeli eszközhöz, ha még nem szerepelne benne.
2. A porthoz adjuk meg a `/usr/libexec/getty` programot, majd hozzá az `/etc/gettytab` állományból válasszuk ki a megfelelő `getty` típust.
3. Adjuk meg a terminál alapértelmezett típusát.
4. Állítsuk a portot "on" (bekapcsolt) állapotúra.
5. Adjuk meg, hogy a port "secure" (biztonságos) legyen-e.
6. Mondjuk meg az `init` programnak, hogy olvassa újra az `/etc/ttys` állományt.

A másik lépés kiegészítő lépéseként az `/etc/gettytab` állományban mi magunk is létrehozhatunk egy saját `getty` típust. A fejezetben ehhez ugyan nem adunk segítséget, de ha érdekel minket a téma, akkor ezzel kapcsolatban a `gettytab(5)` és `getty(8)` man oldalakat érdemes elolvasni.

26.3.2.1. Egy bejegyzés felvétele az `/etc/ttys` állományba

Az `/etc/ttys` állományban találhatjuk meg az összes portot, ahonnan a FreeBSD rendszerünk engedélyezi a bejelentkezést. Például a `ttyv0`, az első virtuális konzol is szerepel benne. Ezen a bejegyzésen keresztül tudunk bejelentkezni a konzolra. Ebben az állományban találjuk meg még a többi virtuális konzol, soros port és pszeudoterminál bejegyzéseit is. A rögzített terminálok esetén egyszerűen csak adjuk meg a soros porthoz tartozó `/dev` könyvtárbeli eszközt a `/dev` előtag nélkül (így például a `/dev/ttyv0` `ttyv0` néven fog megjelenni).

Az alap FreeBSD telepítésben egy olyan `/etc/ttys` állomány található, amely tartalmazza az első négy soros portot, a `ttyd0` eszköztől kezdve a `ttyd3` eszközig. Ha tehát ezekre a portokra csatlakoztatunk egy terminált, akkor már nem kell egy újabb bejegyzést felvennünk hozzájuk.

Példa 26-1. Terminálok felvétele az `/etc/ttys` állományba

Tegyük fel, hogy két eszközt szeretnénk a rendszerünkhöz csatlakoztatni: egy Wyse-50-es terminált és egy régi 286-os IBM PC-t, amelyen a **Procomm** terminálszoftverrel emulálunk egy VT-100-as terminált. A Wyse terminált a második soros portunkra kötjük, míg a 286-ost a hatodik soros portra (például egy többportos soros vonali kártyán). A nekik megfelelő `/etc/ttys` állománybeli bejegyzések így fognak kinézni:

```
ttyd1❶  "/usr/libexec/getty std.38400"❷  wy50❸  on❹  insecure❺
ttyd5   "/usr/libexec/getty std.19200"   vt100  on   insecure
```

- ❶ Az első mezőben általában a terminálhoz tartozó eszközt nevezzük meg, amely a `/dev` könyvtárban található.
- ❷ A második mező a vonalhoz tartozó végrehajtandó parancs, ami általában a `getty(8)`. A `getty` működésbe helyezi és megnyitja a vonalat, beállítja a sebességét, bekéri a felhasználó nevét, majd elindítja a `login(1)` programot.

A `getty` program egy (opcionális) paramétert fogad el a parancssorában, ami a `getty` típusa. Egy ilyen `getty` típus szabja meg a terminálhoz tartozó vonal jellemzőit, például az adatátviteli sebességet és a paritást. A `getty` ezeket a jellemzőket az `/etc/gettytab` állományból olvassa be.

A `/etc/gettytab` egyaránt tartalmaz bejegyzéseket a régi és új típusú terminálokhoz. Az `std` szöveggel kezdődő bejegyzések szinte majdnem minden esetben működnek a hardveres terminálokhoz. Az ilyen bejegyzések figyelmen kívül hagyják a paritást. 110 és 115 200 bps között minden adatátviteli sebességhez tartozik egy-egy `std` bejegyzés. Természetesen ebbe az állományba akár a saját bejegyzéseinket is elkészíthetjük. A `gettytab(5)` man oldal nyújt ehhez átfogó segítséget.

Amikor az `/etc/ttys` állományban megadjuk a `getty` típusát, akkor ellenőrizzük, hogy a beállításai megfelelnek a terminálénak.

A példánknál maradva: a Wyse-50 nem használ paritást és 38 400 bps-en üzemel. A 286-os gép szintén nem dolgozik paritással és 19200 bps-sel kapcsolódik.

- ❸ A harmadik mezőben adjuk meg általában a vonalra csatlakozó terminál típusát. Ez a betárcsázós portok esetében többnyire az `unknown` vagy a `dialup`, mivel ezeken keresztül a felhasználók gyakorlatilag szinte bármilyen típusú terminállal vagy szoftverrel be tudnak jelentkezni. A hardveres termináloknál a terminál típusa azonban nem változik, ezért a `termcap(5)` adatbázisban keressük ki a nekik megfelelőt és adjuk meg ebben a mezőben.

A példánkban a Wyse-50 egy valós termináltípust használ, miközben a 286-ason futó **Procomm** egy VT-100-as típusú terminált emulál.

- ❹ A negyedik mező azt mondja meg, hogy a port engedélyezett-e vagy sem. Ha itt a `on` értéket adjuk meg, akkor az `init` elindítja a második mezőben szereplő `getty` programot. Ha viszont itt az `off` szerepel, akkor a `getty` nem fog elindulni, így ezen a porton be sem fogunk tudni jelentkezni.
- ❺ Az utolsó mezőben a port megbízhatóságát kell megjelölnünk. Ha biztonságosnak (`secure`) állítjuk be a portot, akkor rajta keresztül a `root` (vagy bármelyik nullás felhasználói azonosítóval rendelkező) felhasználó be tud jelentkezni. Amikor viszont nem biztonságos (`insecure`), akkor először egy normál felhasználóval kell bejelentkeznünk, majd a `su(1)` programmal vagy egy hozzá hasonló megoldással kell rendszeradminisztrátorrá válnunk.

Leginkább az `insecure` beállítást javasoljuk, még hét lakat alatt őrzött terminálok esetében is. Valójában sokkal egyszerűbb bejelentkezni, majd kiadni egy `su` parancsot, ha netalán rendszeradminisztrátori jogosultságokra lenne szükségünk.

26.3.2.2. A `init` utasítása az `/etc/ttys` újraolvasására

Miután az `/etc/ttys` állományban elvégeztük a megfelelő módosításokat, a konfigurációs állomány újraolvasásához küldjünk egy `SIGHUP` (bontás) jelzést az `init` programnak. Mint például:

```
# kill -HUP 1
```

Megjegyzés: Mivel mindig az `init` indul el elsőként a rendszerben, ezért a hozzá tartozó azonosító az 1 lesz.

Ha mindent jól állítottunk be, a kábelek is a helyükön vannak és a terminálokat is bekapcsoltuk, akkor minden terminálhoz elindul egy `getty` program, és mindegyikükön megjelenik a bejelentkező képernyő.

26.3.3. A terminálokkal kapcsolatos hibajelenségek

Olykor hiába igyekszünk a lehető legaprólékosabban ügyelni minden apró részletre, könnyen előfordulhat, hogy valamiért a terminál mégsem működik rendesen. Következzen most egy lista néhány ismert tünetről és azok javasolt gyógy módjairól.

26.3.3.1. Nem jelenik meg a bejelentkező képernyő

Ellenőrizzük, hogy a terminált rendesen csatlakoztattuk és áram alá helyeztük. Amikor egy személyi számítógépet használunk terminálnak, akkor nézzük meg, hogy a terminál emulációs program a megfelelő soros porton fut.

Vizsgáljuk meg, hogy a kábel mind a két vége pontosan illeszkedik a portokba. Győződjünk meg róla, hogy valóban a megfelelő típusú kábelt használjuk.

Nézzük meg, hogy a terminál és a FreeBSD is ugyanazon az adatátviteli sebességen és paritási beállítással megy. Ha képernyővel rendelkező terminálunk van, akkor a kontrasztot és fényerősséget is ellenőrizzük. Ha nyomtatós terminálunk van, akkor vizsgáljuk meg a papír és a tinta állapotát.

Győződjünk meg róla, hogy a `getty` valóban fut és rendesen kiszolgálja a terminált. Például a `ps` paranccsal listázzuk ki az összes jelenleg futó programot és keressük meg köztük a `getty` programot:

```
# ps -axww|grep getty
```

Ekkor látnunk kell a terminálhoz tartozó bejegyzést. Például, ha a `getty` második soros portot jelképező `ttyd1` eszközön fut, és az `/etc/gettytab` állományból az `std.38400` nevű bejegyzést használja, akkor ez jelenik meg:

```
22189  d1  Is+      0:00.03 /usr/libexec/getty std.38400 ttyd1
```

Amennyiben semmilyen `getty` nem fut, akkor ellenőrizzük, hogy valóban engedélyeztük-e a portot az `/etc/ttys` állományban. A `ttys` állomány átírása után ne felejtsük el kiadni a `kill -HUP 1` parancsot sem.

Ha a `getty` fut, de a terminálon továbbra sem látjuk a bejelentkező képernyőt, vagy megjelenik, de nem tudunk gépelni, akkor előfordulhat, hogy a terminál vagy kábel nem támogatja a hardveres kézfogást (handshaking). Próbáljuk meg az `/etc/ttys` állományban levő `std.38400` bejegyzést az `3wire.38400` bejegyzésre kicserélni

(de utána ne felejtjük el kiadni a `kill -HUP 1` parancsot). A `3wire` nagyon hasonlít az `std` bejegyzéshez, de elhagyja a hardveres kézfogást. A `3wire` alkalmazásakor viszont a puffer telítődésének megelőzése érdekében próbálkozzunk az adatátviteli sebesség csökkentésével vagy engedélyezzük a szoftveres forgalomirányítást.

26.3.3.2. Amikor mindenféle szemét jelenik meg a képernyőn

Ellenőrizzük, hogy a FreeBSD és a terminál ugyanazt az adatátviteli sebességet és paritási beállítást használja. Nézzük meg a futó `getty` programokat, és hogy a megfelelő `getty` típussal mennek-e. Ha nem, módosítsuk az `/etc/ttys` állományt és adjuk ki a `kill -HUP 1` parancsot.

26.3.3.3. A karakterek duplán jelennek meg, a jelszó begépelésekor látható

Állítsuk át a terminált (vagy a terminál emulációs szoftvert) “half duplex” vagy “local echo” módról “full duplex” módra.

26.4. Betárcsázós szolgáltatások

Készítette: Guy Helmer. Kiegészítette: Sean Kelly.

Amikor egy FreeBSD rendszert akarunk betárcsázós szolgáltatásokhoz beállítani, akkor az nagyon hasonlít a terminálok csatlakoztatásához, azzal a eltéréssel, hogy ilyenkor a terminálok helyett modemekkel kell dolgoznunk.

26.4.1. Külső kontra belső modemek

A külső modemek sokkal kényelmesebbnek tűnnek betárcsázás szempontjából, mivel az ilyenek gyakran a statikus memóriájukban tárolt paraméterek révén tulajdonképpen félig előre be vannak állítva és sok esetben a fontosabb RS-232 jeleket külön lámpácskákkal mutatják. A villogó lámpák könnyen elkápráztatják a laikusokat, de emellett igen fontosak a modem működőképességének megállapításában is.

Ezzel szemben a belső modemeken nem található statikus memória, ezért a paramétereik csak DIP kapcsolókkal módosíthatóak. Még ha egy belső modemem látunk is lámpákat, akkor sem könnyű figyelni rájuk, mert a gépünk burkolata úgylis eltakarja ezeket.

26.4.1.1. Modemek és kábelek

Ha külső modemet használunk, akkor mindenképpen szükségünk lesz hozzá még egy megfelelő kábelre is. Egy szabványos RS-232-es soros kábel erre tökéletesen megfelel egészen addig, amíg a normál jeleket így kötötték be rajta:

Táblázat 26-4. A jelek neve

Rövidítés	Elnevezés
RD	Received Data (fogadott adat)
TD	Transmitted Data (küldött adat)
DTR	Data Terminal Ready (adatterminál kész)

Rövidítés

DSR

DCD

SG

RTS

CTS

Elnevezés

Data Set Ready (adatbeállítás kész)

Data Carrier Detect (vonal észlése — az RS-232 fogadást érzékelő vonala)

Signal Ground (föld)

Request to Send (küldés kérése)

Clear to Send (küldés engedélyezése)

A FreeBSD-nek 2400 bps felett a forgalom irányításához az RTS és CTS jelekre van szüksége. A CD jellel állapítja meg, hogy a hívás létrejött vagy a bontották a vonalat, és a DTR jel hozza alapállapotba a modemet a munkamenet befejezése után. Egyes kábelekben nem mindegyik jelet vezették át, így ha például gondjaink akadnak a bejelentkező képernyővel amikor a vonalat bontjuk, akkor érdemes átnéznünk a kábelt.

A többi UNIX-szerű operációs rendszerhez hasonlóan a FreeBSD is hardveres jelek segítségével igyekszik kideríteni, hogy a hívás megvalósult vagy bontották a vonalat, valamint a hívás befejezése után így bontja a vonalat és állítja vissza a modemet. A FreeBSD igyekszik elkerülni a parancsok küldését a modem felé, vagy a modem állapotának folyamatos ellenőrzését. Ha már van némi tapasztalatunk a PC-alapú BBS-ek modemelését illetően, akkor valószínűleg értjük ezek okait.

26.4.2. A soros vonali felülettel kapcsolatos megfontolások

A FreeBSD ismeri az NS8250-, NS16450-, NS16550- és NS16550A alapú EIA RS-232C (CCITT V.24) szabványú kommunikációs felületeket. A 8250-es és a 16450-es eszközök egykarakteres pufferral rendelkeznek. A 16550-es eszközök 16 karakteres puffert tartalmaznak, amellyel jobb teljesítmény érhető el. (A sima 16550-esben levő hibák miatt azonban ez a 16 karakteres puffer nem használható ki teljesen, ezért lehetőleg a 16550A verziót használjuk). Mivel az operációs rendszer részéről az egykarakteres eszközök jóval több törődést igényelnek, mint a 16 karakteres eszközök, ezért inkább a 16550A alapú soros felületi kártyákat ajánljuk. Amikor a rendszer egyszerre több soros portot is kezel, vagy erős terhelés alatt áll, akkor a 16550A alapú kártyákról általában az is elmondható, hogy kisebb hibával dolgoznak.

26.4.3. Egy gyors áttekintés

Ahogy arról már a terminálok esetében szó esett, az `init` az összes betárcsázós kapcsolathoz tartozó soros porthoz elindít egy `getty` programot. Például, ha a modemet a `/dev/ttyd0` eszközre kapcsoltuk, akkor a `ps ax` parancs kimenetében ezt láthatjuk:

```
4850 ?? I      0:00.09 /usr/libexec/getty V19200 ttyd0
```

Amikor egy felhasználó felhívja a modemet és az kapcsolódik, akkor a modem egy CD (Carrier Detect) jelet küld. A rendszermag ekkor tudomásul veszi a vonal észlelését és a `getty` segítségével megindítja a kommunikációt. A `getty` egy `login:` szöveget küld át a vonalhoz megadott sebességgel. A `getty` elkezd figyelni, hogy a értelmes karakterek érkeznek-e vissza, és egy átlagos konfigurációban, ha ezt szemétnek találja (mert például a modem nem a `getty` számára beállított sebességgel csatlakozott), akkor megpróbálja egészen addig hangolni a vonal sebességét, amíg feldolgozásra alkalmas karaktereket nem kap.

Miután a felhasználó megadta a felhasználói nevét, a `getty` elindítja a `/usr/bin/login` programot, amely befejezi a beléptetést a felhasználó jelszavának bekérésével és annak elfogadása esetén a hozzá tartozó parancsértelmező

elindításával.

26.4.4. A konfigurációs állományok

FreeBSD rendszerünkben a betárcsázós kapcsolatok engedélyezéséhez az `/etc` könyvtárban három állomány módosítására lesz szükségünk. Közülük az első, az `/etc/gettytab` a `/usr/libexec/getty` démon beállításait tartalmazza. A második, az `/etc/ttys` az `/sbin/init` számára mondja meg, hogy melyik `tty` eszközökhöz tartozik `getty`. Végezetül a portok inicializálásához kötődő beállításokat az `/etc/rc.d/serial` szkriptben kell megadnunk.

Két “iskola” jött létre aszerint, hogy UNIX alatt hogyan használják a betárcsázós modemeket. Az egyik csoport úgy szereti beállítani a modemeit és rendszerit, hogy a távoli felhasználó által választott sebességtől függetlenül a számítógép és a modem közti RS-232 felület egy fix sebességen fut. Ennek a beállításnak megvan az az előnye, hogy a távoli felhasználó ilyenkor szinte azonnal megkapja a bejelentkező képernyőt. A hátránya viszont, hogy ebben az esetben a rendszer nem ismeri a felhasználó valódi adatátviteli sebességét, ezért az olyan teljes képernyős alkalmazások, mint például az **Emacs**, nem lesznek képesek a lassabb kapcsolatokhoz szabni a megjelenítésüket.

A másik csoport a modemek RS-232-es felületét a távoli felhasználó kapcsolódási sebessége szerint állítja be. Így például egy V.32bis (14,4 Kbps) kapcsolat esetén a modemhez tartozó RS-232 felület 19,2 Kbps-on fog menni, miközben a 2400 bps sebességű kapcsolatokhoz egy vele azonos sebességű RS-232-es felület fog tartozni. Mivel a `getty` nem képes kommunikálni a modemek által lejelentett csatlakozási sebességen, ezért úgy próbálja azt megállapítani, hogy elküldi a `login:` szöveget az alap sebességgel, majd figyeli a válaszul érkező karaktereket. Ha a felhasználó ilyenkor szemetet lát, akkor feltételezik, hogy addig fogja nyomkodni az **Enter** billentyűt, amíg valami értelmes szöveget meg nem lát. Amikor az adatátviteli sebesség eltér, akkor a `getty` ebből csupán csak annyit vesz észre, hogy a felhasználó “szemetet” küld, ezért egy újabb sebességgel megpróbálja megint elküldeni a `login:` szöveget. Hivatalosan ez a folyamat ismétlődik orrvérzésig, de általában csak egy-két billentyűt kell leütni a megfelelő beállításokhoz. Nyilvánvaló, hogy ilyenkor a bejelentkezés messze nem olyan zavartalan, mint a “rögzített sebességű” esetben, de így a lassabb kapcsolattal rendelkező felhasználók is jobb használatóságot kapnak a teljes képernyős programokkal.

Ebben a szakaszban egy valamennyire kiegyensúlyozott beállítást igyekszünk bemutatni, de részben elfogunk hajlani abban az irányba, amikor a modem a kapcsolat sebességét követi.

26.4.4.1. `/etc/gettytab`

A `/etc/gettytab` egy termcap(5)-szerű állomány, amely a `getty(8)` beállításait tartalmazza. A `gettytab(5)` man oldalon olvashatunk az állomány pontos felépítéséről és benne felsorolt beállításokról.

26.4.4.1.1. A rögzített sebességű beállítás

Ha a modem kommunikációs sebességét rögzíteni akarjuk, akkor ehhez többnyire semmit sem kell megváltoztatnunk az `/etc/gettytab` állományban.

26.4.4.1.2. Az alkalmazkodó sebességű beállítás

Az `/etc/gettytab` állományban létre kell hoznunk egy olyan bejegyzést, amelyen keresztül a `getty` tudni fogja, hogy milyen sebességeken akarjuk használni a modemet. Ha egy 2400 bps sebességű modemünk van, akkor hozzá a már meglevő D2400-as bejegyzést kell használnunk.

#

```
# A gyors betárcsázós terminálokhoz íme egy 2400/1200/300-as váltás
# (bárhonnan kezdődhet):
#
D2400|d2400|Fast-Dial-2400:\
      :nx=D1200:tc=2400-baud:
3|D1200|Fast-Dial-1200:\
      :nx=D300:tc=1200-baud:
5|D300|Fast-Dial-300:\
      :nx=D2400:tc=300-baud:
```

Ha ennél gyorsabb modemünk van, akkor már mindenképpen fel kell vennünk hozzá egy új bejegyzést az `/etc/gettytab` állományba. Ezzel a beállítással egy 14,4 Kbps sebességű modemet tudunk legfeljebb 19,2 Kbps-en használni:

```
#
# Kiegészítések egy V.32bis modemhez:
#
um|V300|High Speed Modem at 300,8-bit:\
      :nx=V19200:tc=std.300:
un|V1200|High Speed Modem at 1200,8-bit:\
      :nx=V300:tc=std.1200:
uo|V2400|High Speed Modem at 2400,8-bit:\
      :nx=V1200:tc=std.2400:
up|V9600|High Speed Modem at 9600,8-bit:\
      :nx=V2400:tc=std.9600:
uq|V19200|High Speed Modem at 19200,8-bit:\
      :nx=V9600:tc=std.19200:
```

Ennek eredménye egy 8 bites, paritásmentes kapcsolat lesz.

A fenti példában a kommunikációt 19,2 Kbps-en (V.32bis kapcsolaton) kezdjük, majd utána haladunk végig a 9600 bps (V.32), 2400, 1200 bps és 300 bps sebességű kapcsolatokon, majd vissza ismét a 19,2 Kbps-re. Az adatátviteli sebesség ilyen típusú váltogatását az `nx=` (“next table”, azaz “következő táblázat”) tulajdonság segítségével valósítják meg. Minden sorban látható még egy `tc=` (“table continuation”, vagyis “a táblázat folytatása”) bejegyzés is, amivel az adott adatátviteli sebesség “szabványos” beállításait adjuk meg.

Ha egy 28,8 Kbps sebességű modemünk van és/vagy egy 14,4 Kbps sebességű modemem akarunk tömörítést használni, akkor a 19,2 Kbps-nél nagyobb kommunikációs sebességet kell használnunk. Íme egy olyan `gettytab`, ami 57,6 Kbps-ről indít:

```
#
# A V.32bis vagy V.34 modemekhez kiegészítés,
# 57,6 Kbps-ről indulunk:
#
vm|VH300|Very High Speed Modem at 300,8-bit:\
      :nx=VH57600:tc=std.300:
vn|VH1200|Very High Speed Modem at 1200,8-bit:\
      :nx=VH300:tc=std.1200:
vo|VH2400|Very High Speed Modem at 2400,8-bit:\
      :nx=VH1200:tc=std.2400:
vp|VH9600|Very High Speed Modem at 9600,8-bit:\
      :nx=VH2400:tc=std.9600:
vq|VH57600|Very High Speed Modem at 57600,8-bit:\
```

```
:nx=VH9600:tc=std.57600:
```

Ha lassú a processzorunk, vagy a rendszerünk túlságosan terhelt és nincs 16550A típusú soros portunk, akkor 57,6 Kbps-en sio “silo” hibák keletkezhetnek.

26.4.4.2. /etc/ttys

Az /etc/ttys állomány beállításáról már a 26-1 Példa adott képet. Ez a modemek esetében sem tér el különösebben, habár a `getty` programnak más termináltípust és -beállításokat kell átadnunk. Akár rögzített, akár alkalmazkodó sebességet akarunk beállítani, ennek általános alakja az alábbi:

```
ttyd0    "/usr/libexec/getty xxx"    dialup on
```

A sorban látható első elem a megfelelő speciális eszköz neve — jelen esetben ez a `ttyd0`, amely a `/dev/ttyd0` eszközre vonatkozik és ezt fogja a `getty` figyelni. A második elem, vagyis a `"/usr/libexec/getty xxx"` (ahol a `xxx` helyére kell beírni a megfelelő `gettytab` állománybeli bejegyzést nevét) lesz az a parancs, amelyet az `init` meghív. A harmadik elem, a `dialup` a terminálok alapértelmezett típusa. A negyedik paraméter, az `on` jelzi az `init` programnak, hogy aktiválja a vonalat. A sorban megjelenhetne továbbá még egy ötödik paraméter is, a `secure`, de ezt csak olyan terminálok esetében érdemes megadni, amelyek fizikailag megbízhatóak (például a rendszerkonzol).

Az alapértelmezett termináltípus (vagyis a fenti példában a `dialup`) a helyi beállításoktól függ. A betárcsázós vonalak esetében hagyományosan a `dialup` a terminál alapértelmezett típusa, amit aztán a felhasználók a bejelentkezéskor lefutó szkriptjeiken keresztül a automatikusan át tudnak állítani a nekik megfelelő terminálra. A szerző saját rendszerében azonban inkább a `vt102` termináltípust volt érdemes megadni alapértelmezettként, mivel ott a felhasználók csak ilyen típusú terminálokat használnak.

Miután az /etc/ttys állományban elvégeztük a szükséges módosításokat, egy HUP jelzéssel figyelmeztessük az `init` programot az újbóli beolvasására. Ehhez a következő parancs ajánlott:

```
# kill -HUP 1
```

Ha még csak állítjuk be először a rendszerünket, akkor az `init` figyelmeztetése előtt legyünk türelmesek, és várjuk meg, amíg a modemek befejezik az inicializálást és kapcsolódnak a vonalakra.

26.4.4.2.1. A rögzített sebességű beállítás

A rögzített sebesség beállításánál a `ttys` állományban a `getty` paramétereként egy szintén rögzített sebességű bejegyzést kell megadnunk. Például az olyan modemeknél, ahol a sebességet 19,2 Kbps-re rögzítjük, a `ttys` így fog kinézni:

```
ttyd0    "/usr/libexec/getty std.19200"    dialup on
```

Amennyiben a modemünk nem ezen a sebességen üzemelne, akkor az `std.sebesség` paramétert használjuk az `std.19200` helyett. Előtte azonban ne felejtsük el ellenőrizni, hogy a megadott típus szerepel-e az /etc/gettytab állományban.

26.4.4.2.2. Az alkalmazkodó sebességű beállítás

Az alkalmazkodó sebességű beállításnál a `ttys` állományban az `/etc/gettytab` állományból a megfelelő "auto-baud" (sic) kell megadnunk. Például, ha modemünk kezdősebessége 19,2 Kbps (és a `gettytab` ehhez tartalmaz egy `V19200` nevű bejegyzést), akkor a `ttys` így fog kinézni:

```
ttyd0    "/usr/libexec/getty V19200"    dialup on
```

26.4.4.3. /etc/rc.d/serial

A gyorsabb, mint például a V.32, V.32bis és V.34 modemeknél meg kell adnunk a hardveres forgalomirányítás (RTS/CTS) használatát is. Az `/etc/rc.d/serial` állományban tudjuk megadni a FreeBSD rendszermagban a vonal használatához szükséges vezérlési beállításokra vonatkozó `stty` parancsokat.

Például állítsuk be az 1-es sorszámú (vagyis a COM2) soros porton a `crtsets` termios beállítást a behíváshoz és a híváshoz használt eszközök inicializálásakor. Ehhez a következő sorokat kell felveszünk az `/etc/rc.d/serial` állományba:

```
# A soros portok kezdeti beállításai:
stty -f /dev/ttyd1.init crtsets
stty -f /dev/cuad1.init crtsets
```

26.4.5. A modemek beállításai

Ha olyan modemeink vannak, amelyek paramétereit egy statikus memóriában tárolták le, akkor ezek beállításához egy terminálprogramot kell használnunk (amilyen például MS-DOS alatt a **Tel**ix vagy FreeBSD alatt a `tip`). A modemet a `getty` programnak megadott kezdeti sebességen csatlakoztassuk és az alábbi elvárások alapján állítsuk be a paramétereit:

- Kapcsolódáskor CD jelzése.
- Működéskor DTR jelzése. A DTR küldésekor bontsa a vonalat és hozza alapállapotba a modemet.
- CTS vezérlésű kimenő adatforgalom.
- A XON/XOFF forgalomvezérlés tiltása.
- RTS vezérlésű bejövő adatforgalom.
- Csendes mód (ne adjon értesítést az eredményekről).
- A parancsokat ne írja vissza.

A modemhez tartozó dokumentációban kell utánajárnunk, hogy milyen parancsok és/vagy DIP kapcsolók átállításával lehet mindezeket elérni.

Például, ha a fenti paramétereket egy U.S. Robotics® Sportster® 14400-as külső modem esetében a következő neki kiküldött paranccsal lehet beállítani:

```
ATZ
AT&C1&D2&H1&I0&R2&W
```

Ilyenkor még akár más egyéb paramétereket is beállíthatunk, például a V.42bis és/vagy az MNP5 tömörítést.

Az U.S. Robotics Sportster 14400 külső modemem ezenkívül még találunk néhány DIP kapcsolót is. Az ilyen modemek esetében például ezeket a beállításokat tudjuk használni:

- 1. kapcsoló: FEL — normális DTR
- 2. kapcsoló: N/A (verbális/numerikus eredményjelző kódok)
- 3. kapcsoló: FEL — az eredményjelző kódok küldésének tiltása
- 4. kapcsoló: LE — nem küldi vissza a parancsokat
- 5. kapcsoló: FEL — automatikus válasz
- 6. kapcsoló: FEL — normális Carrier Detect
- 7. kapcsoló: FEL — a memóriában tárolt alapértelmezések betöltése
- 8. kapcsoló: N/A (intelligens/buta mód)

A modemeknél az eredményjelző kódok kikapcsolása/letiltása ezért fontos, mert így el tudunk kerülni az olyan problémákat, hogy a `getty` tévesen egy `login:` promptot küld a parancs módban levő modemnek, amikor az visszaküldi a parancsot és az eredmény kódját. Ennek eredménye egy hosszúra nyúló, zavaros társalgás lesz a `getty` és a modem között.

26.4.5.1. A rögzített sebességű beállítás

A rögzített sebességű konfiguráció használata esetén úgy kell beállítanunk a modemet, hogy a konkrét adatátviteli sebességtől függetlenül is egy állandó sebességű kapcsolat álljon fenn a számítógép és a modem között. A U.S. Robotics Sportster 14400-as külső modem esetében a most következő parancsokkal tudjuk rögzíteni a kapcsolat sebességét:

```
ATZ
AT&B1&W
```

26.4.5.2. Az alkalmazkodó sebességű beállítás

Amikor változó sebességű konfigurációval dolgozunk, akkor a modemet úgy kell beállítani, hogy a bejövő hívásnak megfelelő adatátviteli sebességre váltson a soros portján. A U.S. Robotics Sportster 14400-as külső modem esetében az alábbi parancsokkal rögzítjük a modemnek küldött hibamentesített parancsok sebességét, miközben engedélyezzük, hogy a soros port sebessége változhasson a nem hibamentesített kapcsolatoknál:

```
ATZ
AT&B2&W
```

26.4.5.3. A modem beállításainak ellenőrzése

A legtöbb nagysebességű modem biztosít valamilyen lehetőséget arra, hogy emberi formában is le tudjuk kérdezni a belső működésének paramétereit. A U.S. Robotics Sportster 14400-as külső modem esetében az `ATI5` parancs a statikus memóriában tárolt beállításokat mutatja meg. A modem valós működési paramétereit (amit ugyebár befolyásolnak a DIP kapcsolók állásai is) viszont az `ATZ` majd `ATI4` parancsok küldésével tudjuk lekérni.

Ha azonban másmilyen márkájú modemünk lenne, akkor a modem leírásában próbáljunk tájékozódni arról, miként tudjuk a modem beállításait ellenőrizni.

26.4.6. Hibaelhárítás

Ebben a szakaszban bemutatunk néhány lépést, amelyeken keresztül ellenőrizhetjük a rendszerünkhöz csatlakoztatott modemet.

26.4.6.1. A FreeBSD rendszer ellenőrzése

Csatlakoztassuk a modemet a FreeBSD rendszerre, indítsuk be a gépet, majd ezután figyeljük a modemünk állapotát jelző lámpákat, hogy közülük a DTR világít-e, amikor a `login:` felirat megjelenik a rendszerkonzolon. Amennyiben erre a válasz igen, akkor az arra utal, hogy a FreeBSD a hozzá tartozó kommunikációs porton elindította a megfelelő `getty` programot és a modem várja a hívásokat.

Amikor viszont a DTR lámpa nem világít, a konzolon keresztül jelentkezzünk be a FreeBSD rendszerbe és adjuk ki egy `ps ax` parancsot, amivel így ellenőrizni tudjuk, hogy a porthoz tartozó `getty` elindult. A futó programok között tehát valami ilyesmit kell majd látnunk:

```
114 ?? I      0:00.10 /usr/libexec/getty V19200 ttyd0
115 ?? I      0:00.10 /usr/libexec/getty V19200 ttyd1
```

Ha viszont például ezt látjuk:

```
114 d0 I      0:00.10 /usr/libexec/getty V19200 ttyd0
```

és modem még nem fogadott hívást, akkor ez azt jeleníti, hogy a `getty` megnyitotta a kommunikációs csatornát. Ez utalhat egyaránt egy hibás kábelre vagy a modem helytelen beállítására, mivel a `getty` egészen addig nem lesz képes megnyitni az adott portot, amíg a modem vissza nem küld neki egy CD (Carrier Detect) jelet.

Ha a listában az adott `ttydN` eszközhöz semmilyen `getty` programot nem találunk, akkor újra nézzük át az `/etc/ttys` állományban szereplő bejegyzéseket, mert előfordulhat, hogy azokban vétettünk valamilyen hibát. Emellett még a `/var/log/messages` naplóban is érdemes utánanézni, hátha az `init` vagy a `getty` küldött valamilyen hibáról értesítést. Ha még ezek után sem találunk semmit, akkor megint kezdjük el keresni hibákat, hiányzó bejegyzéseket vagy eszközöket az `/etc/ttys`, `/etc/gettytab` és a megfelelő `/dev/ttydN` állományokban.

26.4.6.2. A betárcsázás kipróbálása

Próbáljunk meg bejutni a rendszerünkbe. Ehhez a távoli rendszeren ne felejtsük el beállítani a 8 bites adatátvitelt és az 1 stopbitet, illetve a paritást kikapcsolni. Ha erre közvetlenül nem kapunk egy bejelentkezési képernyőt vagy csak szemét jelenik meg, akkor kb. másodpercenként egyszer nyomjuk le az **Enter** billentyűt. Ha még ezután sem látjuk a bejelentkezési képernyőt felbukkani, akkor próbáljunk kiküldeni egy `BREAK` parancsot. Ha a híváshoz nagysebességű modemet használunk, akkor próbáljuk meg a modem sebességét rögzíteni és úgy tárcsázni (ezt például a U.S. Robotics Sportster modemnél az `AT&B1` parancssal tudjuk elérni):

Ha viszont még ezek után sem kapjuk meg a bejelentkező képernyőt, akkor a `/etc/gettytab` állományban megint nézzük át az összes beállítást:

- Az `/etc/ttys` állományban megadott alaptulajdonság neve egyezik az `/etc/gettytab` állományban találhatóval.
- Mindegyik `nx=` bejegyzés után egy másik `gettytab` tulajdonság neve jön.
- Mindegyik `tc=` bejegyzés után egy másik `gettytab` tulajdonság neve következik.

Ha hívunk, de a FreeBSD rendszerünkre kapcsolt modem továbbra sem veszi fel, akkor a modem beállításai között ellenőrizzük, hogy a DTR jel küldésekor a modem fogadja-e a hívást. Ha úgy tűnik, hogy a modem minden ezzel kapcsolatos beállítása stimmel, akkor nézzük meg, hogy a modem lámpái közül a DTR világít-e (már ha van ilyen).

Ha mindent többször is végignéztünk és még mindig nem leljük a megoldást, akkor tartsunk egy kis szünetet és térjünk vissza a problémához később. Ha még ezután sem tudjuk működéssé bírni, akkor küldjünk egy levelet a FreeBSD general questions levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>) címére, amelyben leírjuk a modemünket és a vele kapcsolatos problémát, és a lista tagjai majd megpróbálnak nekünk segíteni.

26.5. A betárcsázós szolgáltatások használata

A következőkben arra vonatkozóan igyekszünk tanácsokat adni, amikor mi magunk akarunk modemmel csatlakozni valamilyen számítógéphez. Ezek tehát olyan esetekben hasznosak, amikor egy távoli géppel akarunk terminálkapcsolatot létesíteni.

A BBS-ek használatára is érvényes.

Ez ilyen típusú kapcsolatok kifejezetten hasznosak tudnak lenni olyan esetekben, amikor az interneten el akarunk érni egy állományt, de gondjaink akadtak a PPP használatával. Ha például egy állományt akarunk letölteni, de a PPP valamiért nem működik, akkor ezt a terminál alapú kapcsolaton keresztül is meg tudjuk tenni. Ilyenkor egy zmodem segítségével tudjuk áttölteni a számítógépünkre.

26.5.1. A gyári Hayes-modem erre nem alkalmas, mihez tudunk vele kezdeni?

A `tip` man oldala valójában már nem is teljesen aktuális, ugyanis tartalmaz egy beépített Hayes-tárcsázót. Úgy tudjuk engedélyezni, ha az `/etc/remote` állományban megadjuk az `at=hayes` beállítást.

A Hayes-eszközök meghajtója nem elég ügyes ahhoz, hogy felismerje az újabb modemek által felkínált fejlettebb lehetőségeket — például a `BUSY, NO DIALTONE` vagy a `CONNECT 115200` üzenetek csak megzavarják. Ezért a `tip` használata során kapcsoljuk ki ezeket az üzeneteket (az `ATXO&W` paranccsal).

Emellett még érdemes tudni, hogy a `tip` a híváskor 60 másodpercig vár. A modemünkön ennél kisebb időt kell beállítanunk, máskülönben a `tip` azt hiszi, hogy valamilyen kommunikációs probléma merült fel. Ehhez próbálkozzunk az `ATS7=45&W` paranccsal.

26.5.2. Hogyan adjuk meg ezeket az AT parancsokat?

Az `/etc/remote` állományban hozzunk létre egy “direct” bejegyzést. Például, ha a modemünk az első soros porton, vagyis a `/dev/cuad0` eszközön tanyázik, akkor a következő sort kell beleírunk:

```
cuad0:dv=/dev/cuad0:br#19200:pa=none
```

A `br` tulajdonságnál a modem által ismert legnagyobb adatátviteli sebességet adjuk meg. Ezután gépeljük be a `tip` `cuad0` parancsot és már kapcsolódunk is a modemhez.

Vagy `root` felhasználóként a `cu` parancsot is használhatjuk:

```
# cu -lvonal -ssebesség
```

Itt a `vonal` a soros port (például `/dev/cuad0`) és a `sebesség` annak sebessége (például `57600`) lesz. Miután befejeztük az AT parancsok kiadását, az `~.` begépelésével tudunk kilépni.

26.5.3. A `pn` tulajdonságnál a `@` jel nem használható!

A `pn` (“phone number”) tulajdonság értékében szereplő `@` jel segítségével az `/etc/phones` állományban tudunk hivatkozni egy telefonszámra. A `@` a tulajdonságokat tároló állományok azonban, így például az `/etc/remote` állomány esetén is megkülönböztetett jelentéssel bírnak. Ezért itt csak egy visszaper jellel tudjuk beírni:

```
pn=\@
```

26.5.4. Hogyan hívjunk fel egy számot parancssorból?

Tegyük egy “általános” bejegyzést az `/etc/remote` állományunkba. Például egy ilyen:

```
tip115200|Dial any phone number at 115200 bps:\
      :dv=/dev/cuad0:br#115200:at=hayes:pa=none:du:
tip57600|Dial any phone number at 57600 bps:\
      :dv=/dev/cuad0:br#57600:at=hayes:pa=none:du:
```

Ezután már ilyen is tudni fogunk:

```
# tip -115200 5551234
```

Ha viszont a `tip` helyett inkább a `cu` programot használnánk szívesen, akkor ehhez készítsünk egy általános bejegyzést:

```
cu115200|Use cu to dial any number at 115200bps:\
      :dv=/dev/cuad1:br#57600:at=hayes:pa=none:du:
```

Majd gépeljük be ezt:

```
# cu 5551234 -s 115200
```

26.5.5. Ehhez minden adandó alkalommal meg kell adnom a sebességet is?

Hozzunk létre egy `tip1200` vagy `cu1200` nevű bejegyzést, de a `br` tulajdonságnál adjuk meg a használni kívánt sebességet. Mivel a `tip` szerint az `1200 bps` egy megfelelő alapértelmezés, ezért alpból a `tip1200` bejegyzést fogja keresni. Ez természetesen nem jelenti azt, hogy ilyen sebességgel is akarunk dolgozni.

26.5.6. A terminálszerveren keresztül több más gépet is elérek

Ahelyett, hogy minden alkalommal megvárnánk a kapcsolódás befejezést és begépnénk a `CONNECT gép` parancsot, használjuk a `cm` tulajdonságát. Például nézzük meg ilyen bejegyzést az `/etc/remote` állományban:

```
pain|pain.deep13.com|Forrester's machine:\
      :cm=CONNECT pain\n:tc=deep13:
muffin|muffin.deep13.com|Frank's machine:\
      :cm=CONNECT muffin\n:tc=deep13:
deep13:Gizmonics Institute terminal server:\
      :dv=/dev/cuad2:br#38400:at=hayes:du:pa=none:pn=5551234:
```

Ennek hatására elég csak annyit megadnunk, hogy `tip pain` vagy `tip muffin`, és már kapcsolódunk is a `pain` vagy `muffin` gépekhez. A `tip deep13` paranccsal pedig egyenesen a terminálszerverhez jutunk el.

26.5.7. Több vonalon is lehet egy géphez csatlakozni?

Ez gyakran okoz gondot olyan esetekben, amikor egy egyetemnek több betárcsázó vonala van, és azokon keresztül több ezer hallgató próbál meg dolgozni.

Vegyük fel az egyetemet az `/etc/remote` állományba és használjuk a `pn` tulajdonság megadásánál a `@` jelet:

```
nagy-egyetem:\
      :pn=@:tc=dialout
dialout:\
      :dv=/dev/cuad3:br#9600:at=courier:du:pa=none:
```

Ezután adjuk hozzá az `/etc/phones` állományhoz az egyetem telefonszámait:

```
nagy-egyetem 5551111
nagy-egyetem 5551112
nagy-egyetem 5551113
nagy-egyetem 5551114
```

A `tip` mindegyik telefonszámot az adott sorrendben próbálja tárcsázni és végén feladja a próbálkozást. Ha folyamatosan akarjuk ezeket a számokat hívni, akkor `tip` parancsot tegyük egy ciklusba.

26.5.8. Miért kell kétszer lenyomni a Ctrl+P gombokat, hogy egyszer elküldje a Ctrl+P kombinációt?

A **Ctrl+P** billentyűkombináció alapértelmezés szerint a “kikényszerítést” jelenti, amivel a `tip` programnak tudunk szólni, hogy a következő adat szó szerint értendő. A `~s` szekvenciával bármelyik másik karakternek át tudjuk adni ezt a szerepet, ami egy változó beállítását jelenti (“set a variable”).

Gépeljük be, hogy `~sforce=egyetlen-karakter` és zárjuk le egy újsorral. Az *egyetlen-karakter* helyére tetszőleges, egykarakteres szimbólumot megadhatunk. Ha itt nem adunk meg semmit, akkor a kikényszerítő karakter a `null` lesz, amit a **Ctrl+2** vagy a **Ctrl+Szóköz** lenyomásával tudunk előhozni. Az *egyetlen-karakter* szerepére például tökéletes a **Shift+Ctrl+6**, amit csak nagyon kevés terminálszerver alkalmaz.

A kikényszerítést végző karaktert az `$HOME/.tiprc` állományban tetszőleges karakterre át tudjuk állítani:

```
force=egyetlen-karakter
```

26.5.9. Miért lett hirtelen minden begépelte betű nagybetűs??

Valószínűleg sikerült lenyomnunk a **Ctrl+A** gombkombinációt, ami a `tip` “betűmód váltás” funkciójának felel meg. Ezt olyanok számára dolgozták ki, akiknél nem működik a **Caps Lock** billentyű. Az előbb bemutatott `~s` használatával állítsuk át a `raisechar` változót valami másra. Tulajdonképpen akár ugyanarra is állíthatjuk, mint a kikényszerítő karaktert, ha nem áll szándékunkban használni.

Ebben a példában egy olyan `.tiprc` állomány szerepel, amely tökéletesen megfelel azon **Emacs** felhasználók számára, akik sokat használják a **Ctrl+2** és **Ctrl+A** kombinációkat:

```
force=^^
raisechar=^^
```

A `^^` a **Shift+Ctrl+6** billentyűkombinációt jelenti.

26.5.10. Hogyan mozgassunk állományokat a `tip` használatával?

Amikor más UNIX rendszerekkel vesszük fel a kapcsolatot, akkor állományokat a `~p` (mint put, vagyis adni) és `~t` (mint take, vagyis venni) használatával tudunk mozgatni. Ezek a parancsok a távoli rendszeren a `cat` és az `echo` felhasználásával fogadnak és küldenek állományokat. Alakjuk a következő:

`~p helyi-állomány [távoli-állomány]`

`~t távoli-állomány [helyi-állomány]`

Ilyenkor nincs hibaellenőrzés, ezért inkább egy másik protokollt, például `zmodem`et érdemes használnunk.

26.5.11. Hogyan lehet `zmodem`et használni a `tip` programban?

Állományokat úgy tudunk fogadni, ha előtte a kapcsolat távolabbi végén elindítjuk a küldést végző programot. Ezután a `~C rz` parancs kiadásával kezdetjük meg helyben a fogadását.

Állományokat úgy tudunk küldeni, ha előtte a kapcsolat másik végén elindítjuk a fogadó programot. Ezután a `~C sz` *állományok* parancs kiadásával tudjuk megkezdeni a küldést.

26.6. A soros vonali konzol beállítása

Készítette: Kazutaka YOKOTA. Az alapján szolgáló írást készítette: Bill Paul.

26.6.1. Bevezetés

A FreeBSD képes úgy is elindulni, ha konzolként mindössze egy buta terminált kapcsolunk rá soros porton keresztül. Az ilyen típusú konfigurációs alapvetően két típus számára bizonyul hasznosnak: azon rendszergazdák számára, akik billentyűzettel és monitorral nem rendelkező gépekre akarnak FreeBSD-t telepíteni, és olyan fejlesztők számára, akik a rendszermag vagy különböző eszközmeghajtók működését akarják nyomon követni.

Ahogy arról már a 12 fejezetben is szó esett, a FreeBSD három indítási fokozattal rendelkezik. Az első két fokozat a rendszerindító blokk kódjában foglal helyet, amely pedig a lemezen található FreeBSD slice elején. A rendszer indulásakor ez a blokk betöltődik és lefuttatja a harmadik fokozatot képviselő rendszertöltőt (a `/boot/loader` állományt).

Ha sörös vonali konzol beállításához tehát be kell állítanunk a rendszerindító blokkot, a rendszertöltőt és a rendszermagot.

26.6.2. A sörös konzol beállítása, rövidített változat

Ebben a szakaszban azt feltételezzük, hogy az alap beállításokkal dolgozunk és csupán egy gyors áttekintésre van szükségünk a sörös vonali konzolról.

1. Csatlakoztassunk egy sörös kábelt a COM1 portra és a terminálra.
2. Rendszeradminisztrátorként a következő parancs kell kiadnunk ahhoz, hogy a sörös konzolon láthassuk az összes rendszerindításhoz tartozó üzenetet:

```
# echo 'console="comconsole"' >> /boot/loader.conf
```
3. Nyissuk meg az `/etc/ttys` állományt, és a `ttvd0` eszközhöz tartozó sorban írjuk át az `off` paramétert az `on` értékre és a `dialup` paramétert a `vt100` értékre. Ha nem ezeket állítjuk be, akkor a sörös konzol keresztül jelszó megadása nélkül is be tudunk jelentkezni, ami viszont egy biztonsági rés veszélyével fenyeget.
4. A változtatások érvényesítéséhez indítsuk újra a rendszerünket.

Ha ettől eltérő beállításokra lenne szükségünk, akkor a folyamat egyes lépéseibe a 26.6.3 Szakaszban kaphatunk mélyebb betekintést.

26.6.3. A sörös vonali konzol beállítása

1. Készítsük elő a sörös kábelt.

Vagy a null-modem kábelre vagy pedig egy szabványos sörös kábelre és egy null-modem átalakítóra lesz szükségünk. A sörös kábelekkel kapcsolatosan a 26.2.2 Szakaszt érdemes elolvasni.

2. Húzzuk ki a billentyűzetet.

A legtöbb személyi számítógép az indítása (vagyis a Power-On Self-Test, POST) során hibát jelez, ha nem érzékel billentyűzetet. Egyes gépek hangosan panaszoľják a billentyűzet hiányát, és nem is hajlandóak egészen addig elindulni, amíg nem csatlakoztatunk egyet.

Ha a számítógépünk hibát küld, de ennek ellenére mégis elindul, akkor semmit nem kell csinálnunk. (Némelyik Phoenix BIOS-os gépen ilyenkor megjelenik a `Keyboard failed` hibaüzenet, de ettől még rendesen elindul a gép.)

Amennyiben a számítógépünk nem hajlandó billentyűzet nélkül elindulni, állítsuk be a BIOS-ban a "hiba" figyelmen kívül hagyását (már ha ez lehetséges). Az alaplap leírásában találhatjuk meg ennek pontos részleít.

Tipp: A BIOS paraméterei között a billentyűzetet állítsuk "Not installed" állapotúra. Ilyenkor még továbbra is használható a billentyűzet, ezzel mindössze csak a BIOS számára tiltjuk le az indításkori ellenőrzést, ezért

nem fog panaszkodni a hiánya miatt. Tehát a billentyűzetet még a “Not installed” beállítása esetén is nyugodtan csatlakoztatjuk, mert működni fog.

Megjegyzés: Ha a rendszerünkön PS/2®-es egér is található, akkor jó eséllyel a billentyűzettel együtt az egeret is ki tudjuk húzni. Mivel a PS/2-es egér osztozik a billentyűzettel bizonyos hardvereken, ezért ha nem húzzuk ki az egeret is, akkor az alaplap még továbbra is képes azt gondolni, hogy a billentyűzet ott van. Például az AMI BIOS-os Gateway 2000-as 90 MHz-es Pentium rendszer pontosan így működik. Általában véve azonban ez nem szokott gondot okozni, mivel az egér billentyűzet nélkül úgy sem ér túlságosan sokat.

3. Csatlakoztassunk egy buta terminált a COM1 (sio0) portra.

Ha nem rendelkezünk buta terminállal, akkor erre célra ugyanúgy alkalmas egy régi XT-s PC valamilyen modemprogrammal vagy egy soros porton csatlakozó másik UNIX-os gép. Ha nincs COM1 (sio0) portunk, akkor szerezzünk egyet. Jelen pillanatban a rendszerindító blokk újrafordítása nélkül a COM1 porton kívül nem tudunk másikat választani. Ha a COM1 portra már raktunk valamilyen másik eszközt, akkor azt ideiglenesen húzzuk le, majd a FreeBSD telepítése és elindítása után tegyünk fel egy másik rendszerindító blokkot. (Egyébként feltételezzük, hogy a COM1 elérhető egy állomány/számító/terminálszerveren — ha valóban valamilyen másik célra szükségünk lenne a COM1 portra (és semmiképpen sem tudjuk átrakni a COM2 (sio1) portra), akkor valószínűleg nem is ezzel kellene elsőként foglalkoznunk.)

4. Gondoskodjunk róla, hogy a rendszermag beállításait tartalmazó állományban a COM1 (sio0) eszközhöz megadtuk a megfelelő paramétereket.

Ezek az alábbiak:

0x10

A konzolos működési mód engedélyezése az adott egységhez. Ha megadjuk ezt a paramétert, akkor a többi a rendszer figyelmen kívül hagyja. Pillanatnyilag legfeljebb egy egység birtokolhatja ezt a beállítást. Ha több ilyen adtunk volna meg, akkor (a felírás sorrendje szerint) az első kap ilyen szerepet. Ez a beállítás önmagában még nem teszi a soros portot konzollá. Ehhez még szükségünk van a következő beállításra, vagy a -h megadására is.

0x20

Az egység konzollá nyilvánítása (hacsak nincs egy tőle nagyobb prioritású konzol), függetlenül a lentebb ismertetendő -h opciótól. A 0x20 értéket a 0x10 értékkel együtt kell megadni.

0x40

(A 0x10 értékkel együtt) az egységet kivonja a normális elérés alól. Ezt a beállítást ne használjuk, ha soros vonali konzolt akarunk üzemeltetni az adott porton. Ezzel az egységet csak a rendszermag távoli nyomkövetéséhez tudjuk használni. A távoli nyomkövetésről a fejlesztők kézikönyvében (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/developers-handbook/index.html) olvastunk bővebben.

Példa:

```
device sio0 at isa? port IO_COM1 flags 0x10 irq 4
```

A további részletekről a sio(4) man oldal tud felvilágosítást nyújtani.

Ha nem állítottuk be a megfelelő paramétereket, akkor (egy másik konzolon) futtassuk a UserConfig programot vagy fordítsuk újra a rendszermagot.

5. Hozzunk létre egy `boot.config` állományt a rendszer indításához használt meghajtó a partíciójának gyökerében.

Ez az állomány mondja meg a rendszerindító blokkban található kódnak, hogy miként akarjuk indítani a rendszerünket. A sörös vonali konzol életrekeltéséhez a most következő opciók közül kell megadnunk egyet vagy többet — amennyiben többet akarunk megadni, akkor mindegyiket egyetlen sorban szerepeltessük:

-h

A belső és a sörös vonali konzolok közti átkapcsolás. Ezzel tudunk a konzolos eszközök között váltani. Például, ha egy belső (video) konzolról indítjuk a rendszert, akkor a rendszertöltőnek és a rendszermagnak átadott -h paraméterrel arra tudjuk ezeket utasítani, hogy konzolként a sörös portot használják. Vagy ha sörös porton keresztül indítjuk a rendszert, akkor -h megadásával megkérhetjük a rendszertöltőt és a rendszermagot, hogy ezután már a videokártyát használja konzolként.

-D

Az egy- és kétkonzolos beállítások közti váltás. Az egykonzolos konfigurációban a konzol lehet belső (video) vagy sörös vonali, attól függően, hogy miként használtuk a fenti -h opciót. A kétkonzolos konfigurációban azonban a videokártyán és a sörös vonalon keresztül is egyszerre megjelenik a konzol, függetlenül a -h hatásától. Ilyenkor viszont vegyük figyelembe, hogy ez a kétkonzolos konfiguráció csak a rendszerindító blokk futása alatt él. Amint a rendszerindító megkapja a vezérlést, a -h által megadott konzol válik az egyedülivé.

-P

A rendszerindító blokk megpróbálja megkeresni a billentyűzetet. Ha nem találja, akkor magától beállítja a -D és -h opciókat.

Megjegyzés: Tárbeli korlátozások miatt a rendszerindító blokk jelenlegi változata a -P paraméterrel csak a kiterjesztett billentyűzeteket képes kezelni. A 101 gombnál kevesebbel (tehát F11 és F12 gombokkal nem) rendelkező billentyűzeteket ezért nem feltétlenül fogja észlelni. Ugyanezen korlátozás miatt egyes laptopokon sem minden esetben sikerül érzékelni a billentyűzetet. Ha ez a rendszerünkön problémához vezetne, akkor egyszerűbb lesz elhagyni a -P használatát. Sajnos, jelenleg semmilyen megoldás nincs erre.

Vagy a -P opcióval állítassuk be automatikusan a konzolt, vagy pedig a -h opcióval engedélyezzük a sörös vonali konzolt.

Természetesen itt a `boot(8)` man oldalon szereplő összes többi paramétert is megadhatjuk.

A -P kivételével az összes opció a rendszertöltőnek (`/boot/loader`) kerül átadásra. A rendszertöltő egyedül a -h állapotából dönti el, hogy mely belső videoeszközön vagy sörös porton legyen a konzol. Ez azt jelenti, hogy a `/boot.config` állományban ha megadjuk a -D opciót, de mellette nem szerepel a -h, akkor a sörös vonali konzolt csak a rendszerindító blokk futása alatt tudjuk elérni — a rendszertöltő ugyanis alapból a videokártyát használja konzolként.

6. Kapcsoljuk be a számítógépünket.

Amikor elindítjuk a FreeBSD-s gépünket, a rendszerindító blokk kiírja a `/boot.config` tartalmát a konzolra. Például így:

```
/boot.config: -P
Keyboard: no
```

A második sor csak olyankor jelenik meg, ha a `/boot.config` állományban a `-P` beállítás is szerepel, és a billentyűzet jelenlétét (yes) vagy hiányát (no) jelzi. A `/boot.config` tartalmától függően ezek az üzenetek vagy a soros vonali vagy a belső konzolon jelennek meg, esetleg mind a kettőn.

Beállítás	Ahol megjelenik
nincs	belső konzol
-h	soros vonali konzol
-D	soros vonali és belső konzol
-Dh	soros vonali és belső konzol
-P, van billentyűzet	belső konzol
-P, nincs billentyűzet	soros vonali konzol

Az iménti üzenetek felbukkanása után a további konzolos üzenetek küldésében egy rövid szünet következik, amíg a rendszerindító blokk a rendszertöltő betöltésével folytatja a rendszer indítását. Normális körülmények között ezt a folyamatot nem kell megszakítanunk, de esetleg olyankor mégis érdemes lehet, ha le akarjuk ellenőrizni a beállításainkat.

A rendszerindítási folyamat félbeszakításához az **Enter** billentyűn kívül nyomjuk le valamelyik másikat. Ekkor a rendszerindító blokk megáll és várja a további parancsokat. Ekkor valami ilyesmit láthatunk:

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

Nézzük meg, hogy `/boot.config` beállításainak megfelelően a fenti üzenet a soros vonali konzolon vagy a belső konzolon, illetve mind a kettőn megjelenik-e. Ha az üzenet a megfelelő konzolon megjelenik, akkor az **Enter** lenyomásával folytathatjuk a rendszer indítását.

Ha nekünk a soros vonali konzolra lenne szükségünk, de semmi nem jelenik meg a soros terminálon, akkor valamit valószínűleg nem jól állítottunk be. A rendszerindító bloktól kapott parancssorban a `-h` begépelésével és az **Enter** vagy **Return** lenyomásával (ha lehetséges) jelezzük neki (és így a rendszertöltőnek és a rendszermagnak is) a soros vonali konzol kiválasztását. Miután befejeződött a rendszer indítása, menjünk vissza és ellenőrizzük a megfelelő paramétereket.

Ahogy sikerült elindítani a rendszertöltőt és a rendszerindítás harmadik fokozatába léptünk, a rendszertöltő megfelelő környezeti változóin keresztül még mindig van lehetőségünk váltani a soros vonali és a belső konzol között, lásd 26.6.6 Szakasz.

26.6.4. Összefoglalás

Itt most röviden összefoglaljuk az eddig tárgyalt különböző beállításokat és ténylegesen kiválasztott konzolt.

26.6.4.1. 1. eset: a `sio0` eszköznél a `0x10` beállítást adjuk meg

```
device sio0 at isa? port IO_COM1 flags 0x10 irq 4
```

A /boot.config beállításai	Konzol a rendszerindító blokk alatt	Konzol a rendszertöltő alatt	Konzol a rendszermagban
nincsenek	belső	belső	belső
-h	soros vonali	soros vonali	soros vonali
-D	soros vonali és belső	belső	belső
-Dh	soros vonali és belső	soros vonali	soros vonali
-P, van billentyűzet	belső	belső	belső
-P, nincs billentyűzet	soros vonali és belső	soros vonali	soros vonali

26.6.4.2. 2. eset: a sio0 eszköznél 0x30 beállítása

```
device sio0 at isa? port IO_COM1 flags 0x30 irq 4
```

A /boot.config beállításai	Konzol a rendszerindító blokk alatt	Konzol a rendszertöltő alatt	Konzol a rendszermagban
nincsenek	belső	belső	soros vonali
-h	soros vonali	soros vonali	soros vonali
-D	soros vonali és belső	belső	soros vonali
-Dh	soros vonali és belső	soros vonali	soros vonali
-P, van billentyűzet	belső	belső	soros vonali
-P, nincs billentyűzet	soros vonali és belső	soros vonali	soros vonali

26.6.5. Tanácsok a soros vonali konzol használatához

26.6.5.1. Nagyobb soros vonali sebesség beállítása

A soros port alapértelmezései a következők: 9600 baud, 8 bites átvitel, paritás nincs és 1 stopbit. Ha a konzol alapsebességét meg akarjuk változtatni, akkor ahhoz a következőket kell tennünk:

- Fordítsuk újra a rendszerindító blokkokat úgy, hogy a `BOOT_COMCONSOLE_SPEED` változóban a konzolnak egy másik sebességet adjunk meg. Az új rendszerindító blokkok fordításáról és telepítéséről a 26.6.5.2 Szakaszban kapunk részletes leírást.

Ha a soros vonali konzolt nem a `-h` opcióval állítottuk be, vagy ha a rendszermag a rendszerindító blokkoktól eltérő módon éri el a soros vonali konzolt, akkor a rendszermag beállításai közé még az alábbi is fel kell vennünk, majd újra kell fordítanunk:

```
options CONSPEED=19200
```

- A rendszermagban adjuk át a `-S` rendszerindítási paramétert. A `-S` parancssori opció a `/boot.config` állományban is megadható. A boot(8) man oldalon tudhatjuk meg, hogy a `/boot.config` beállításai közé hogyan tudjuk felvenni és ott milyen további lehetőségeink vannak még.

- A `/boot/loader.conf` állományban engedélyezzük a `comconsole_speed` beállítást.

Ez a beállítás a szintén a `/boot/loader.conf` állományban megadható `console`, `boot_serial` és `boot_multicons` változóktól függ. A soros vonali konzol sebességét tehát például így tudjuk megváltoztatni a `comconsole_speed` megadásával:

```
boot_multicons="YES"
boot_serial="YES"
comconsole_speed="115200"
console="comconsole,vidconsole"
```

26.6.5.2. Soros vonali konzol a `sio0` porton kívül máshol

Ha valamilyen okból kifolyólag nem a `sio0` porton keresztül akarjuk használni a konzolt, akkor ahhoz a rendszerindító blokkok, a rendszertöltő és a rendszermag forrásait újra kell fordítanunk az alábbiak szerint:

1. Szerezzük be a rendszermag forrását. (Lásd 24 fejezet)
2. Írjuk át a `/etc/make.conf` állományban a `BOOT_COMCONSOLE_PORT` címét az általunk használt porthoz tartozóéra (`0x3F8`, `0x2F8`, `0x3E8` vagy `0x2E8`). Itt csak a `sio0` és `sio3` (`COM1` és `COM4`) közti portok használhatóak — a töbportos soros kártyák címei nem adhatóak meg. A megszakításokat nem kell beállítanunk.
3. Készítsünk egy saját rendszermag beállításait tartalmazó állományt, és vegyük fel bele a használni kívánt soros port megfelelő paramétereit. Például, ha a `sio1` (`COM2`) eszközt akarjuk konzolként használni:

```
device siol at isa? port IO_COM2 flags 0x10 irq 3
```

vagy

```
device siol at isa? port IO_COM2 flags 0x30 irq 3
```

A konzolra vonatkozó beállításokat a többi soros portnál ne adjuk meg.

4. Fordítsuk újra és telepítsük a rendszerindító blokkot és a rendszertöltőt:

```
# cd /sys/boot
# make clean
# make
# make install
```

5. Fordítsuk és telepítsük újra a rendszermagot.
6. A `bsdlabel(8)` segítségével másoljuk az új rendszerindító blokkot a rendszer indítását végző lemezre és töltsük be az új rendszermagot.

26.6.5.3. A DDB elérése a soros vonalról

Ha a soros vonali konzolról akarjuk használni a rendszermagba épített nyomkövetőt (ami hasznos lehet távoli vizsgálódáskor, de egyben veszélyes is, ha a soros porton tévesen kiküldünk egy `BREAK` jelzést!), akkor a rendszermagot a következő beállításokkal kell fordítanunk:

```
options BREAK_TO_DEBUGGER
options DDB
```

26.6.5.4. A bejelentkező képernyő elérése a soros vonali konzolról

Habár erre nincs feltétlenül szükségünk, a rendszer üzeneteinek és a rendszermag nyomkövetőjének elérése után akár *be is tudunk jelentkezni* a soros vonalon keresztül. Íme!

Nyissuk meg az `/etc/ttys` állományt a kedvenc szövegszerkesztőnkkel és keressük meg a következő sorokat:

```
ttyd0 "/usr/libexec/getty std.9600" unknown off secure
ttyd1 "/usr/libexec/getty std.9600" unknown off secure
ttyd2 "/usr/libexec/getty std.9600" unknown off secure
ttyd3 "/usr/libexec/getty std.9600" unknown off secure
```

A `ttyd0` és `ttyd3` közti sorok pontosan a `COM1` és `COM4` közti portoknak felelnek meg. A használni kívánt port sorában szereplő `off` paramétert írjuk át az `on` értékre. Ha a soros port sebességét is megváltoztattuk, minden bizonnyal a `std.9600` helyett is az adott sebességhez illeszkedő paramétert kell megadnunk, például az `std.19200` értékkel.

Érdemes továbbá még az `unknown` helyett megadni az adott terminál típusát.

Az állomány módosítását követően a változtatások érvényesítéséhez ki kell adnunk a `kill -HUP 1` parancsot is.

26.6.6. A konzol megváltoztatása a rendszertöltőből

A korábbi szakaszokban arról beszéltünk, hogy miként állítsuk be a soros vonali konzolt a rendszerindító blokk megpiszkálásával. Ebben a szakaszban viszont azt mutatjuk meg, hogy különböző parancsokon és környezeti változókon keresztül miként tudjuk megadni a konzolt a rendszertöltőben. Mivel a rendszertöltőre a rendszerindítás harmadik fokozatában kerül sor, az ott megadott értékekkel felül tudjuk bírálni a rendszerindító blokk beállításait.

26.6.6.1. A soros vonali konzol beállítása

A rendszertöltő és a rendszermag az `/boot/loader.conf` állományon keresztül elég könnyen rávehető a soros vonali konzol használatára:

```
set console="comconsole"
```

Ez a rendszerindító blokk előző szakaszban tárgyalt beállításaitól függetlenül érvényesül.

A fenti sort a `/boot/loader.conf` állomány elejére érdemes tennünk, így a soros vonali konzolon már a lehető leghamarabb megjelennek a rendszer üzenetei.

Ehhez hasonló módon a belső konzolt is megadhatjuk:

```
set console="vidconsole"
```

Ha a rendszertöltőben nem adjuk meg a `console` környezeti változó értékét, akkor a rendszertöltő, és így a rendszermag is, a rendszerindító blokkban a `-h` opció által meghatározott konzolt fogja használni.

A konzol a `/boot/loader.conf.local` vagy a `/boot/loader.conf` állományokban adható meg.

A részletekkel kapcsolatban lásd a `loader.conf(5)` man oldalt.

Megjegyzés: Jelen pillanatban a rendszertöltőnek nincs a `-p` paraméterrel ekvivalens értékű beállítása, ezért a billentyűzet jelenléte alapján nem képes magától választani a belső és a soros vonali konzol között.

26.6.6.2. Soros vonali konzol a `si00` porton kívül máshol

A rendszertöltőt ne a `si00` eszközzel fordítsuk újra a soros vonali konzolhoz. Ehhez kövessük a 26.6.5.2 Szakaszban leírt eljárás lépéseit.

26.6.7. Figyelmeztetések

A szakaszban szereplő ötletek alapján sokan így most már könnyen be tudnak állítani egy billentyűzet és grafikus hardver nélküli dedikált szerveret. Sajnos azonban a legtöbb rendszer nem engedi a billentyűzet nélküli indítást, és akad néhány olyan is, amely pedig a grafikus kártya hiányában nem is indul el. Az AMI BIOS-os gépeknél a grafikus kártya nélküli indításhoz elegendő csupán a beállítások között a grafikus kártyát (“graphics adapter”) “Not installed” (nem telepített) állapotúra állítani. Ha ilyen opció nem található a BIOS-ban, akkor helyette keressük a “Halt on Error” (leállás hiba esetén) változatot. Ha ezt a “All but Keyboard” (a billentyűzet kivételével minden) vagy akár a “No Error” (soha) értékre állítjuk, az előbbi eredményt kapjuk.

Ennek ellenére előfordulhat azonban, hogy egyes gépeken egyáltalán nem találunk ilyen lehetőséget és videokártya nélkül nem indulnak el. Ezekben az esetekben tegyünk a gépbe valamilyen kártyát (ehhez elég egy egyszerű típus is), de monitort már ne kössünk rá. Esetleg megpróbálkozhatunk még AMI BIOS telepítésével is.

Fejezet 27. A PPP és a SLIP

Átdolgozta, átrendezte és aktualizálta: Jim Mock.

27.1. Áttekintés

A FreeBSD számos módon képes összekötni két számítógépet. Ha betárcsázós modemmel akarunk hálózati vagy internetes kapcsolatot felépíteni, esetleg azt szeretnénk, hogy mások képesek legyenek minket ilyen módon elérni, akkor ahhoz PPP-t, illetve SLIP-et kell használnunk. Ebben a fejezetben a modem kommunikáció beállításait mutatjuk be részletesebben.

A fejezet elolvasása során megismerjük:

- hogyan állítsunk be felhasználói PPP-t;
- hogyan állítsunk be rendszerszintű PPP-t (csak FreeBSD 7.x);
- hogyan állítsunk be egy PPPoE (PPP over Ethernet, vagyis “PPP Ethernet felett”) kapcsolatot;
- hogyan állítsunk be egy PPPoA (PPP over ATM, vagyis “PPP ATM felett”) kapcsolatot;
- hogyan állítsunk be SLIP klienst és szerveret (csak FreeBSD 7.x).

A fejezet elolvasásához ajánlott:

- az alapvető hálózati technológiák ismerete;
- a betárcsázós kapcsolatok, a PPP és/vagy SLIP alapjainak és céljainak megértése.

Talán érdekli a kedves olvasót, hogy mi az alapvető különbség a felhasználói és a rendszerszintű PPP között. A válasz egyszerű: a felhasználói PPP a beérkező és kimenő adatokat nem a rendszermagban, hanem a felhasználói szinten dolgozza fel. Ez költséges abból a szempontból, hogy emiatt adatokat kell másolgatni a rendszer és a felhasználói szint között, azonban egy sokkal többet tudó PPP implementációnak ad ezzel utat. A felhasználói PPP a tun eszközzön keresztül kommunikál a külvilággal, miközben a rendszermagban található PPP mindezt a `ppp` eszközzel valósítja meg.

Megjegyzés: A fejezetben a felhasználói PPP-t egyszerűen csak **ppp** néven fogjuk hivatkozni, hacsak nem lesz szükséges különbséget tennünk közte és más PPP szoftverek, mint például a **pppd** között. Ha mást nem mondunk, akkor a fejezetben ismertetett összes parancsot `root` felhasználóként kell kiadni.

27.2. A felhasználói PPP alkalmazása

Frissítette és javította: Tom Rhodes. Eredetileg készítette: Brian Somers. Segített még: Nik Clayton, Dirk Fröberg, és Peter Childs.

Figyelem A FreeBSD 8.0 változatától kezdődően a soros portokhoz tartozó eszközök nevei `/dev/cuaN` helyett `/dev/cuauN`, illetve `/dev/ttydN` helyett `/dev/ttyuN` lettek. A FreeBSD 7.x felhasználóknak ezeknek a változásoknak megfelelően kell olvasniuk az itt szereplő dokumentációt.

27.2.1. A felhasználói PPP

27.2.1.1. Előfeltételek

A leírás feltételezi, hogy rendelkezünk a következőkkel:

- Olyan internet-előfizetés, ahol PPP-n keresztül csatlakozunk
- Egy modem vagy más olyan rendszerünkhöz csatlakozó eszköz, amelyen keresztül el tudjuk érni az internet-szolgáltatónkat
- Az internet-előfizetés betárcsázásához szükséges telefonszámok
- A bejelentkezési nevünk és jelszavunk. (Vagy a megszokott UNIX-os felhasználói név és jelszó páros, vagy egy PAP esetleg CHAP bejelentkezési név és jelszó.)
- Egy vagy több névszerver IP-címe. Ehhez az internet-szolgáltatók általában két IP-címet adnak meg. Ha egyet sem kaptunk, akkor a `ppp.conf` állományban erre a célra használhatjuk az `enable dns` parancsot, és ekkor a **ppp** majd automatikusan be fogja állítani nekünk a névszervereket. Ezt a lehetőséget az befolyásolja, hogy az internet-szolgáltató oldalán működő PPP implementáció támogatja-e a névfeloldás egyeztetését (DNS negotiation).

A következő információkat is megkaphatjuk az internet-előfizetésünkhöz, de nem feltétlenül szükségesek:

- Az internet-szolgáltató átjárójának IP-címe. Az átjáró az a gép, amelyen keresztül a gépünk csatlakozik és számára ez lesz az *alapértelmezett átjáró*. Ha nem rendelkezünk ezzel az információval, akkor csak állítsunk be valamit, és majd a csatlakozáskor a szolgáltató PPP szervere felülírja a megfelelő beállításokkal.

Erre a címre a **ppp** HISADDR néven hivatkozik.

- A használandó hálózati maszk. Amennyiben a szolgáltató ezt nem adta meg, nyugodtan használjuk erre a `255.255.255.255` értéket.
- Ha a szolgáltatónk statikus IP-címet és rögzített hálózati nevet is biztosít nekünk, ezt is megadhatjuk. Minden más esetben egyszerűen csak hagyjuk, hogy a rendszer automatikusan válasszon nekünk egyet.

Ha a szükséges információknak nem vagyunk birtokában, akkor vegyük fel a kapcsolatot az internet-szolgáltatókkal.

Megjegyzés: Ebben a szakaszban a példákban szereplő konfigurációs állományok sorait számozva láthatjuk. Ezek a sorszárok a bemutatás és a tárgyalás megkönnyítése érdekében szerepelnek, és nem az eredeti állományok részei. Mindezek mellett a tabulátorok és szóközök megfelelő használata is fontos.

27.2.1.2. A PPP automatikus beállítása

A `ppp` és a `pppd` (a PPP rendszerszintű megvalósítása) egyaránt az `/etc/ppp` könyvtárban található konfigurációs állományokat használja. A felhasználói PPP-hez ezenkívül még a `/usr/share/examples/ppp/` könyvtárban vannak példák.

A ppp parancs beállítása az igényeinktől függően számos állomány módosítását igényelheti. A tartalmukat nagyban befolyásolja, hogy a szolgáltatónk részéről a címeket kiosztása statikus (vagyis egy adott címet kapunk és folyamatosan azt használjuk) esetleg dinamikus (vagyis az IP-címünk minden egyes kapcsolódáskor más és más).

27.2.1.2.1. PPP statikus IP-címmel

Ebben az esetben az `/etc/ppp/ppp.conf` konfigurációs állományt kell átszerkesztenünk. Tartalma az alábbi példához hasonlítható.

Megjegyzés: A : karakterrel végződő sorok mindig az első oszlopban kezdődnek (tehát a sor elején), míg az összes többi sort tabulátorok vagy szóközök használatával bentebb kell raknunk.

```

1      default:
2          set log Phase Chat LCP IPCP CCP tun command
3          ident user-ppp VERSION (built COMPILATIONDATE)
4          set device /dev/cuau0
5          set speed 115200
6          set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \
7                  \"\" AT OK-AT-OK ATE1Q0 OK \\dATDT\\T TIMEOUT 40 CONNECT"
8          set timeout 180
9          enable dns
10
11     szolgáltato:
12         set phone "(123) 456 7890"
13         set authname ize
14         set authkey mize
15         set login "TIMEOUT 10 \"\" \"\" gin:--gin: \\U word: \\P col: ppp"
16         set timeout 300
17         set ifaddr x.x.x.x y.y.y.y 255.255.255.255 0.0.0.0
18         add default HISADDR

```

1. sor:

Ez azonosítja be az alapértelmezett bejegyzést. Az itt szereplő parancsok a ppp minden egyes futásakor magukból végrehajtódnak.

2. sor:

Beállítja a naplózás paramétereit. Amikor a beállításaink már kifogástalanul működnek, akkor ezt a sort érdemes átírni a következőre:

```
set log phase tun
```

Ezzel jelentős mértékben vissza tudjuk fogni a naplózás mértékét.

3. sor:

Ezzel mondjuk meg a PPP-nek, hogy a többiek felé miként azonosítsa magát. A PPP akkor azonosítja magát a társak felé, ha valamilyen gondja akad az egyeztetésekkel és a kapcsolat beállításával. Az így továbbított információk a másik oldal rendszergazdái számára nyújthatnak segítséget az ilyen jellegű problémák felderítésében.

4. sor:

Itt adjuk meg az eszközt, amelyre a modem csatlakozik. A COM1 neve `/dev/cuau0`, a COM2 neve pedig `/dev/cuau1`.

5. sor:

A csatlakozás sebességét adjuk meg. Ha a 115 200-as érték itt nem működne (ez egyébként minden újabb gyártmányú modem esetében elfogadható), akkor helyette használjuk a 38400-as beállítást.

6. és 7. sorok:

A híváshoz használt karakterlánc. A felhasználói PPP a `chat(8)` programhoz hasonló “küldök-várok” típusú szerkesztést alkalmaz. A kihasználható lehetőségekről a man oldalán olvashatunk részletesebben.

Az olvashatóság kedvéért a parancs a következő sorban folytatódik. A `ppp.conf` állományban bármelyik parancs, ahol a `\` karakterrel zárjuk a sort, az ugyanígy folytatható a következőben.

8. sor:

A kapcsolathoz tartozó üresjáratit időt állítja be. Ennek értéke alaphoz 180 másodperc, így ez a sor pusztán csak az érthetőséget szolgálja.

9. sor:

Arra utasítja a PPP-t, hogy a többiektől kérdezze le a helyi névfeloldó beállításait. Ha saját névszervert futtatunk, akkor ezt a sort tegyük inkább megjegyzésbe vagy töröljük ki.

10. sor:

Ez az üres sor az átláthatóság kedvéért került bele. A PPP az összes üres sort figyelmen kívül hagyja.

11. sor:

Itt kezdődik a “szolgáltató” nevű szolgáltatóhoz tartozó bejegyzés. Ezt később akár ki is cserélhetjük az internet-szolgáltatónk nevére, így a `load szolgáltató` beállítással tudjuk majd beindítani a kapcsolatot.

12. sor:

Beállítjuk a szolgáltatóhoz tartozó telefonszámot. A kettőspont (`:`) vagy a csővezeték (`|`) karakterekkel elválasztva több telefonszámot is meg tudunk adni. A `ppp(8)` oldalon olvashatunk a két elválasztó közti különbségekről. Röviden ezeket úgy foglalhatnánk össze, hogy ha váltogatni akarunk a számok között, akkor használjuk a kettőspontot. Ha mindig az elsőként megadott számot akarjuk hívni és a többit csak akkor, ha ez nem működik, akkor a csővezeték karakterre lesz szükségünk. Ahogy a példa is mutatja, az összes telefonszámot tegyük mindig idézőjelek közé.

Ha a telefonszámban egyébként is szerepelnek szóközök, akkor is idézőjelek (`"`) közé kell tennünk. Ennek elhagyásával egy egyszerű, ámde kényes hibát ejtünk.

13. és 14. sor:

A felhasználói nevet és jelszót tartalmazza. Amikor egy UNIX fajtájú bejelentkezést kapunk, akkor ezekre az értékekre a `set login` parancsban `\U` és `\P` változókkal tudunk hivatkozni. Ha PAP vagy CHAP használatával jelentkezünk be, akkor ezek az értékek a hitelesítéskor kerülnek felhasználásra.

15. sor:

Ha a PAP vagy CHAP protokollok valamelyikét használjuk, akkor nem lesz szükségünk a login változóra, ezért ezt megjegyzésbe is tehetjük, vagy akár ki is törölhetjük. A PAP és CHAP hitelesítésről szóló részben olvashatjuk ennek további részleteit.

A bejelentkezéshez használt karakterlánc hasonlít a behíváshoz használt, chat-szerű felépítéssel rendelkező karakterláncához. A példában látható karakterlánc egy olyan szolgáltatáshoz illeszkedik, ahol a bejelentkezés valahogy így néz ki:

```
A Világ Legjobb Szolgáltatója
login: izé
password: mizé
protocol: ppp
```

Ezt a szkriptet alakítsuk a saját igényeinkhez. Ha először próbálkozunk ilyen szkript írásával, akkor lehetőleg kapcsoljuk be a rendszerek között lezajló “beszélgetés” naplózását, hogy ellenőrizni tudjuk minden a megfelelően módon történik-e.

16. sor:

Beállítjuk a kapcsolathoz tartozó alapértelmezett időkorlátot (másodpercben). Itt a kapcsolat automatikusan lezárul 300 másodperc tétlenséget követően. Ha nem akarunk ilyen korlátot szabni, akkor ezt az értéket állítsuk nullára vagy használjuk a `-ddial` parancssori kapcsolót.

17. sor:

A felülethez tartozó címeket állítja be. A `x.x.x.x` helyére a szolgáltató által kiosztott IP-címet kell beírunk. A `y.y.y.y` helyett pedig a szolgáltató átjárója kerül be (lényegében az a gép, amelyhez csatlakozunk). Amennyiben az internet-szolgáltatónk nem adott meg semmilyen átjárót, erre a célra a `10.0.0.2/0` cím is használhatjuk. Amikor “nekünk kell kitalálnunk” ezeket a címeket, akkor ne felejtünk el létrehozni hozzájuk egy bejegyzést az `/etc/ppp/ppp.linkup` állományban a PPP dinamikus IP-címmel szakaszban szereplők szerint. Ha nem adjuk meg ezt a sort, akkor a `ppp` parancs nem képes `-auto` módban működni.

18. sor:

A szolgáltató átjárójához felvesz egy alapértelmezett útvonalat. A `HISADDR` kulcsszót a 17. sorban megadott átjáró címével helyettesítjük. Ezért fontos, hogy ez a 17. sor után szerepeljen, különben a `HISADDR` nem lesz képes inicializálódni.

Ha a `ppp` parancsot nem akarjuk `-auto` módban futtatni, akkor ezt a sort a `ppp.linkup` állományba is átrakhatjuk.

Ha statikus IP-címmel rendelkezünk és a `ppp -auto` módban fut, akkor a `ppp.linkup` állományba egészen addig nem kell semmit sem írunk, amíg a csatlakozás előtt az útválasztási táblázatokban a megfelelő adatok találhatók. Olyankor is jól jöhet, amikor a csatlakozást követően meg akarunk hívni bizonyos programokat. Ezt majd a **sendmailes** példában fogjuk bővebben kifejteni.

Erre példákat a `/usr/share/examples/ppp/` könyvtárban találhatunk.

27.2.1.2.2. PPP dinamikus IP-címmel

Ha az internet-szolgáltatónktól nem kaptunk statikus IP-címet, akkor a `ppp` parancssal is be tudjuk állítani a helyi és távoli címeket. Ez az IP-címek “kitalálásával” történik, valamint úgy, hogy a `ppp` számára a csatlakozás után

lehetővé tesszük az IP konfigurációs protocol (IP Configuration Protocol, IPCP) használatát. A `ppp.conf` tartalma szinte teljesen megegyezik a PPP statikus IP-címmel részben szereplővel, egyetlen apró különbséggel:

```
17      set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.255
```

Ismét szeretnénk elmondani, hogy a sorszámot ne írjuk bele, hiszen az csak hivatkozási céllal szerepel. Legalább egy szóközzel kezdjük bentebb.

17. sor:

A / után megjelenő szám azoknak a biteknek a számát adja meg, amire a ppp támaszkodik. A környezetünknek jobban megfelelő IP-címeket is megadhatunk, de a fenti példa minden esetben működni fog.

Az utolsó paraméterrel (0.0.0.0) azt mondjuk a PPP-nek, hogy az egyeztetést ne a 10.0.0.1, hanem a 0.0.0.0 címmel kezdje meg, amire egyes szolgáltatók esetén szükségünk is lesz. A `set ifaddr` első paramétereként azonban soha ne adjuk meg a 0.0.0.0 címet, mivel ezzel a PPP -auto módban nem tudja beállítani a kezdeti útvonalat.

Ha nem -auto módban indítjuk, akkor az `/etc/ppp/ppp.linkup` állományban meg kell adnunk még egy bejegyzést is. A `ppp.linkup` állományt a kapcsolat létrejötte után dolgozzuk fel. Itt már a ppp megkapta a felülethez tartozó címeket, így az útválasztási táblázatba fel tudjuk venni hozzájuk a megfelelő bejegyzéseket:

```
1      szolgáltato:
2      add default HISADDR
```

1. sor:

A kapcsolat felépítése során a ppp a `ppp.linkup` állományban a következő szabályok szerint fogja keresni a bejegyzéseket: először a `ppp.conf` állományban megadott címkét próbálja megtalálni. Ha ez nem sikerül, akkor az átjárónknak megfelelő bejegyzést kezdi el keresni. Ez egy négy byte-ból álló, felírásában az IP-címekhez hasonlító címke. Ha még ez a címke sem található, akkor a MYADDR bejegyzést keresi.

2. sor:

Ez a sor mondja meg a ppp programnak, hogy vegyen fel egy HISADDR címre vonatkozó alapértelmezett útvonalat. A HISADDR címet az IPCP által egyeztetett átjáró IP-címére cseréljük ki.

Ha erre a részletesebb példát akarunk látni, akkor a `/usr/share/examples/ppp/ppp.conf.sample` és `/usr/share/examples/ppp/ppp.linkup.sample` állományokban a `pmdemand` bejegyzést nézzük meg.

27.2.1.2.3. A bejövő hívások fogadása

Amikor egy helyi hálózathoz csatlakozó gépen akarjuk a **ppp** programot beállítani a bejövő hívások fogadására, akkor azt is el kell döntenünk, hogy engedélyezzük-e a csomagok továbbküldését a belső hálózat felé. Amennyiben igen, akkor a becsatlakozó gépenek a belső hálózatunkon ki kell osztani egy külön címet és az `/etc/ppp/ppp.conf` állományban, és meg kell adnunk az `enable proxy` parancsot. Emellett még az `/etc/rc.conf` állományban se felejtjük el megadni a következő sort:

```
gateway_enable="YES"
```

27.2.1.2.4. Melyik getty?

A FreeBSD beállítása betárcsázós kapcsolatokhoz nagyon jól bemutatja a betárcsázós szolgáltatások beállítását a `getty(8)` segítségével.

A `getty` helyett egyébként az `mgetty` (<http://mgetty.greenie.net/>), a `getty` egy ügyesebb változata is használható (a `comms/mgetty+sendfax` portból), amely kifejezetten a betárcsázós vonalakhoz készült.

A `mgetty` használatának többek közt az egyik előnye, hogy *aktívan tartja a kapcsolatot* a modemekkel, tehát hogy ha az `/etc/ttys` állományban letiltjuk a modemet, akkor nem is fog válaszolni a hívásokra.

Emellett az `mgetty` későbbi változatai (a 0.99 beta változattól kezdve) még a PPP folyamatok automatikus észlelését is támogatják, ezáltal a kliensek szkriptek nélkül is képesek elérni a szervert.

Ha erről többet akarunk megtudni, akkor az `mgetty` paranccsal kapcsolatban olvassuk el Az `mgetty` és az `AutoPPP` című szakaszt.

27.2.1.2.5. A PPP engedélyei

A `ppp` parancsot általában `root` felhasználóként kell futtatni. Ha viszont a `ppp` parancsot tetszőleges felhasználóval akarjuk server módban futtatni az iméntiek szerint, akkor ahhoz fel kell vennünk az `/etc/group` állományban szereplő `network` csoportba.

Ezekén kívül még az `allow` paranccsal is engedélyeznünk kell konfigurációs állomány egy vagy több részének elérését is:

```
allow users fred mary
```

Ha ezt a parancsot a `default` bejegyzésnél adjuk meg, akkor az így megadott felhasználók mindenhez hozzá tudnak férni.

27.2.1.2.6. PPP shellek a dinamikus IP-címek használóinak

Hozzunk létre egy `/etc/ppp/ppp-shell` nevű állományt, amelyben a következők szerepelnek:

```
#!/bin/sh
IDENT='echo $0 | sed -e 's/^\.*-\(.*\)$/\1/'`
CALLEDAS="$IDENT"
TTY='tty'

if [ x$IDENT = xdialup ]; then
    IDENT='basename $TTY'
fi

echo "PPP for $CALLEDAS on $TTY"
echo "Starting PPP for $IDENT"

exec /usr/sbin/ppp -direct $IDENT
```

Ez a szkript legyen végrehajtható. Ezután az alábbi paranccsal `ppp-dialup` néven készítsünk egy szimbolikus linket erre a szkriptre:

```
# ln -s ppp-shell /etc/ppp/ppp-dialup
```

Ez a szkript lesz az összes betárcsázó felhasználónk *shellje*. A most következő példa az `/etc/passwd` állományban szereplő, `pchlds` nevű PPP felhasználó bejegyzését mutatja be (ne felejtjük el, hogy soha ne közvetlenül szerkesszük a jelszavakat tároló állományt, hanem a `vipw(8)` segítségével).

```
pchlds:*:1011:300:Peter Childs PPP:/home/ppp:/etc/ppp/ppp-dialup
```

Hozzunk létre egy `/home/ppp` nevű könyvtárat a következő bárki által olvasható 0 byte-os állományokkal:

```
-r--r--r-- 1 root wheel 0 May 27 02:23 .hushlogin
-r--r--r-- 1 root wheel 0 May 27 02:22 .rhosts
```

Ezek hatására az `/etc/motd` állomány tartalma nem jelenik meg.

27.2.1.2.7. PPP shellek a statikus IP-címek használóinak

Az iméntiekhez hasonló módon készítsük el a `ppp-shell` állományt, és mindegyik statikus IP-vel rendelkező hozzáféréshez csináljunk egy szimbolikus linket a `ppp-shell` szkriptre.

Például, ha három betárcsázós ügyfelünk van, `fred`, `sam` és `mary`, feléljük 24 bites CIDR hálózatokat közvetítünk, akkor a következőket kell begépelnünk:

```
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-fred
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-sam
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-mary
```

A fentebb szereplő betárcsázós felhasználók elérésiéhez tartozó shelleket állítsuk be az itt létrehozott szimbolikus linkekre (így tehát `mary shellje` az `/etc/ppp/ppp-mary` lesz).

27.2.1.2.8. A `ppp.conf` beállítása a dinamikus IP-címek használóinak

Az `/etc/ppp/ppp.conf` állományban a következő sorok valamelyikének kellene szerepelnie:

```
default:
    set debug phase lcp chat
    set timeout 0

ttyu0:
    set ifaddr 203.14.100.1 203.14.100.20 255.255.255.255
    enable proxy

ttyu1:
    set ifaddr 203.14.100.1 203.14.100.21 255.255.255.255
    enable proxy
```

Megjegyzés: A bentebb kezdett sorokat mi is kezdjük bentebb.

A `default:` szakasz minden kapcsolat esetén betöltődik. Az `/etc/ttys` állományban engedélyezett mindegyik betárcsázós vonal létrehoz a fenti `ttyu0:` szakaszhoz hasonló bejegyzést. Minden vonal kap egy egyedi IP-címet a dinamikus felhasználók számára szánt címtartományból.

27.2.1.2.9. A *ppp.conf* beállítása a statikus IP-vel rendelkezők számára

A `/usr/share/examples/ppp/ppp.conf` állományban szereplő tartalom mellett az összes statikus kiosztású IP-címmel rendelkező betárcsázó felhasználóhoz még hozzá kell tennünk egy szakaszt. A példánkban ezek továbbra is fred, sam és mary.

```
fred:
    set ifaddr 203.14.100.1 203.14.101.1 255.255.255.255

sam:
    set ifaddr 203.14.100.1 203.14.102.1 255.255.255.255

mary:
    set ifaddr 203.14.100.1 203.14.103.1 255.255.255.255
```

Amennyiben szükséges, az `/etc/ppp/ppp.linkup` tartalmazhat további útválasztási információkat is az egyes statikus IP-címmel rendelkező felhasználókhoz. A lentebb bemutatott sor a kliens ppp összeköttetésén keresztül vesz fel egy útvonalat a 203.14.101.0/24 hálózat felé.

```
fred:
    add 203.14.101.0 netmask 255.255.255.0 HISADDR

sam:
    add 203.14.102.0 netmask 255.255.255.0 HISADDR

mary:
    add 203.14.103.0 netmask 255.255.255.0 HISADDR
```

27.2.1.2.10. Az *mgetty* és az *AutoPPP*

Az `comms/mgetty+sendfax` port alapértelmezés szerint az `AUTO_PPP` beállítással érkezik, amely lehetővé teszi, hogy az `mgetty` képessé legyen a PPP kapcsolatok LCP fázisát észlelni és magától létrehozni hozzá egy `ppp` shellt. Mivel az alapértelmezett név/jelszó páros azonban ilyenkor nem jelenik meg, a felhasználókat a PAP vagy a CHAP protokollon keresztül lehet hitelesíteni.

Ez a szakasz most feltételezi, hogy a sikeresen beállítottuk, lefordítottuk és telepítettük az `comms/mgetty+sendfax` portot.

Az `/usr/local/etc/mgetty+sendfax/login.config` állományban ne felejtsük ellenőrizni, hogy szerepel a következő:

```
/AutoPPP/ - - /etc/ppp/ppp-pap-dialup
```

Ezzel utasítjuk az `mgetty` programot arra, hogy az észlelt PPP kapcsolatokhoz futtassa le a `ppp-pap-dialup` szkriptet.

Hozzunk létre az `/etc/ppp/ppp-pap-dialup` nevű állományt, amelyben majd a következők fognak szerepelni (az állomány legyen végrehajtható):

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

Az `/etc/ttys` állományban engedélyezett összes betárcsázós vonalhoz készítsük el a megfelelő bejegyzést az `/etc/ppp/ppp.conf` állományban. Ezek remekül meg fognak férni az imént készített definíciókkal.

```
pap:
    enable pap
    set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
    enable proxy
```

Minden olyan felhasználónak, aki ezzel a módszerrel jelentkezik be, szüksége lesz egy név/jelszó kombinációra az `/etc/ppp/ppp.secret` állományban, vagy az alábbi beállítás megadásával választhatjuk azt is, hogy a felhasználókat az `/etc/passwd` állományon keresztül a PAP protokoll segítségével azonosítsuk.

```
enable passwdauth
```

Ha statikus IP-címet akarunk kiosztani némely felhasználóknak, akkor az `/etc/ppp/ppp.secret` állományban ezt megadhatjuk a harmadik paraméternek. Erről bővebben a `/usr/share/examples/ppp/ppp.secret.sample` állományban láthatunk példát.

27.2.1.2.11. A Microsoft kiterjesztései

A PPP úgy is beállítható, hogy kérésre DNS és NetBIOS típusú névfeloldáshoz is szolgáltatson információkat.

A PPP 1.x változatával úgy lehet engedélyezni ezeket a kiterjesztéseket, ha az `/etc/ppp/ppp.conf` állomány megfelelő részeibe felvesszük a következő sorokat:

```
enable msextns
set ns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

A PPP második és későbbi változataiban pedig:

```
accept dns
set dns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

Ezzel a kliens megkapja az elsődleges és másodlagos névszerverek címeit, valamint a NetBIOS névszerveret.

Ha a második és az azt követő verziókban a `set dns` sort elhagyjuk, akkor a PPP az `/etc/resolv.conf` állományban található értékeket fogja használni.

27.2.1.2.12. A PAP és CHAP hitelesítés

Egyes internet-szolgáltatók úgy állítják be a rendszerüket, hogy a kapcsolat felépítése során a hitelesítés a PAP vagy CHAP mechanizmusok valamelyikével történik. Ilyenkor a szolgáltató nem egy `login:` sorral fogja bekérni a szükséges adatokat, hanem közvetlenül a PPP kapcsolatot kezdi el használni.

A PAP nem olyan biztonságos, mint a CHAP, de itt a biztonság nem is annyira fontos, mivel a jelszavak, amelyeket ugyan a PAP titkosítatlan formában küld tovább, csak egy soros vonalon haladnak át. A rossz indulatú támadók itt nem sok mindent tudnak "lehallgatni".

A PPP statikus IP-címmel és a PPP dinamikus IP címmel című szakaszokhoz képest a következő módosításokat kell elvégeznünk:

```

13      set authname AFelhasználóiNevem
14      set authkey AJelszavam
15      set login

```

13. sor:

Ebben a sorban adjuk meg a PAP/CHAP felhasználói nevünket, amelyet *AFelhasználóiNevem* helyett kell beírni.

14. sor:

Ebben a sorban adjuk meg a PAP/CHAP jelszavunkat, *AJelszavam* helyett. Szándékunk egyértelműsítése érdekében ezek mellett még egy további sort is érdemes felvennünk, tehát:

```

16      accept PAP

```

vagy

```

16      accept CHAP

```

Alapértelmezés szerint a PAP és CHAP is egyaránt elfogadott.

15. sor:

A PAP és CHAP alkalmazásakor általában nem is kell bejelentkeznünk a szolgáltató szerverére. Ezért a “set login” parancsnál használt karakterláncot le is kell tiltanunk.

27.2.1.2.13. A *ppp* beállításainak megváltoztatása menet közben

A háttérben futó *ppp* programhoz menet közben is tudunk beszélni, de csak olyankor, amikor az ehhez szükséges portot megadtuk. Ezt úgy tudjuk megtenni, ha beállítások közé felvesszük az alábbi:

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

Így a PPP az előre megadott UNIX tartománybeli socketen keresztül fogja várni a kapcsolódásunkat, és a konkrét hozzáféréshez jelszót kér. A névben szereplő *%d* a használatban levő *tun* eszköz sorszámát jelöli.

Miután a csatlakozás beállítódott, a szkriptekben a *pppctl(8)* program használható a futó program vezérléséhez.

27.2.1.3. A PPP hálózati címfordítási képességének kihasználása

A PPP képes a rendszermag rásegítése nélkül képes hálózati címfordítást végezni. Ezt a lehetőséget a következő sor hozzáadásával tudjuk aktiválni az */etc/ppp/ppp.conf* állományban:

```
nat enable yes
```

A PPP-be épített hálózati címfordítás a *-nat* parancssori paraméterrel is bekapcsolható. Az */etc/rc.conf* állományban is található hozzá egy *ppp_nat* változó, amely alapértelmezés szerint engedélyezett.

Amikor használjuk ezt a lehetőséget, az */etc/ppp/ppp.conf* állományban a következő opciókkal engedélyezhetjük a bejövő kapcsolatok továbbítását:

```

nat port tcp 10.0.0.2:ftp ftp
nat port tcp 10.0.0.2:http http

```

vagy egyáltalán ne bízunk meg a külvilágban:

```
nat deny_incoming yes
```

27.2.1.4. A rendszer végső beállítása

Mostanra ugyan már beállítottuk a ppp programot, azonban még néhány dolgot be kell állítanunk, mielőtt ténylegesen nekilátnánk használni. Ezek mindegyike az `/etc/rc.conf` állomány módosítását igényli.

Az állományt fentről lefelé fogjuk feldolgozni, de előtte ne felejtünk el értéket adni a `hostname=` változónak, például:

```
hostname="ize.minta.com"
```

Amennyiben a szolgáltatónk statikus IP-címet és nevet biztosít számunkra, az lesz a legjobb, ha itt a tőle kapott nevet adjuk meg.

Keressük meg a `network_interfaces` változót. Ha a rendszerünkben kérésre akarjuk tárcsázni a szolgáltatónk, akkor a `tun0` eszközt mindenképpen vegyük fel az értékebe, minden más esetben pedig távolítsuk el.

```
network_interfaces="lo0 tun0"
ifconfig_tun0=
```

Megjegyzés: Az `ifconfig_tun0` változónak üres értéket kell megadnunk, és létre kell hoznunk egy `/etc/start_if.tun0` nevű állományt. Ebben a következő sornak kell szerepelnie:

```
ppp -auto arendszerem
```

Ez a szkript a hálózat beállításakor fut le, és a ppp démont automatikus módban indítja el. Ha az adott gép egy helyi hálózat átjárója is egyben, akkor az `-alias` kapcsolót is érdemes megadnunk mellette. A pontosabb részletek tekintetében olvassuk el a megfelelő man oldalt.

Az `/etc/rc.conf` állományban a `NO` érték megadásával tiltsuk le az útválasztást végző program használatát:

```
router_enable="NO"
```

Fontos, hogy a `routed` démon ne induljon el, mivel `routed` hajlamos törölni a ppp által létrehozott alapértelmezett útválasztási bejegyzéseket.

Ezenkívül még a `sendmail_flags` változóról szóló sorból is érdemes kivenni a `-q` opciót, máskülönben a `sendmail` minden művelet megkezdése előtt nekiáll felderíteni a hálózatot, és ezzel megindítja a tárcsázást. Próbáljuk meg így átírni az értékét:

```
sendmail_flags="-bd"
```

Ezért cserébe viszont a `sendmail` programot a ppp kapcsolat létrejöttkor mindig utasítanunk kell, hogy újból ellenőrizze a levelezési sort. Ezt a következők begépelésével érhetjük el:

```
# /usr/sbin/sendmail -q
```

Ugyanezt automatikusan is meg tudjuk tenni a `!bg` paranccsal a `ppp.linkup` állományban:

```

1      szolgaltato:
2      delete ALL
3      add 0 0 HISADDR
4      !bg sendmail -bd -q30m

```

Ha nem felelne meg ez a megoldás, akkor egy “dfilter” is beállítható az SMTP forgalom szűrésére. A példák között megtaláljuk ennek pontos minkéntjét.

Ezután már csak a gépünk újraindítása maradt hátra. Az újraindítás után már be is gépelhetjük:

```
# ppp
```

ahol a `dial szolgaltato` parancs kiadásával meg tudjuk kezdeni a PPP kapcsolat felépítését, vagy a `ppp` programot megkérhetjük arra, hogy automatikusan kezdje el, amint van kimenő forgalom (és nem készítettük el a `start_if.tun0` szkriptet). Ekkor gépeljük be ezt:

```
# ppp -auto szolgaltato
```

27.2.1.5. Összefoglalás

Gyorsan foglaljuk össze, hogy az `ppp` beállításához milyen lépések megtétele szükséges az első alkalommal:

A kliens oldalán:

1. Győződjünk meg róla, hogy a `tun` eszköz benne van a rendszermagban.
2. Ellenőrizzük, hogy a `tunN` eszközhöz tartozó állomány rendelkezésre áll a `/dev` könyvtárban.
3. Hozzunk létre egy bejegyzést az `/etc/ppp/ppp.conf` állományban. A `pmdemand` példából a legtöbb szolgáltató esetében ki tudunk indulni.
4. Ha dinamikus IP-címet kapunk, akkor az `/etc/ppp/ppp.linkup` állományba is vegyünk fel egy bejegyzést.
5. Frissítsük az `/etc/rc.conf` állományunkat.
6. Ha igény szerint akarunk tárcsázni, akkor hozzunk létre `start_if.tun0` néven egy szkriptet.

A szerver oldalán:

1. Gondoskodjunk róla, hogy a `tun` eszköz támogatása szerepel rendszermagban.
2. Győződjünk meg róla, hogy a `tunN` eszköz megtalálható a `/dev` könyvtárban.
3. Az `/etc/passwd` állományban (a `vipw(8)` program használatával) hozzunk létre bejegyzéseket.
4. A felhasználók könyvtáraiban hozzunk létre egy olyan profilt, amely `ppp -direct direct-server` vagy egy ehhez hasonló parancsot futtat le.
5. Az `/etc/ppp/ppp.conf` állományban adjuk meg egy bejegyzést. A `direct-server` példa ehhez egy remek alapot biztosít.
6. Az `/etc/ppp/ppp.linkup` állományban hozzunk létre egy bejegyzést.
7. Frissítsük az `/etc/rc.conf` állományunkat.

27.3. A rendszerszintű PPP alkalmazása

Egyes részeit készítette: Gennady B. Sorokopud és Robert Huff.

Figyelem Ez a szakasz csak FreeBSD 7.x esetén érvényes.

27.3.1. A rendszerszintű PPP beállítása

Mielőtt a gépünkön nekikezdünk a PPP beállításának, ellenőrizzük, hogy a `pppd` megtalálható a `/usr/sbin` könyvtárban és az `/etc/ppp` könyvtár létezik.

A `pppd` két módban képes működni:

1. “kliensként” — a gépünket soros vonali vagy modemcsatlakoztatjuk a külvilághoz
2. “szerverként” — a számítógépünk egy hálózat része, ahol a többiek a PPP használatával kapcsoljuk össze

Mind a két esetben egy konfigurációs állomány tartalmát kell összeállítanunk (ez az `/etc/ppp/options` vagy a `~/ .ppprc`, ha a gépünkön több felhasználó is PPP-t akar használni).

Egy modemcsatlakoztatott soros vonali szoftverre is szükségünk lesz (ez többnyire a `comms/kermit`), amellyel távoli gépeket tudunk felhívni és feléjük kapcsolatot felépíteni.

27.3.2. A `pppd` mint kliens

Az alapjául szolgáló információkat adta: Trev Roydhouse.

A most következő `/etc/ppp/options` állománnyal egy Cisco terminál szerverhez tudunk kapcsolódni egy PPP vonalon keresztül.

```
crtstcts      # a hardveres forgalomirányítás engedélyezése
modem         # modem vezérlővonal
noipdefault   # a távoli PPP szervernek kell IP-címet adnia
               # ha az IPCP alapú egyeztetés során a távoli gép nem küld
               # nekünk IP-címet, akkor vegyük ki ezt a beállítást
passive       # LCP csomagokat várunk
domain ppp.ize.com      # ide írjuk be a hálózati nevünket

:távoli_ip    # ide kell írni a távoli PPP szerver IP-címét
               # a PPP kapcsolaton keresztül erre fogjuk továbbküldeni a csomagokat
               # ha nem adtuk meg "noipdefault" beállítást, akkor ezt a sort
               # írjuk át helyi_ip:távoli_ip alakúra

defaultroute  # adjuk meg ezt a sort is, ha a PPP szerverünket egyben az
               # alapértelmezett átjárónak is be akarjuk állítani
```

Így kapcsolódunk:

1. Tárcsázzuk a távoli gépet a **Kermit** (vagy bármilyen más modemes program) elindításával, majd adjuk meg a felhasználói nevünket és jelszavunkat (vagy bármi mást, amivel a távoli gépen engedélyezni tudjuk a PPP használatát).
2. Lépünk ki a **Kermit** programból (anélkül, hogy bontanánk a vonalat).
3. Írjuk be a következőket:

```
# /usr/sbin/pppd /dev/tty01 19200
```

Ne felejtjük el megadni a megfelelő sebességet és eszközt.

A számítógépünk most már PPP-n keresztül csatlakozik. Ha valamilyen okból nem sikerülne felépíteni a kapcsolatot, akkor vegyük fel a debug beállítást is az `/etc/ppp/options` állományba, majd a konzolra érkező üzenetek segítségével próbáljuk meg felderíteni a probléma okát.

Az alábbi `/etc/ppp/pppup` szkript mind a három fázist automatikussá teszi:

```
#!/bin/sh
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
pgrep -l kermi
pid=`pgrep kermi`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermi, PID=' ${pid}
    kill -9 ${pid}
fi

ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.dial
pppd /dev/tty01 19200
```

Az `/etc/ppp/kermit.dial` egy olyan **Kermit** szkript, amivel tárcsázni tudunk és a távoli gépen elvégezni az összes szükséges hitelesítést (a leírás végén találhatunk is egy ilyen szkriptet példaként).

Az alábbi `/etc/ppp/pppdown` szkripttel tudjuk bontani a PPP vonalat:

```
#!/bin/sh
pid=`pgrep pppd`
if [ X${pid} != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill -TERM ${pid}
fi

pgrep -l kermi
pid=`pgrep kermi`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermi, PID=' ${pid}
    kill -9 ${pid}
fi
```

```
/sbin/ifconfig ppp0 down
/sbin/ifconfig ppp0 delete
kermit -y /etc/ppp/kermit.hup
/etc/ppp/ppptest
```

A /usr/etc/ppp/ppptest elindításával ellenőrizni tudjuk, hogy a pppd még mindig fut. Ez valahogy így néz ki:

```
#!/bin/sh
pid=`pgrep pppd`
if [ X${pid} != "X" ] ; then
    echo 'pppd running: PID=' ${pid-NONE}
else
    echo 'No pppd running.'
fi
set -x
netstat -n -I ppp0
ifconfig ppp0
```

A vonal bontásához az /etc/ppp/kermit.hup szkriptet kell elindítanunk, amiben a következő szerepelnek:

```
set line /dev/tty01      ; ide írjuk be a saját modemünket
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
echo \13
exit
```

A kermit helyett a chat programot is használhatjuk:

A következő két állomány már elég egy kapcsolat létrehozásához pppd használatával:

/etc/ppp/options:

/dev/cuad1 115200

```
crtstcts          # a hardveres forgalomirányítás engedélyezése
modem             # modemes vezérlővonal
connect "/usr/bin/chat -f /etc/ppp/login.chat.script"
noipdefault       # a távoli PPP kiszolgálónak adnia kell egy IP-címet
                  # ha a távoli gép nem küldi az IP-címünk az IPCP alapú egyeztetés során
                  # akkor távolítsuk el ezt a beállítást
```

```

passive          # LCP csomagokat várunk
domain saját.tartomány # ide írjuk be a saját tartománynevünket

:               # a távoli PPP kiszolgáló IP-címét tegyük ide
               # ezen keresztül fogjuk továbbküldeni a PPP kapcsolaton áthaladó csomagokat
               # nem adtuk meg a "noipdefault" beállítást, akkor ezt
               # sort írjuk át helyi_ip:távoli_ip alakúra

defaultroute     # ez a sor akkor kell, ha a PPP szerver lesz az
               # alapértelmezett átjárónk is

/etc/ppp/login.chat.script:

```

Megjegyzés: A most következőt egyetlen sorba kell írunk.

```

ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDTtelefon.szám
CONNECT "" TIMEOUT 10 ogin:-\\r-ogin: bejelentkezési-azonosító
TIMEOUT 5 sword: jelszó

```

Miután ezeket telepítettük és a megfelelőképpen módosítottuk, már csak a `pppd` parancsot kell kiadnunk, valahogy így:

```
# pppd
```

27.3.3. A `pppd` mint szerver

Az `/etc/ppp/options` állományban nagyjából a következőknek kell szerepelnie:

```

crtstcts          # hardveres forgalomirányítás
netmask 255.255.255.0 # hálózati maszk (nem kötelező)
192.114.208.20:192.114.208.165 # a helyi és távoli gépek IP-címei
                                # a helyi IP-nek el kell térnie az Ethernet
                                # (vagy más egyéb) felülethez tartozó címtől.
                                # a távoli IP a távoli géphez rendelt IP-cím

domain ppp.ize.com  # a saját tartományunk
passive             # az LCP csomagok várása
modem               # modemcsatlakozás

```

Az alábbi `/etc/ppp/pppserv` szkript a `pppd` démon szervernek állítja be:

```

#!/bin/sh
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
pgrep -l kermi
pid=`pgrep kermi`
if [ "X${pid}" != "X" ] ; then

```

```

        echo 'killing kermi, PID=' ${pid}
        kill -9 ${pid}
    fi

# reset ppp interface
ifconfig ppp0 down
ifconfig ppp0 delete

# enable autoanswer mode
kermi -y /etc/ppp/kermi.ans

# run ppp
pppd /dev/tty01 19200

```

A szerver leállítására a következő /etc/ppp/pppservdown szkriptet kell használnunk:

```

#!/bin/sh
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
pgrep -l kermi
pid=`pgrep kermi`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermi, PID=' ${pid}
    kill -9 ${pid}
fi
ifconfig ppp0 down
ifconfig ppp0 delete

kermi -y /etc/ppp/kermi.noans

```

A következő **Kermi** szkript (/etc/ppp/kermi.ans) engedélyezi vagy tiltja le a modem automatikus válaszadását. Körülbelül így épül fel:

```

set line /dev/tty01
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13

```

```

inp 5 OK
echo \13
out AT$0=1\13      ; "AT$0=0\13"-ra írjuk át, ha le akarjuk tiltani az
                   ; automatikus válaszadást

inp 5 OK
echo \13
exit

```

Az `/etc/ppp/kermi$ dial` elnevezésű szkriptet használhatjuk arra, hogy tárcsázzunk távoli gépeket és hitelesítsük magunkat rajtuk. Írjuk át az igényeinknek megfelelően, tegyük bele a bejelentkezéshez szükséges azonosítót és jelszót, illetve a modemünk és a távoli gép válasza szerint módosítsuk az input utasításokat.

```

;
; írjuk ide azt a com vonalat, amire a modemünk csatlakozik:
;
set line /dev/tty01
;
; ide kerül a modem sebessége:
;
set speed 19200
set file type binary           ; teljes 8 bites állomány-átvitel
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none
set modem hayes
set dial hangup off
set carrier auto              ; adjuk meg a SET CARRIER utasítást is, ha kell
set dial display on          ; adjuk meg a SET DIAL utasítást is, ha kell
set input echo on
set input timeout proceed
set input case ignore
def \%x 0                     ; a bejelentkezés számlálója
goto slhup

:slcmd                        ; tegyük a modemet parancs módba
echo Tegyük a modemet parancs módba.
clear                         ; töröljük a be nem olvasott karaktereket a bemeneti pufferből
pause 1
output +++                    ; a Hayes-féle helyettesítési szekvenciák használata
input 1 OK\13\10              ; várjuk meg az OK jelzést
if success goto slhup
output \13
pause 1
output at\13
input 1 OK\13\10
if fail goto slcmd            ; ha a modem nem válaszol OK-val, akkor próbálkozzunk újra

:slhup                        ; bontsuk a vonalat

```

```

clear                                ; töröljük ki a be nem olvasott karaktereket a bemeneti pufferból
pause 1
echo A vonal bontasa.
output ath0\13                       ; a kapcsolat létrejöttét jelző Hayes-parancs
input 2 OK\13\10
if fail goto slcmd                   ; ha nincs OK válasz, akkor tegyük a modemet parancs módba

:sldial                              ; tárcsázzuk a számot
pause 1
echo Dialing.
output atdt9,550311\13\10           ; ide írjuk a telefonszámot
assign \%x 0                        ; nullázzuk le az időzítőt

:look
clear                                ; töröljük az olvasatlan karaktereket a bemeneti pufferból
increment \%x                       ; számoljuk a másodperceket
input 1 {CONNECT }
if success goto sllogin
reinput 1 {NO CARRIER\13\10}
if success goto sldial
reinput 1 {NO DIALTONE\13\10}
if success goto slnodial
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 60 goto look
else goto slhup

:sllogin                             ; bejelentkezés
assign \%x 0                        ; nullázzuk le az időzítőt
pause 1
echo A bejelentkezés keresese.

:slloop
increment \%x                       ; számoljuk a másodperceket
clear                                ; töröljük az olvasatlan karaktereket a bemeneti pufferból
output \13
;
; ide írjuk be a várható bejelentkezési sablont:
;
input 1 {Felhasználói nev: }
if success goto sluid
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 10 goto slloop             ; tízszer próbálkozzunk a bejelentkezéssel
else goto slhup                     ; 10 sikertelen próbálkozás után bontsuk a vonalat és kezdjük újra

:sluid
;
; ide írjuk be a felhasználói azonosítónkat:

```

```

;
output ppp-login\13
input 1 {Jelszo: }
;
; ide tegyük a hozzá tartozó jelszót:
;
output ppp-password\13
input 1 {Atvaltas SLIP modba.}
echo
quit

:slnodial
echo \7Nincs vonal. Ellenorizzuk a telefonvonalat!\7
exit 1

; local variables:
; mode: csh
; comment-start: "; "
; comment-start-skip: "; "
; end:

```

27.4. PPP kapcsolatok hibaelhárítása

Készítette: Tom Rhodes.

FigyelemA FreeBSD 8.0 kiadásától kezdődően a sio(4) meghajtó szerepét a uart(4) veszi át. Emiatt a soros vonali eszközöket `/dev/cuadN` és `/dev/cuauN` helyett `/dev/ttydN` és `/dev/ttyuN` néven lehet elérni. A FreeBSD 7.x változatok felhasználóinak ennek megfelelően kell olvasniuk ezt a leírást.

Ebben a szakaszban összefoglalunk néhány olyan problémát, ami a PPP modemeken keresztüli használata során keletkezhet. Például pontosan tisztában kell lennünk azzal, hogy a tárcsázott rendszer milyen adatokat és hogyan fog tőlünk bekérni. Egyes szolgáltatók egy `ssword` promptot, míg mások egy `password` promptot adnak. Ha a `ppp` szkript nem illeszkedik ezekhez az elvárásokhoz, akkor nem tudunk bejelentkezni. A `ppp` csatlakozások nyomonkövetésének egyik leggyakoribb módja a manuális kapcsolódás. A következőkben ezért a manuális csatlakozásokra vonatkozó legszükségesebb ismereteket mutatjuk be lépésről lépésre.

27.4.1. Az eszközeírók ellenőrzése

Ha saját rendszermagot használunk, ne felejtjük el felvenni a következő sort a konfigurációs állományba:

```
device    uart
```

A `GENERIC` rendszermag az `uart` eszközt már alapértelmezés szerint tartalmazza, ezért ilyenkor már nincs több teendőnk. Egyszerűen csak a `dmesg` parancs kimenetében keressük meg a modemes eszközhöz tartozó adatokat:

```
# dmesg | grep uart
```

Ennek eredményeképpen kapunk egy rövid összefoglalást a `uart` típusú eszközökről. Ezek lesznek a számunkra fontos COM portok. Amennyiben a modemünk egy szabványos soros portként működik, akkor a `uart1` vagy `COM2` néven kell keresnünk. Ha megtaláltuk, akkor nem kell új rendszermagot fordítanunk. Amikor a soros vonali modemünk a `uart1` vagy `COM2` porton csatlakozik DOS-ban, akkor itt a neki megfelelő eszköz a `/dev/cuaul` lesz.

27.4.2. Kapcsolódás manuálisan

A `ppp` kézi irányításával gyorsan, egyszerűen és minden fájdalomtól mentesen tudunk csatlakozni az internethez, de olyankor is hasznos, ha ki akarjuk deríteni, hogy az internet-szolgáltatónk milyen módon kezeli a kliensek `ppp` csatlakozásait. Nos, akkor ehhez indítsuk is el a **PPP** alkalmazást a parancssorból. Az alábbi példákban rendre a *pelda* névvel hivatkozunk a **PPP**-t működtető gépre. A `ppp` tehát a `ppp` parancs begépelésével indítható:

```
# ppp
```

Ezzel elindítottuk a `ppp` programot.

```
ppp ON pelda> set device /dev/cuaul
```

Beállítjuk a modemünket, ami ebben az esetben a `cuaul`.

```
ppp ON pelda> set speed 115200
```

Beállítjuk a csatlakozás sebességét, ami ebben az esetben 115 200 kbit/mp.

```
ppp ON pelda> enable dns
```

Azt mondjuk a `ppp` programnak, hogy állítsa be a névfeloldót és az `/etc/resolv.conf` állományt egészítse ki a megfelelő névszerverekkel. Ha a `ppp` nem képes megállapítani a gépünk nevét, akkor később ezt még kézzel is be tudjuk állítani.

```
ppp ON pelda> term
```

Váltunk "terminál" módba, így mi irányítjuk a modemet.

```
deflink: Entering terminal mode on /dev/cuaul
type '~h' for help
```

```
at
```

```
OK
```

```
atdt123456789
```

Az `at` paranccsal hozzuk alaphelyzetbe a modemet, majd a `atdt` paranccsal és egy telefonszám megadásával megkezdjük a szolgáltató tárcsázását.

```
CONNECT
```

Ezzel jelez vissza a kapcsolódás megkezdéséről. Ha itt bármilyen hardvertől független csatlakozási probléma merülne fel, akkor ezen a ponton tudunk ellene tenni valamit.

```
ISP Login: felhasználonev
```

Itt kell megadnunk a felhasználói nevünket, ami megegyezik a szolgáltató által adott azonosítónkkal.

```
ISP Pass:jelszo
```

Ezúttal a jelszavunkat kell megadni, amit szintén a szolgáltató bocsátott rendelkezésünkre az azonosító mellett. Akárcsak amikor bejelentkezünk a FreeBSD-be, itt sem fog látszódni a jelszavunk.

```
Shell or PPP:ppp
```

Szolgáltatótól függően előfordulhat, hogy ez a sor soha nem is jelenik meg. Itt kérdezik meg, hogy a szolgáltatónál egy shellt akarunk használni, vagy csak elindítani egy ppp kapcsolatot. Ebben a példában természetesen a ppp opciót választjuk, mivel egy internet-előfizetés birtokosai vagyunk.

```
Ppp ON pelda>
```

Figyeljük meg, hogy az első p nagybetűssé vált. Ezzel jelzi a program, hogy sikeresen csatlakoztunk a szolgáltatónkhoz.

```
PPp ON pelda>
```

Sikeresen azonosítottuk magunkat a szolgáltató felé és várjuk az IP-címünket.

```
PPP ON pelda>
```

Megkaptuk az IP-címünket és ezzel sikeresen felépült a kapcsolat.

```
PPP ON pelda>add default HISADDR
```

Itt adjuk hozzá az alapértelmezett útvonalat, amire mindenképpen szükségünk van ahhoz, hogy a külvilággal is kapcsolatban tudjunk lépni, mivel jelenleg csak a vonal másik végén lévő gépet érjük el. Ha ezt bizonyos, már meglevő útvonalak miatt nem sikerül felvenni, akkor az add elé tegyünk egy ! jelet. Ezt viszont a kapcsolat felépítése előtt is megtehetjük, így menet közben az új útvonalat felveszi a többi közé.

Ha eddig minden remekül ment, akkor ezen ponton már egy élő internet-kapcsolattal rendelkezünk, és a programot a **CTRL+z** lenyomásával a háttérbe is tehetjük. Ha a PPP felirat ismét a ppp feliratra váltana, akkor az arra utal, hogy elvesztettük a kapcsolatot. Erre nem árt figyelni, mivel ezzel jelzi az aktuális kapcsolat állapotát. A nagybetűs P-k jelöli, hogy az adott szinten megvan a kapcsolat a szolgáltató felé, a kisbetűs p-k pedig arra utalnak, hogy azon a szinten a kapcsolat valamiért megszűnt. A ppp csak ezt a két állapotot ismeri.

27.4.2.1. Nyomkövetés

Ha közvetlen vonalunk van és mégsem sikerül kapcsolatot létesíteni, akkor tiltsuk le a hardveres CTS/RTS forgalomirányítást a `set ctsrts off` paranccsal. Ez leginkább akkor fordul elő, ha csatlakoztunk egy olyan terminálszerverhez, amely valamennyire képes kezelni a PPP kapcsolatokat, de a PPP megáll, mikor adatot próbál írni a kommunikációs csatornára, mivel arra a CTS (Clear To Send — “lehet küldeni”) jelzésre vár, amely soha nem fog megérkezni. Ha mégis ezt a beállítást akarjuk használni, akkor a `set accmap` beállításra is szükségünk lesz, mivel ez kell bizonyos karakterek hardverfüggetlen átküldésének felülbírálásához, legtöbb esetben a XON/XOFF miatt. A ppp(8) man oldalon találhatunk erről és ennek használatáról részletesebb leírást.

Ha egy régebbi gyártmányú modemünk van, akkor a `set parity even` beállítás alkalmazása is javasolt. Alapértelmezés szerint ugyanis nincs paritás, de a régebbi modemek és (a forgalom növekedésével) egyes szolgáltatók még használják hibaellenőrzésre. Ha Compuserve előfizetésünk van, mindenképpen kapcsoljuk be.

Amikor a **PPP** nem tér vissza parancs módba, akkor gyaníthatóan az egyeztetésben lesz valahol probléma, mivel a szolgáltató a kliensüktől várja a kezdeményezését. Ezen a ponton a `~p` paranccsal utasíthatjuk a ppp programot a konfigurációs információk átküldésének megkezdésére.

Ha egyáltalán nem kapunk promptot a bejelentkezéshez, akkor nagy a alószínűsége, hogy az iménti UNIX stílusú hitelesítés helyett PAP vagy CHAP protokollt kell használnunk. A PAP vagy CHAP használatához mindössze a következő beállításokat kell megadnunk **PPP** programnak a terminál mód aktiválása előtt:

```
ppp ON pelda> set authname felhasználonev
```

ahol a *felhasználonev* helyett a szolgáltatótól kapott azonosítót kell beírunk.

```
ppp ON pelda> set authkey jelszo
```

ahol a *jelszo* helyett a szolgáltatótól kapott jelszót kell megadnunk.

Ha sikeresen csatlakoztunk, de még nem találunk semmilyen tartománynevet, akkor a ping(8) és IP-cím segítségével tudjuk megvizsgálni, hogy működőképes-e a kapcsolat. Ha 100 százalékos (100%) csomagvesztést (packet loss) tapasztalunk, akkor szinte biztos, hogy nincs meg az alapértelmezett útvonal. Nézzük meg újra, hogy az `add default HISADDR` beállítást megadtuk-e a kapcsolat felépítésekor. Ha viszont már el tudunk érni egy távoli IP-címet, akkor nagyon valószínű, hogy az `/etc/resolv.conf` állományba nem került bele a megfelelő névfeloldó címe. Az említett állománynak valahogy így kellene kinéznie:

```
domain minta.com
nameserver x.x.x.x
nameserver y.y.y.y
```

Ahol az *x.x.x.x* és *y.y.y.y* címeket a szolgáltatónk névszervereinek címével kell behelyettesíteni. Ez nem minden esetben található meg az előfizetői szerződésben, de ha felhívjuk a szolgáltatónkot, akkor minden bizonnyal elárulják ezeket a címeket.

A syslog(3) is alkalmas a **PPP** kapcsolatok naplózására. Ehhez csupán ennyit kell megadnunk az `/etc/syslog.conf` állományban:

```
!ppp
*. * /var/log/ppp.log
```

A legtöbb esetben ez a lehetőség már eleve adott.

27.5. A PPP használata Ethernet felett (PPPoE)

Készítette (a <http://node.to/freebsd/how-tos/how-to-freebsd-pppoe.html> alapján): Jim Mock.

Ebben a szakaszban azt ismertetjük, hogyan állítsuk be a PPP-t Ethernet felett (PPP over Ethernet, PPPoE).

27.5.1. A rendszermag beállítása

A PPPoE működéséhez most már semmilyen módosításra nincs szükség a rendszermag beállításában. Amennyiben a hozzá szükséges Netgraph támogatás nem található a rendszermagban, akkor azt a **ppp** önműködően betölti.

27.5.2. A ppp.conf beállítása

Íme egy működő ppp.conf állomány:

```
default:
    set log Phase tun command # itt akár egy részletesebb naplózást is be tudunk állítani
    set ifaddr 10.0.0.1/0 10.0.0.2/0

a_szolgaltato_neve:
    set device PPPoE:x11 # az x11 helyére írjuk be a saját Ethernet eszközünket
    set authname FELHASZNALONEV
    set authkey JELSZO
    set dial
    set login
    add default HISADDR
```

27.5.3. A ppp futtatása

root felhasználóként adjuk ki az alábbi parancsot:

```
# ppp -ddial a_szolgaltato_neve
```

27.5.4. A ppp indítása a rendszerindítás során

Az /etc/rc.conf állományba vegyük fel a következőket:

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_nat="YES" # csak akkor, ha címfordítás kell a helyi hálózaton, máskülönben "NO"
ppp_profile="a_szolgaltato_neve"
```

27.5.5. A szolgáltatási címkék használata

Bizonyos esetekben szolgáltatási címkét (service tag) is használnunk kell a kapcsolat létrehozásához. A szolgáltatási címkék segítségével tudjuk megkülönböztetni az adott hálózaton elérhető különböző PPPoE szervereket.

A szolgáltatótól kapott dokumentációban szerepelnie kell minden ehhez kapcsolódó információnak. Amennyiben nem találjuk, érdeklődjünk a szolgáltatónál.

Utolsó reményként megpróbálhatjuk a Portgyűjteményben található Roaring Penguin PPPoE (<http://www.roaringpenguin.com/pppoe/>) nevű program által javasolt módszert. Ennél vegyük azonban számításba, hogy félre tudja programozni a modemünket, amitől akár használhatatlanná is válhat, ezért kétszer is gondoljuk meg, mielőtt használni kezdjük. Egyszerűen csak tegyük fel a szolgáltatótól a modemünk mellé kapott szoftvert. Ezután lépünk be a program **System** menüjébe. Itt kell lennie a megfelelő profilnak, ami általában az *ISP*.

A profil neve (a szolgáltatás címkéje) a ppp.conf állományban a PPPoE bejegyzés részeként jelenik meg a set device parancsban (ennek pontos részleteit lásd a ppp(8) man oldalon). Tehát nagyjából így néz ki:

```
set device PPPoE:x11:ISP
```

Az *x11* eszköz nevét ne felejtjük el a megfelelő Ethernet kártyához tartozó eszköz nevére kicserélni.

Az *ISP* helyett pedig írjuk be az imént kiderített profil nevét.

A témával kapcsolatban az alábbi helyeken találhatunk további információkat:

- Cheaper Broadband with FreeBSD on DSL (<http://renaud.waldura.com/doc/freebsd/pppoe/>), írta: Renaud Waldura (angolul).
- Nutzung von T-DSL und T-Online mit FreeBSD (<http://www.ruhr.de/home/nathan/FreeBSD/tdsl-freebsd.html>), írta: Udo Erdelhoff (németül).

27.5.6. PPPoE és a 3Com® HomeConnect® ADSL Modem Dual Link

Ez a modem nem felel meg az RFC 2516 (<http://www.faqs.org/rfcs/rfc2516.html>) előírásainak (*A Method for transmitting PPP over Ethernet (PPPoE)*), írta: L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone és R. Wheeler). Helyette az Ethernet keretekben eltérő csomagtípus kódokat használ. A 3Com-nál (<http://www.3com.com/>) panaszkodjunk, ha szerintünk is be kellene tartaniuk a PPPoE specifikációját.

A FreeBSD is csak akkor lesz képes együttműködni ezzel az eszközzel, ha beállítjuk a megfelelő `sysctl` változót. Ezt a rendszerindítás során automatikusan meg tudjuk tenni az `/etc/sysctl.conf` módosításával:

```
net.graph.nonstandard_pppoe=1
```

vagy közvetlenül az alábbi paranccsal:

```
# sysctl net.graph.nonstandard_pppoe=1
```

Sajnos, mivel ez egy rendszerszintű beállítás, ezért a 3Com HomeConnect® ADSL Modem és más normális PPPoE kliens vagy szerver egyszerre nem használható.

27.6. PPP ATM felett (PPPoA)

Most a PPP ATM feletti (PPP over ATM, PPPoA) beállítását fogjuk bemutatni. A PPPoA az európai DSL szolgáltatók körében igen nagy népszerűségnek örvend.

27.6.1. PPPoA használata az Alcatel SpeedTouch™ USB-vel

Az ilyen eszközökhöz tartozó PPPoA támogatás a FreeBSD-ben portként áll rendelkezésre, mivel az ehhez szükséges firmware csak az Alcatel licenclési feltételei szerint (http://www.speedtouchdsl.com/disclaimer_lx.htm) terjeszthető, ezért nem lehet része az alap FreeBSD rendszernek.

A szoftver telepítéséhez ezért a Portgyűjteményt kell használnunk. Telepítsük a `net/pppoa` portot és kövessük a mellékelt utasításokat.

Sok más USB-s eszközhöz hasonlóan az Alcatel SpeedTouch™ USB-nek a gépünkről kell letöltenie a működéséhez szükséges firmware-t. Ez a folyamat FreeBSD alatt automatizálható, tehát ez a másolás minden esetben megtörténik, amikor az eszközt az USB portra csatlakoztatjuk. Ehhez az `/etc/usbd.conf` állományba a következő adatokat kell beletennünk. Az állományt `root` felhasználóként tudjuk csak szerkeszteni.

```
device "Alcatel SpeedTouch USB"
    devname "ugen[0-9]+"
    vendor 0x06b9
    product 0x4061
    attach "/usr/local/sbin/modem_run -f /usr/local/libdata/mgmt.o"
```

Az **usbd**, vagyis az USB démon engedélyezéséhez az `/etc/rc.conf` állományba tegyük bele az alábbi:

```
usbd_enable="YES"
```

Emellett még a **ppp** kapcsolatot is be tudjuk állítani az indítás során. Ehhez mindössze a következő sort kell megadnunk az `/etc/rc.conf` állományban. Ismét megemlíjtük, hogy ezt a műveletet csak a **root** felhasználóval tudjuk végrehajtani.

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_profile="adsl"
```

Ezután úgy tudjuk szóra bírni a kapcsolatot, ha a `net/pppoe` porthoz mellékelt `ppp.conf` állományt használjuk fel kiindulásként.

27.6.2. Az mpd használata

Az **mpd** segítségével többféle szolgáltatáshoz, köztük a PPTP-hez hozzá tudunk férni. Az **mpd** a Portgyűjteményben `net/mpd` néven található meg. Sok ADSL modemnek szüksége van egy PPTP tunnelre közte és gép között. Ilyen modem például az Alcatel SpeedTouch Home is.

Először magát a portot kell telepítenünk, majd ezután már be tudjuk állítani az **mpd**-t a saját és a szolgáltatónk igényei szerint. A port a rengeteg leírással megtűzdelt minta konfigurációs állományait a `PREFIX/etc/mpd/` könyvtárba teszi. Itt a `PREFIX` azt a könyvtárat jelöli, ahova a portok kerülnek. Ez alapból a `/usr/local/`. Az **mpd** beállításáról szóló teljes dokumentáció a telepítés után elérhető HTML formátumban a `PREFIX/share/doc/mpd/` könyvtárban. Íme egy példa az **mpd** beállítására ADSL kapcsolatok esetében. Az ezzel kapcsolatos beállításaink két állományra bomlanak, melyek közül az első az `mpd.conf`:

```
default:
    load adsl

adsl:
    new -i ng0 adsl adsl
    set bundle authname felhasználónév ❶
    set bundle password jelszó ❷
    set bundle disable multilink

    set link no pap acfcomp protocomp
    set link disable chap
    set link accept chap
    set link keep-alive 30 10

    set ipcp no vjcomp
    set ipcp ranges 0.0.0.0/0 0.0.0.0/0

    set iface route default
```

```

set iface disable on-demand
set iface enable proxy-arp
set iface idle 0

open

```

- ❶ A felhasználói azonosító, amellyel a szolgáltató felé hitelesítjük magunkat.
- ❷ Az azonosítóhoz tartozó jelszó, amelyet szintén a szolgáltatótól kaptunk.

Az `mpd.links` állomány tartalmazza a felépítendő kapcsolatra vagy kapcsolatokra vonatkozó információkat. Például az előbbiekhöz tartozó `mpd.links` tartalma ez:

```

adsl:
    set link type pptp
    set pptp mode active
    set pptp enable originate outcall
    set pptp self 10.0.0.1 ❶
    set pptp peer 10.0.0.138 ❷

```

- ❶ A FreeBSD-s számítógépünk címe, ahonnan az **mpd** indul.
- ❷ Az ADSL modemünk IP-címe. Az Alcatel SpeedTouch Home esetén ez a cím alapértelmezés szerint a 10.0.0.138.

A kapcsolat ezek után pillanatok alatt felépíthető, ha a `root` felhasználóval kiadjuk a következő parancsot:

```
# mpd -b adsl
```

A kapcsolat állapotát a következő paranccsal tudjuk ezután ellenőrizni:

```

% ifconfig ng0
ng0: flags=88d1<UP,POINTOPOINT,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
    inet 216.136.204.117 --> 204.152.186.171 netmask 0xffffffff

```

FreeBSD alatt az **mpd** használata ajánlott az ADSL szolgáltatások eléréséhez.

27.6.3. A pptpclient használata

FreeBSD alatt a `net/pptpclient` segítségével is tudunk PPPoA típusú szolgáltatásokhoz kapcsolódni.

A `net/pptpclient` felhasználásával úgy tudunk DSL szolgáltatásokat elérni, ha feltelepítjük a hozzá tartozó portot vagy csomagot, majd módosítjuk az `/etc/ppp/ppp.conf` állományt. Mind a két műveletet csak `root` felhasználóként tudjuk lebonyolítani. Ehhez egy `ppp.conf` állományt lentebb adtunk meg. A `ppp.conf` állományban található további beállítási lehetőségekről a `ppp(8)` man oldalon olvashatunk.

```

adsl:
    set log phase chat lcp ipcp ccp tun command
    set timeout 0
    enable dns
    set authname felhasználónév ❶
    set authkey jelszó ❷
    set ifaddr 0 0

```

```
add default HISADDR
```

- ❶ A DSL szolgáltatóunktól kapott felhasználói név.
- ❷ Az előfizetéshez tartozó jelszó.

Figyelem Mivel az előfizetéshez tartozó jelszót a `ppp.conf` állományba titkosítatlan formában kell szerepeltetnünk, ezért gondoskodjunk róla, hogy senki sem képes olvasni a tartalmát. A most következő parancsokkal beállítjuk, hogy ez az állomány csak a `root` felhasználó számára legyen olvasható. A részletekért lásd a `chmod(1)` és `chown(8)` man oldalakat.

```
# chown root:wheel /etc/ppp/ppp.conf
# chmod 600 /etc/ppp/ppp.conf
```

Ezzel a paranccsal a DSL útválasztónk felé nyitunk egy tunnelt a PPP kapcsolathoz. Az Ethernetes DSL modemek általában egy előre beállított helyi hálózati IP-címmel rendelkeznek, amelyhez tudunk csatlakozni. Az Alcatel SpeedTouch Home esetében ez a cím a `10.0.0.138`. Az útválasztóhoz adott dokumentációban keressük meg, hogy az eszközeinkhöz konkrétan milyen cím tartozik. A tunnel megnyitásához és a PPP kapcsolat megindításához a következő parancsot kell kiadnunk:

```
# pptp cím adsl
```

Tipp: Az iménti parancs végére még érdemes odatenni az “et” jelet (“&”) is, mivel így a **pptp** működését a háttérben folytatja.

A parancs hatására a virtuális tunnelt megtestesítő `tun` eszköz jön létre a **pptp** és **ppp** programok között. Miután visszakaptuk a parancssort, vagy a **pptp** program megerősítette a kapcsolódás sikerességét, a keletkezett járatot így tudjuk ellenőrizni:

```
% ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
    inet 216.136.204.21 --> 204.152.186.171 netmask 0xffffffff00
    Opened by PID 918
```

Ha nem tudnánk valamiért csatlakozni, akkor először nézzük meg az útválasztónk beállításait, ami általában a **telnet** vagy egy böngésző segítségével elérhető. Ha még mindig nem vagyunk képesek csatlakozni, akkor a **pptp** parancs kimenetében és **ppp** `/var/log/ppp.log` néven elérhető naplójában kereshetünk árulkodó nyomokat.

27.7. A SLIP használata

Eredetileg készítette: Satoshi Asami. A hozzávalókat biztosította: Guy Helmer és Piero Serini.

Figyelem Ez a szakasz csak FreeBSD 7.x rendszerekre érvényes.

27.7.1. A SLIP kliensek beállítása

A következőkben azt mutatjuk be, hogy egy FreeBSD-s gépet miként tudunk egy hálózaton statikus névvel beállítani a SLIP használatával. A dinamikus hálózati nevek használatakor (vagyis amikor a címünk minden egyes tárcsázáskor megváltozhat) egy valamivel bonyolultabb beállításra van szükségünk.

Először is állapítsuk meg, hogy a modemünk melyik soros portra csatlakozik. Sokan `/dev/modem` néven egy szimbolikus linket hoznak létre a valódi eszközre, például a `/dev/cuadN` leíróra. Ennek köszönhetően az eszköz tényleges névetől el tudunk vonatkoztatni és soha nem kell módosítanunk semmit, ha a modemet például egy másik portra kell átraknunk. Ugyanis könnyedén kacifántossá tud válni a helyzet, amikor egyszerre kell megváltoztatnunk egy rakat dolgot az `/etc` könyvtárban és módosítanunk az összes `.kermrc` állományt!

Megjegyzés: A `/dev/cuad0` a COM1 port, a `/dev/cuad1` a COM2 és így tovább.

A rendszermag beállításait tartalmazó állományban a következőnek mindenképpen szerepelnie kell:

```
device    sl
```

Mivel ez általában a `GENERIC` rendszermagban megtalálható, így ez nem okoz semmilyen gondot, kivéve, hogy ha korábban már kitöröltük.

27.7.1.1. Amit csak egyszer kell megtenni

1. Vegyük fel az otthoni gépünket, az átjárónkat és a névszervereket az `/etc/hosts` állományba. Erre álljon itt egy konkrét példa:

```
127.0.0.1          localhost localhost
136.152.64.181     water.CS.Example.EDU water.CS water
136.152.64.1       inr-3.CS.Example.EDU inr-3  slip-gateway
128.32.136.9       ns1.Example.EDU ns1
128.32.136.12      ns2.Example.EDU ns2
```

2. Figyeljünk oda, hogy az `/etc/nsswitch.conf` állományban szereplő `hosts` szakaszban a `dns` szó előtt a `files` szónak kell megjelennie. Ezek nélkül mókás dolgok tudnak történni rendszerünkben.
3. Szerkesszük át az `/etc/rc.conf` állományt.

1. A hálózati nevünket a következő sorban tudjuk megadni:

```
hostname="az.en.nevem"
```

Ide a gépünk teljes internetes hálózati nevét kell beírunk.

2. Az alapértelmezett átjárót az alábbi sor módosításával tudjuk beállítani úgy, hogy a

```
defaultrouter="NO"
```

változó értékét átírjuk:

```
defaultrouter="slip-gateway"
```

4. Készítsük el az `/etc/resolv.conf` állományt, amelyben majd a következők legyenek:

```
domain CS.Example.EDU
nameserver 128.32.136.9
nameserver 128.32.136.12
```

Látható, hogy ezek a névfeloldásért felelős szerverek címei. Természetesen a ténylegesen beírandó tartomány (domain) neve és a névszerverek címei mindig az adott környezetünktől függenek.

5. Állítsuk be egy jelszót a root éstoor felhasználóknak (és mindenki másnak, akinek még nem lenne).
6. Indítsuk újra a számítógépünket és utána győződjünk meg róla, hogy a megfelelő hálózati névvel rendelkezik.

27.7.1.2. A SLIP kapcsolatok felépítése

1. Tárcsázzunk és gépeljük be a slip parancsot, majd ezt követően a gépünk nevét és a jelszót. Ez leginkább a konkrét környezettől függ. Ha a **Kermit** nevű programot használjuk, akkor egy ilyen szkripttel is próbálkozhatunk:

```
# a kermit beállítása
set modem hayes
set line /dev/modem
set speed 115200
set parity none
set flow rts/cts
set terminal bytesize 8
set file type binary
# a következő makró felelős a tárcsázásért és a bejelentkezésért
define slip dial 643-9600, input 10 =>, if failure stop, -
output slip\x0d, input 10 Azonosito:, if failure stop, -
output silvia\x0d, input 10 Jelszo:, if failure stop, -
output ***\x0d, echo \x0aCONNECTED\x0a
```

Természetesen a felhasználói nevet és a jelszót a sajátunkra kell benne kicserélnünk. Miután ezzel is megvagyunk, a **Kermit** parancssorában a csatlakozáshoz egyszerűen csak írjuk be, hogy slip.

Megjegyzés: Nem javasoljuk, hogy az állományrendszeren a jelszavakat titkosítatlan formában tároljuk. Mindeki csak a saját felelősségére tegyen ilyet.

2. Hagyjuk el a **Kermit** programot (a **Ctrl-z** billentyűkombinációval bármikor fel tudjuk függeszteni a futását) és root felhasználóként írjuk be a következőt:

```
# slattach -h -c -s 115200 /dev/modem
```

Ha ezután már képesek vagyunk a ping paranccsal elérni az útválasztó másik oldalán található gépet, akkor az azt jelenti, hogy sikerült csatlakoznunk! Ha viszont itt még nem járnánk sikerrel, akkor az slattach parancsnak ne a -c paramétert adjuk meg, hanem a -a paramétert.

27.7.1.3. Hogyan bontsunk egy kapcsolatot

Tegyük a következőket:

```
# kill -INT `cat /var/run/slattach.modem.pid`
```

Ez leállítja az `slattach` programot. Ne felejtsük el azonban, hogy ezt csak a `root` felhasználóval tudjuk végrehajtani. Ezután térjünk vissza a `kermit` programhoz (ha felfüggesztettük volna, akkor ehhez a `fg` parancsra lesz szükségünk), és lépünk ki belőle (`q`).

Az `slattach(8)` man oldala ehhez a `ifconfig sl0 down` parancsot javasolja, amellyel lényegében leállítjuk a hozzá tartozó felületet. Igazából a kettő között nincs semmilyen komolyabb eltérés (mivel az `ifconfig sl0` is ugyanezt eredményezi.)

Néha előfordulhat, hogy a modem egyszerűen nem hajlandó eldobni a vonalat. Ilyen esetekben indítsuk el a `kermit` programot és lépünk ki megint. Másodjára általában már sikerül.

27.7.1.4. Hibaelhárítás

Ha valamiért ez mégsem válna be, akkor csak nyugodtan kérdezősködjünk a `freebsd-net` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-net>) levelezési listán. A tapasztalatok szerint az embereknek eddig a következőkkel voltak problémáik:

- Az `slattach` meghívásakor sem a `-c`, sem pedig a `-a` paramétert nem adták meg. (Ez ugyan nem végzetes hiba, de egyes felhasználók szerint ez segített megoldani a gondokat.)
- Az `sl0` helyett `s10`-et írtak be (egyes betűtípusoknál könnyen össze lehet téveszteni ezeket).
- Az `ifconfig sl0` segítségével ellenőrizhető a felület állapota. Például ilyet láthatunk:

```
# ifconfig sl0
sl0: flags=10<POINTOPOINT>
    inet 136.152.64.181 --> 136.152.64.1 netmask ffffffff00
```

- Ha a `ping(8)` `no route to host` hibaüzenetet ad, akkor az útválasztási táblázattal van a gond. A `netstat -r` parancssal gyorsan ki tudjuk listázni a rendszerünkben jelenleg nyilvántartott utakat:

```
# netstat -r
Routing tables

Destination      Gateway           Flags      Refs      Use  IfaceMTU    Rtt      Netmasks:

(root node)
(root node)

Route Tree for Protocol Family inet:
(root node) =>
default          inr-3.Example.EDU  UG          8    224515  sl0 -        -
localhost.Exampl localhost.Example. UH          5     42127  lo0 -        0.438
inr-3.Example.ED water.CS.Example.E UH          1         0  sl0 -        -
water.CS.Example localhost.Example. UGH         34  47641234  lo0 -        0.438
(root node)
```

Az előző példákat egy viszonylag forgalmas rendszerből ragadtuk ki. A rendszerünkön megjelenő számok a hálózati aktivitás mértékének függvényei.

27.7.2. A SLIP szerverek beállítása

Ebben a leírásban igyekszünk bemutatni hogyan kell egy FreeBSD típusú rendszer alatt SLIP szervert beállítani, ami általában annyit jelent, hogy a rendszerünben a távoli SLIP kliensek csatlakozásakor automatikusan elindítjuk a kapcsolatokat.

27.7.2.1. Előfeltételek

Ez a szakasz igen szakmai jellegű, ezért az olvasó részéről feltételezzük a témában némi alapismeretet. Ez alatt alapvetően a TCP/IP hálózati protokollt értjük, különös hangsúllyal a hálózatok és hálózati csomópontok címezésén, a hálózati maszkokon, alhálózatokon, útválasztáson, az olyan útválasztási protokollokon, mint például a RIP. A SLIP beállítása egy betárcsázós szerveren mindezen fogalmak ismeretét igényli, és ha ezekkel még nem lennénk tisztában, akkor olvassuk el például Craig Hunt *TCP/IP Network Administration* című könyvét (O'Reilly & Associates, Inc.; ISBN: 0-937175-82-X) vagy Douglas Comer TCP/IP protokollról szóló könyveit.

Mindezek mellett még feltételezzük, hogy már beállítottuk a modem(ek)et és a rajtuk keresztüli bejelentkezéshez szükséges állományokat. Ha még nem készítettük volna fel erre a rendszerünket, akkor a 26.4 Szakasz ad részletes tájékoztatást a betárcsázós szolgáltatások beállításáról. A soros vonali eszközmeghajtóval kapcsolatban továbbá érdemes átolvasni a `sio(4)` oldalt, valamint a `ttys(5)`, `gettytab(5)`, `getty(8)` és `init(8)` oldalakat a bejelentkezések modemén keresztüli fogadásáról, illetve talán az `stty(1)` oldalt a soros port paramétereinek megfelelő beállításáról (mint például a `cllocal` a közvetlenül csatlakozó soros felületek esetében).

27.7.2.2. Gyors áttekintés

A FreeBSD SLIP szerverként általában a következő módon üzemel: a SLIP felhasználó tárcsázza a FreeBSD-s SLIP szerverünket, majd bejelentkezik egy speciális SLIP bejelentkezési azonosító használatával, amely a `/usr/sbin/sllogin` shellt használja. A `sllogin` program az `/etc/sliphome/slip.hosts` állományban megkeresi a speciális felhasználóhoz tartozó sort, és ha talál egy ilyet, akkor csatlakoztatja a soros vonalat egy rendelkezésre álló SLIP felületre, amelyen aztán a SLIP felület beállításához lefuttatja az `/etc/sliphome/slip.login` shell szkriptet.

27.7.2.2.1. Példa SLIP szerveren keresztüli bejelentkezésre

Például, ha a SLIP felhasználó azonosítója `Shelmerg`, akkor az `/etc/master.passwd` állományban a hozzá tartozó bejegyzést nagyjából ilyen:

```
Shelmerg:password:1964:89::0:0:Guy Helmer - SLIP:/usr/users/Shelmerg:/usr/sbin/sllogin
```

Amikor `Shelmerg` bejelentkezik, a `sllogin` az `/etc/sliphome/slip.hosts` állományban keresni fog egy felhasználó azonosítójához illeszkedő sort. Például tegyük fel, hogy az `/etc/sliphome/slip.hosts` állományban szerepel egy ilyen sor:

```
Shelmerg          dc-slip sl-helmer          0xfffffc00          autocomp
```

A `sllogin` ezt a sor fogja megtalálni, majd a soros vonalat a következő elérhető SLIP felülethez kapcsolja, amelyen ezután végrehajtja az `/etc/sliphome/slip.login` szkriptet a következő módon:

```
/etc/sliphome/slip.login 0 19200 Shelmerg dc-slip sl-helmer 0xfffffc00 autocomp
```

Ha minden jól megy, akkor az `/etc/sliphome/slip.login` kiad egy `ifconfig` parancsot azon a SLIP felületen, amelyre a `sllogin` magát csatlakoztatta (amely a fenti példában a 0. SLIP felület volt, és amelyet meg is adtunk

`slip.login` első paramétereként), és így beállítja a helyi IP-címet (`dc-slip`), a távoli IP-címet (`sl-helmer`), a SLIP felülethez tartozó hálózati maszkot (`0xfffffc00`) valamint a további opciókat (`autocomp`). Ha valami rosszul sülné el, akkor a `sliplogin` ezekről általában nagyon jó minőségű, információdús üzeneteket készít, amelyeket a **syslogd** démon pedig a `/var/log/messages` állományba rögzít. (A `syslogd(8)` és `syslog.conf(5)` man oldalak és talán maga az `/etc/syslog.conf` segíthet kideríteni, hogy a **syslogd** jelenleg naplóz-e, és ha igen, akkor hova.)

27.7.2.3. A rendszermag beállítása

A FreeBSD alap (vagyis a `GENERIC`) rendszermagja támogatja a SLIP (`sl(4)`) használatát. Ha viszont saját rendszermagunk van, akkor előfordulhat, hogy beállítások közé fel kell vennünk a következő sort is:

```
device    sl
```

Alapértelmezés szerint a FreeBSD nem továbbít semmilyen csomagot. Amennyiben a FreeBSD SLIP szerverünket útválasztóként is működtetni akarjuk, úgy az `/etc/rc.conf` állományban a `gateway_enable` változót át kell állítanunk a `YES` értékre. Ennek hatására az újraindítás után is megmarad a csomagok továbbítása.

A változtatások azonnali életbeléptetéséhez adjuk ki `root` felhasználóként a következő parancsot:

```
# /etc/rc.d/routing start
```

Ha a FreeBSD rendszermag beállítása során segítségre szorulnánk, akkor olvassuk el 8 fejezetet.

27.7.2.4. A sliplogin beállítása

Ahogy arra már korábban is utaltunk, az `/etc/sliphome` könyvtárban három állomány felelős a `/usr/sbin/sliplogin` beállításáért (lásd `sliplogin(8)`): a `slip.hosts`, amelyekben a SLIP felhasználókat és a hozzájuk tartozó IP-címeket adjuk meg; a `slip.login`, amely általában csak a SLIP felületet állítja be; (az elhagyható) `slip.logout`, amely a soros vonal bontásakor a `slip.login` hatását igyekszik visszafordítani.

27.7.2.4.1. A slip.hosts beállítása

Az `/etc/sliphome/slip.hosts` soraiban whitespace karakterekkel tagoltan legalább négy elem szerepel:

- a SLIP felhasználó bejelentkezési azonosítója
- a SLIP kapcsolat helyi címe (a SLIP szerveréhez képest)
- a SLIP kapcsolat távoli címe
- hálózati maszk

A helyi és távoli címek lehetnek hálózati nevek is (amelyeket vagy az `/etc/hosts`, vagy pedig az `/etc/nsswitch.conf` állományban szereplő beállítások alapján tudunk feloldani IP-címre), illetve a hálózati maszk is lehet egy olyan név, amelyet az `/etc/networks` fel tud oldani. A példaként bemutatott rendszerünkben az `/etc/sliphome/slip.hosts` állomány nagyjából így épül fel:

```
#
# login helyi-cím          távoli-cím          maszk          opc1      opc2
#                               (normal,compress,noicmp)
#
```

```
Shelmerg dc-slip      sl-helmerg      0xfffffc00      autocomp
```

A sorok végén az alábbi opciók közül egy vagy több szerepelhet:

- `normal` — a fejléceket nem tömörítjük
- `compress` — a fejlécek tömörítése
- `autocomp` — ha a távoli végpont engedi, akkor tömörítsük a fejléceket
- `noicmp` — az ICMP csomagok tiltása (így például a “ping” által generált csomagok is eldobódnak a sávszélesség felemésztese helyett)

A SLIP kapcsolathoz tartozó helyi és távoli címek megválasztása függ attól, hogy egy külön TCP/IP alhálózatot szentelünk-e neki, vagy a SLIP szerverünkön egy “ARP proxy”-t használunk (amely tulajdonképpen nem egy “valódi” ARP proxy, de ebben a szakaszban így fogunk rá hivatkozni). Ha nem vagyunk biztosak benne, hogy melyik módszert válasszuk vagy hogy miként osszuk ki az IP-címeket, akkor nézzünk utána ezekenek a SLIP használatával kapcsolatos előfeltételek között megemlített könyvekben (27.7.2.1 Szakasz) és/vagy konzultáljunk a hálózatunk karbantartójával.

Ha a SLIP klienseknek külön alhálózatokat osztunk ki, akkor a saját IP-címünkből kell létrehoznunk és kiadnunk ezeket. Ezután valószínűleg a SLIP szerverünkön keresztül még meg kell adnunk egy statikus útvonalat legközelebbi IP útválasztó felé.

Minden más esetben az “ARP proxy” módszert kell alkalmaznunk, ahol a SLIP kliensek IP-címeit a SLIP szerver Ethernet alhálózatából osztjuk ki, és ennek megfelelően az `/etc/sliphome/slip.login` és `/etc/sliphome/slip.logout` szkripteket módosítanunk kell úgy, hogy az `arp(8)` segítségével képesek legyenek a SLIP szerver ARP táblázatában kezelni a “proxy ARP” bejegyzéseket.

27.7.2.4.2. A `slip.login` beállítása

Egy átlagos `/etc/sliphome/slip.login` állomány körülbelül ilyen:

```
#!/bin/sh -
#
#      @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# Egy általános slip vonali bejelentkezési állomány. A sliplogin ezt az alábbi
# paraméterekkel hívja meg:
#      1      2      3      4      5      6      7-n
#      slipegys. ttyseb. azonosító helyi-cím távoli-cím maszk egyéb-pmek.
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
```

Ez a `slip.login` állomány az `ifconfig` segítségével pusztán beállítja a megfelelő SLIP felülethez tartozó helyi, valamint távoli címet és a hálózati maszkot.

Ha ehelyett azonban az “ARP proxy” módszerét választottuk volna (tehát a SLIP klienseknek nem akarunk egész alhálózatokat kiutalni), akkor az `/etc/sliphome/slip.login` állomány eképpen alakul:

```
#!/bin/sh -
#
#      @(#)slip.login  5.1 (Berkeley) 7/1/90
```

```
#
# Egy általános slip vonali bejelentkezési állomány. A sliplogin ezt az alábbi
# paraméterekkel hívja meg:
#      1      2      3      4      5      6      7-n
#   slipegys. ttyseb. azonosító helyi-cím távoli-cím maszk egyéb-pmek.
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
# A SLIP kliensre vonatkozó ARP kéréseket a mi Ethernet címünkkel
# válaszoljuk meg:
/usr/sbin/arp -s $5 00:11:22:33:44:55 pub
```

Láthatjuk, hogy az előbbi `slip.login` állomány egy `arp -s $5 00:11:22:33:44:55 pub` paranccsal egészült ki, ami a SLIP szerver ARP táblázatában hoz létre egy ARP bejegyzést. Ez az ARP bejegyzés gondoskodik róla, hogy a SLIP szerver válaszoljon a saját Ethernetes MAC-címével, amikor egy másik IP csomópont a SLIP kliens IP-címe felől érdeklődik.

Amikor a fenti példából indulunk ki, a benne megadott MAC-címet (00:11:22:33:44:55) feltétlenül cseréljük a rendszerünk Ethernet kártyájának MAC-címével, mert különben az “ARP proxy” egyáltalán nem fog működni! A SLIP szerverünk MAC-címét a `netstat -i` paranccsal deríthetjük ki, amelynek a kimenetében a második sor valahogy így néz ki:

```
ed0    1500    <Link>0.2.c1.28.5f.4a          191923          0    129457          0    116
```

Ebből derül ki, hogy az adott rendszer valódi MAC-címe a 00:02:c1:28:5f:4a — az `arp(8)` számára azonban a `netstat -i` kimenetében szereplő pontokat kettőspontokra kell cserélni, és a tagokat ki kell egészíteni kétkarakteres hexadecimális számokká. Az `arp(8)` man oldalán tudhatunk meg ennek részleteiről többet.

Megjegyzés: Amikor létrehozzuk az `/etc/sliphome/slip.login` és `/etc/sliphome/slip.logout` állományokat, akkor ne felejtjük el hozzájuk beállítani a “végrehajtást” engedélyező bitet sem (tehát ilyenkor mindig adjuk ki a `chmod 755 /etc/sliphome/slip.login /etc/sliphome/slip.logout` parancsokat is), különben a `sliplogin` ezeket nem tudja majd elindítani.

27.7.2.4.3. A `slip.logout` beállítása

Az `/etc/sliphome/slip.logout` állományra nincs feltétlenül szükségünk (hacsak nem egy “ARP proxy”-t akarunk csinálni), de ha valamiért mégis el akarjuk készíteni, akkor ehhez a következő alapvető `slip.logout` szkript használható:

```
#!/bin/sh -
#
#      slip.logout
#
# Egy logout állomány a slip vonalhoz. A sliplogin ezt a szkriptet a
# következő paraméterekkel hívja:
#      1      2      3      4      5      6      7-n
#   slipegys. ttyseb. login helyi-cím távoli-cím maszk opc-pmek.
#
/sbin/ifconfig sl$1 down
```

Ha az “ARP proxy” módszert használjuk, és az `/etc/sliphome/slip.logout` felhasználásával akarjuk a SLIP klienshez tartozó ARP bejegyzést törölni, akkor ebből induljunk ki:

```
#!/bin/sh -
#
#      @(#)slip.logout

#
# Egy logout állomány a slip vonalhoz. A sliplogin ezt a szkriptet a
# következő paraméterekkel hívja:
#      1      2      3      4      5      6      7-n
#      slipegys. ttyseb. login helyi-cím távoli-cím maszk opc-pmek.
#
sbin/ifconfig sl$1 down
# Ne válaszoljunk többet a SLIP kliensre vonatkozó ARP kérésekre
/usr/sbin/arp -d $5
```

Az `arp -d $5` parancs eltávolítja az “ARP proxy” működéséhez bejegyzést, amelyet még a `slip.login` szkripttel vettünk fel a SLIP kliens bejelentkezésekor.

Talán felesleges ismételtetésnek tűnhet: az `/etc/sliphome/slip.logout` állománynak létrehozása után állítsuk be a végrehajtásra szóló bitet (vagyis adjuk ki a `chmod 755 /etc/sliphome/slip.logout` parancsot).

27.7.2.5. Az útválasztással kapcsolatos megfontolások

Ha a hálózatunk többi része (lényegében az internet) és a SLIP klienseink között nem az “ARP proxy” módszerrel közvetítjük a csomagokat, akkor a legközelebbi alapértelmezett átjáróhoz minden bizonnyal fel kell vennünk statikus útvonalakat, így a SLIP kliensek alhálózatai a SLIP szerverünkön keresztül ki tudnak jutni.

27.7.2.5.1. Statikus útvonalak

A legközelebbi alapértelmezett átjárók felé nem minden esetben könnyű felvenni statikus útvonalakat (vagy egyes esetekben pedig egyenesen lehetetlen, mivel nincsenek meg hozzá a jogaink). Ha az intézményünkön belül több átjáró is megtalálható, akkor bizonyos útválasztók, például a Cisco és Proteon gyártmányúak esetében nem csak a SLIP alhálózatok felé kell beállítanunk statikus útvonalakat, hanem azt is meg kell mondanunk, hogy ezekről milyen más útválasztók is tudjanak. Pontosan emiatt a statikus útválasztás beüzemeléséhez szükségünk lesz egy kis utánajárásra és próbálgatásra.

Fejezet 28. Elektronikus levelezés

Eredetileg készítette: Bill Lloyd. Átdolgozta: Jim Mock.

28.1. Áttekintés

Az “elektronikus levelezés”, más néven e-mail, a kommunikáció egyik legjobban elterjedt formája. Ebben a fejezetben bemutatjuk, hogyan futtassunk FreeBSD-n levelező szervert, illetve hogyan küldjünk és fogadjunk e-maileket a FreeBSD használatával. Ez azonban semmiképpen sem tekinthető egy teljes referenciának és tulajdonképpen számos fontos tényezőről szót sem ejtünk. A témára úgy kaphatunk egy sokkal átfogóbb rálátást, ha a B Függelékben felsorolt remek könyveket is elolvassuk.

A fejezet elolvasása során megismerjük:

- milyen szoftverkomponensek játszanak szerepet az elektronikus levelek küldésében és fogadásában;
- FreeBSD-ben hol találhatóak a **sendmail** konfigurációs állományai;
- mi a különbség a helyi és távoli postaládák között;
- hogyan akadályozzuk meg, hogy a levelező szerverünk a kéréstlen levélszemetet továbbítson;
- rendszerünkön hogyan telepítsünk és állítsunk be más levelező szervereket a **sendmail** helyett;
- hogyan oldjuk meg a levelező szerverekkel kapcsolatban felmerülő általános problémákat;
- hogyan használjuk az SMTP protokollt az UUCP protokollal;
- hogyan kell rendszerüket csak levélküldésre beállítani;
- hogyan levelezzünk betárcsázós kapcsolattal;
- hogyan növeljük rendszerünk védelmét az SMTP hitelesítésének engedélyezésével;
- hogyan telepítsünk és használjunk a levelek küldésére és fogadására például a **mutt**hoz hasonló levelező klienseket;
- hogyan töltsük le leveleinket egy távoli POP vagy IMAP szerverről;
- hogyan alkalmazzunk automatikusan adott szabályokat vagy szűrőket az érkező levelekre.

A fejezet elolvasása előtt ajánlott:

- az internet-csatlakozásunk megfelelő beállítása (31 fejezet);
- a névfeloldás beállítása (29 fejezet);
- a külső fejlesztésű alkalmazások telepítésének ismerete (4 fejezet).

28.2. Az elektronikus levelezés használata

Öt fontosabb részre bonthatjuk a levelezést. Ezek: a felhasználói program (mail user agent), a levélküldő démon (mail transfer agent), a névfeloldás, a helyi vagy távoli postaláda és természetesen maga a levelező szerver (mail host).

28.2.1. A felhasználói program

Ide soroljuk a különböző parancssoros programokat, mint például a **mutt**, **pine**, **elm** és **mail**, valamint a különféle grafikus alkalmazásokat, mint például a **balsa** és az **xfmail**, csak hogy felsoroljuk néhány újabb, egy webböngészőhöz hasonlóan “kifinomult” eszközt is. Ezek a programok egyszerűen átküldik az elektronikus levelekkel kapcsolatos tranzakciókat a helyi “levelező szervernek” vagy meghívják valamelyik levélküldő démon, esetleg közvetlenül a TCP protokollon keresztül kézbesítenek.

28.2.2. A levélküldő démon

A FreeBSD alapból a **sendmail** nevű programot ajánlja fel erre a célra, de támogat más levelező szervereket is, ezek közül meg is említettünk néhányat ízelítőként:

- **exim**
- **postfix**
- **qmail**

Ez a démon általában két feladatot lát el — a beérkező levelek fogadásáért és a kimenő levelek elküldéséért felelős. *Nem* tartozik azonban a feladatai közé, hogy a POP vagy IMAP protokollokhoz hasonlóan olvashatóvá tegye a leveleinket, illetve csatlakozni engedjen a helyi mbox vagy Maildir formátumú postaládáinkhoz. Ezekhez a műveletekhez egy külön démon szükségeltetik.

FigyelemA **sendmail** régebbi változatai tartalmaznak olyan komoly biztonsági hibákat, amelyek kihasználásával az illetéktelen behatolók helyi és/vagy távoli hozzáférést tudnak szerezni a gépünkön. Az ilyen jellegű problémák elkerülése érdekében igyekezzünk mindig a legfrissebb verzióját használni. Vagy a FreeBSD Portgyűjteményéből telepítsünk fel egy másik levélküldő demont.

28.2.3. Az elektronikus levelek és a névfeloldás

A névfeloldás (Domain Name System, DNS) és a hozzá tartozó **named** démon nagy szerepet játszik az elektronikus levelek továbbításában. A démon a leveleket úgy küldi át az egyik gépről a másikra, hogy a névfeloldáson keresztül megkeresi azt a távoli gépet, amelynek a leveleket címezték. Ez a folyamat szintén végbemegy, amikor egy távoli gépről levelet küldenek a mi szerverünkre.

A DNS valósítja meg a hálózati nevek és az IP-címek összerendelését valamint ez tárolja el a levélküldésre vonatkozó információkat is, amelyeket MX rekordoknak hívnak. Az MX (Mail eXchanger, “levélváltó”) rekord adja meg azt a gépet vagy azokat a gépeket, amelyek az adott névtartományban fogadják a leveleket. Ha a hálózati nevünkhöz vagy tartományunkhoz nem tartozik MX rekord, akkor a levél közvetlenül a gépünkre vándorol feltéve, hogy rendelkezik olyan A rekorddal, amely összerendeli a gépünk nevét az IP-címével.

A **host(1)** parancs használatával az alábbi példához hasonlóan tetszőleges tartomány MX rekordját meg tudjuk nézni:

```
% host -t mx
FreeBSD.org FreeBSD.org mail is handled (pri=10) by
mx1.FreeBSD.org
```

28.2.4. Az elektronikus levelek fogadása

A tartományunkhoz tartozó leveleket fogadását a levelező szerver végzi. Össze gyűjti a tartományunkba küldött összes levelet és ezeket a beállításainktól függően vagy `mbx` (a levelek tárolásának alapértelmezett módja) vagy pedig Maildir formátumban eltárolja. Ahogy eltárolt egy levelet, úgy helyben egyből el is tudjuk olvasni például a `mail(1)` vagy a **mutt** használatával, illetve távolról a POP vagy IMAP és a hasonló protokollokkal tudjuk elérni és begyűjteni. Ezért tehát ha csak a helyi gépen kívánjuk olvasni a leveleinket, akkor ahhoz egyáltalán nem kell POP vagy IMAP szervert telepítenünk.

28.2.4.1. Távoli postaládák elérése a POP és IMAP használatával

A távoli postaládák eléréséhez tudnunk kell csatlakozni egy POP vagy IMAP szerverhez. Ezeken a protokollokon keresztül tudják a felhasználók minden különösebb nehézség nélkül elérni távolról a helyi postaládáikat. Noha a POP és az IMAP segítségével egyaránt el tudjuk így érni a postaládákat, az IMAP használatának mégis több előnye van, íme néhány közülük:

- Az IMAP a levelek leszedése mellett tárolni is képes a távoli szerveren.
- Az IMAP támogat párhuzamos lekéréseket.
- Az IMAP hihetetlenül hasznos tud lenni lassabb összeköttetések esetében, mivel lehetővé teszi a felhasználók számára, hogy csak az üzenetek vázát töltsék le és ne az egészet. Továbbá a szerver és a kliens közti adatmozgás csökkentése érdekében képes bizonyos feladatokat a szerveren elvégezni, például keresni.

Egy POP vagy IMAP szerver telepítéséhez az alábbi lépések megtétele szükséges:

1. Válasszuk ki az igényeinket legjobban kielégítő IMAP vagy POP szervert. A következő POP és IMAP szerverek eléggé elterjedtek és egyben remek példák:
 - **qpopper**
 - **teapop**
 - **imap-uw**
 - **courier-imap**
2. A Portgyűjteményből telepítsük fel a kiválasztott POP vagy IMAP démont.
3. Ha szükséges, akkor a POP vagy IMAP szerver betöltéséhez írjuk át az `/etc/inetd.conf` állományt.

Figyelem Meg kell említenünk, hogy mind a POP és az IMAP az összes információt, tehát belértve a felhasználók neveit és jelszavait titkosítatlan formában továbbítja. Ez azt jelenti, hogy ha ezeket a protokollokat biztonságos módon szeretnénk elérni, akkor az `ssh(1)` használatával hozzunk létre hozzá egy tunnelt és azon keresztül használjuk. Erről részletesebben a 14.11.8 Szakaszban olvashatunk.

28.2.4.2. A helyi postaládák elérése

A helyi postaládákat a szerveren levő levelező kliensek közvetlen használatával érhetjük el. Ilyen alkalmazások például a **mutt** vagy a `mail(1)`.

28.2.5. A levelező szerver

A levelező szerver az a szerver, amely a gépünk vagy akár az egész hálózatunk irányába érkező levelek fogadásáért és elküldéséért felelős.

28.3. A sendmail beállítása

Írta: Christopher Shumway.

A sendmail(8) a FreeBSD alapértelmezett levéltovábbító ügynöke (Mail Transfer Agent, MTA). A **sendmail** feladata fogadni a levelező kliensektől (Mail User Agent, MUA) érkező leveleket és kézbesíteni azokat a konfigurációs állományában megadott megfelelő levelezőknek. A **sendmail** hálózati kapcsolatokat is fogad, képes a helyi postaládákba vagy akár más programoknak is leveleket továbbítani.

A **sendmail** a következő állományban tárolja beállításait:

Állomány	Szerep
/etc/mail/access	A sendmail által engedélyezett hozzáféréseket tároló adatbázis
/etc/mail/aliases	A postaládák álnevei
/etc/mail/local-host-names	Azon nevek felsorolása, amelyek számára a sendmail leveleket fogad
/etc/mail/mailer.conf	A levelező programok beállításai
/etc/mail/mailertable	A levelező programok kézbesítési táblázata
/etc/mail/sendmail.cf	A sendmail központi beállításait tároló állomány
/etc/mail/virtusertable	Virtuális felhasználók és tartományok táblázatai

28.3.1. /etc/mail/access

Az engedélyezett hozzáféréseket tároló adatbázis tartalmazza milyen hálózati neveken vagy IP-címeken lehet elérni a helyi levelező szervert és azok milyen típusú hozzáférést kapnak. A gépek az OK (rendben), REJECT (visszautasít), RELAY (továbbítás) beállításokat alkalmazhatjuk, vagy egyszerűen meghívhatjuk hozzájuk a **sendmail** hibakezelő rutinját egy adott kézbesítési hibával. Ha egy gépet az OK beállítással veszük fel a listára, ami egyébként alapértelmezés, akkor ez a gép levelet tud küldeni egészen addig, amíg a végső cél a helyi gép marad. A REJECT beállítással felsorolt gépek számára semmiféle levelezés nem engedélyezett. Ha pedig egy gép mellett a RELAY beállítás jelenik meg, akkor a szerveren keresztül tetszőleges címre küldhet.

Példa 28-1. A sendmail elérését szabályozó adatbázis beállítása

cyberspammer.com	550 Nem szeretjük a spammereket
FREE.STEALTH.MAILER@	550 Nem szeretjük a spammereket
another.source.of.spam	REJECT
okay.cyberspammer.com	OK
128.32	RELAY

Ebben a példában öt bejegyzést láthatunk. A táblázat bal felének valamelyik sorára illeszkedő küldőkre a táblázatban a sor jobb felén megjelenő cselekvés érvényesül. Az első két sorban a **sendmail** hibakezelő rutinjának adunk át

hibakódokat. A hozzá tartozó üzenet akkor fog megjelenni a távoli gépen, amikor a tőle érkező levél illeszkedik a bal oldali szabályra. Az ezeket követő bejegyzésben visszalökünk minden olyan levelet, amely az internetről egy adott számítógéptől érkezik, például az `another.source.of.spam` címről. A következő bejegyzésben az `okay.cyberspammer.com` címről elfogadjuk a kapcsolódást, ami viszont sokkal pontosabb megjelölés a fentebb szereplő `cyberspammer.com` sornál. A pontosabban kifejtett nevek felülbírálják a kevésbé pontosan megnevezetteket. Végül az utolsó bejegyzésben engedélyezzük a levelek továbbküldését minden olyan gép számára, amelynek címe a `128.32` előtaggal kezdődik. Ezek tehát képesek ezen a levelező szerveren keresztül bárhova leveleket küldeni.

Az állomány módosítása után az adatbázis frissítéséhez mindig le kell futtatnunk egy `make` parancsot az `/etc/mail/` könyvtárban.

28.3.2. `/etc/mail/aliases`

Az álneveket tartalmazó adatbázis virtuális postaládákat sorol fel, amelyek más felhasználókra, állományokra, programokra vagy további álnevekre vonatkozhatnak. Íme néhány példa az `/etc/mail/aliases` állományban szereplő bejegyzésekre:

Példa 28-2. Virtuális postaládák

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket: /dev/null
procmail: "|/usr/local/bin/procmail"
```

A formai szabályok egyszerűek: a kettőspont bal oldalára kell írni azt a postaládát, amely a jobb oldalán levő célokra bomlik. A példa első sorában egyszerűen megfeleltetjük a `root` postaládáját a `localuser` postaládájának, majd ezt a nevet keressük az álnevek adatbázisában. Ha nem találunk már rá illeszkedést, akkor az üzenetet a `localuser` nevű helyi felhasználónak továbbítjuk. A következő sorban címek listáját láthatjuk. Ennek megfelelően a `ftp-bugs` postaláda címére küldött levelek három további helyi postaládára mennek tovább: ezek név szerint a `joe`, `eric` és `paul` felhasználók postaládái. Itt a távoli postaládák `<felhasználó@példa.hu>` alakban adhatóak meg. A következő sor az állományok használatát példázza, ahol konkrétan a `/dev/null` állományba irányítjuk át az adott címre érkező leveleket. Az utolsó sorban pedig a programok használatára láthatunk példát, ahol ebben az esetben a levél egy UNIX-os csövön keresztül a `/usr/local/bin/procmail` szabványos bemenetére kerül.

Ha megváltoztatjuk ezt az állományt, akkor utána az adatbázis frissítéséhez ne felejtsük el meghívni a `make` parancsot az `/etc/mail/` könyvtárban.

28.3.3. `/etc/mail/local-host-names`

Ebben az állományban adhatjuk meg, hogy a `sendmail(8)` milyen hálózati neveket fogadjon el helyi hálózati névként. Ide kell raknunk azokat a tartományokat vagy címeket, amelyekről a **sendmail** leveleket fogad el. Például, ha a levelező szerver az `minta.com` tartományból és a `level.minta.com` címről fogad el leveleket, akkor a `local-host-names` valahogy így fog kinézni:

```
minta.com
level.minta.com
```

Az állomány módosításakor a `sendmail(8)` programot újra kell indítani a változások érvényesítéséhez.

28.3.4. /etc/mail/sendmail.cf

Ahogy a **sendmail** központi konfigurációs állománya, a `sendmail.cf` irányítja a **sendmail** átfogó viselkedését, beleértve mindent az e-mail címek átírásától kezdve a távoli szervereknek küldött elutasító üzenetek küldéséig. Mivel ennyire sokfajta szerepet tölt be egyszerre, ezért ez a konfigurációs állomány meglehetősen összetett és a részletezése meghaladná ennek a leírásnak a határait. Szerencsére az átlagos levelező szerverek esetében ezt az állományt nagyon ritkán kell módosítani.

A **sendmail** központi konfigurációs állománya a **sendmail** lehetőségeit és viselkedését meghatározó `m4(1)` makrókból építhető fel. A pontosabb részleteket a `/usr/src/contrib/sendmail/cf/README` állományban találjuk meg.

Az állomány megváltoztatása után a módosítások érvényesítéséhez újra kell indítani a **sendmail** programot.

28.3.5. /etc/mail/virtusertable

A `virtusertable` állomány képezi le a virtuális tartományokhoz tartozó címeket valódi postaládák címekre. Ezek a postaládák lehetnek helyiek, távoliak, az `/etc/mail/aliases` állományban megadott álnevek vagy állományok.

Példa 28-3. Példa a virtuális tartományok leképezésére

<code>root@minta.com</code>	<code>root</code>
<code>postmaster@minta.com</code>	<code>postmaster@noc.minta.net</code>
<code>@minta.com</code>	<code>joe</code>

A fenti példában megadtunk egy leképezést a `minta.com` tartományhoz. Ez az állomány úgy dolgozódik fel, hogy fentről lefelé illesztődnek a címek, egészen az első egyezésig. Az első bejegyzés szerint a `<root@minta.com>` a helyi `root` felhasználó postaládájára képződik le. A következő bejegyzés szerint a `<postmaster@minta.com>` a `noc.minta.net` címen található `postmaster` nevű felhasználó postaládájára képződik le. Végezetül, ha a `minta.com` címről eddig még semmi sem illeszkedett volna, akkor az utolsó leképezés veszi át, amely az `minta.com` tartományon belül az összes többi címre küldött levelet a helyi `joe` nevű felhasználó postaládájára képezi le.

28.4. A levéltovábbító ügynök megváltoztatása

Írta: Andrew Boothman. Levelei segítségével szolgáltak: Gregory Neil Shapiro.

Ahogy arról már korábban szó esett, a FreeBSD alapból tartalmazza a **sendmail** programot mint levéltovábbító ügynököt (MTA, Mail Transfer Agent). Ennélfogva alapértelmezés szerint ez a felelős a kimenő és beérkező levelek kezeléséért.

Számtalan okból eredően egyes rendszergazdák azonban mégis szeretnék lecserélni a rendszerükhöz tartozó levéltovábbítót. Ennek oka lehet egyszerűen csak annyi, hogy ki akarunk próbálni egy másik programot vagy éppen egy olyan eszközre van szükségünk, amely kizárólag csak máshol található meg. Szerencsére a FreeBSD megkönnyíti ezt a váltást.

28.4.1. Az új levéltovábbító telepítése

A levéltovábbítók széles köre elérhető. A FreeBSD Portgyűjteményéből elindulva sok ilyen programot találhatunk. Természetesen teljesen mindegy, hogy melyik levéltovábbítót választjuk egészen addig, amíg képesek vagyunk FreeBSD alatt rendesen futtatni.

Kezdjük tehát az új levéltovábbító telepítésével. Miután sikerült telepíteni, lehetőségünk van eldönteni, hogy valóban eleget tesz-e az igényeinknek, sőt az új szoftvert még az előtt be tudjuk állítani, hogy átvenné a **sendmail** helyét. Vigyázzunk azonban, hogy az új szoftver telepítésekor ne írjon felül olyan rendszerszintű binárisokat, mint például a `/usr/bin/sendmail`. Másrészt az új levelező szoftvert szolgálatba helyezése előtt mindenképpen fontos megfelelően beállítanunk.

A kiválasztott levéltovábbító beállításával kapcsolatban olvassuk el a hozzá tartozó dokumentációt.

28.4.2. A sendmail letiltása

Figyelem Amikor letiltjuk a **sendmail** kimenő levél szolgáltatását, soha ne felejtjük el pótolni valamilyen más levelező rendszerrel. Ha nem így cselekszünk, akkor például a `periodic(8)` és a hozzá hasonló programok nem lesznek képesek a tőlük megszokott módon e-mailben elküldeni a futásuk eredményét. A rendszer bizonyos részei ráadásul egy működő, **sendmail**-kompatibilis rendszert feltételeznek. Ha letiltása után az alkalmazások továbbra is a **sendmail** segítségével próbálnak levelet küldeni, akkor ez a levél a **sendmail** inaktív sorába kerülhet, ahonnan soha nem kerül kézbesítésre.

A **sendmail** teljes leállításához, beleértve a kimenő levelekhez tartozó szolgáltatást is, a következőket kell megadni az `/etc/rc.conf` állományban:

```
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
```

Ha csak a **sendmail** beérkező levelekre vonatkozó szolgáltatását akarjuk tiltani, akkor ahhoz az `/etc/rc.conf` állományban a következőt állítsuk be:

```
sendmail_enable="NO"
```

A **sendmail** indításával kapcsolatos további beállításokat az `rc.sendmail(8)` man oldalon találjuk.

28.4.3. Az új levéltovábbító elindítása a rendszerrel együtt

Az új levéltovábbítót úgy tudjuk elindítani a rendszerrel együtt, ha az `/etc/rc.conf` állományba felvesszük a következő sort, például a **postfix** esetében:

```
# echo 'postfix_enable="YES"' >> /etc/rc.conf
```

Az új levéltovábbító így most már magától el fog indulni a rendszer indításakor.

28.4.4. A sendmail mint a rendszer alapértelmezett levelező eszközeinek lecserélése

A **sendmail** annyira elterjedt szabványos szoftver a UNIX rendszereken, hogy egyes szoftverek egyszerűen feltételezik a jelenlétét. Emiatt sok levéltovábbítóhoz tartozik egy **sendmail** kompatibilis parancssoros felület is, amellyel igyekeznek megkönnyíteni a **sendmail** “gyors” lecserélését.

Ennek következtében tehát, ha egy másik levelező eszközt használunk, akkor valamilyen módon meg kell bizonyosodnunk róla, hogy a szabványos **sendmail** binárisok, mint például a `/usr/bin/sendmail`, valóban a kiválasztott levéltovábbítót fogják aktiválni. Szerencsére a FreeBSD pontosan emiatt tartalmaz egy mailwrapper(8) nevű rendszert.

Amikor a **sendmail** telepítése szerint működik, valami hasonlót fogunk találni az `/etc/mail/mailer.conf` állományban:

```
sendmail      /usr/libexec/sendmail/sendmail
send-mail     /usr/libexec/sendmail/sendmail
mailq        /usr/libexec/sendmail/sendmail
newaliases   /usr/libexec/sendmail/sendmail
hoststat     /usr/libexec/sendmail/sendmail
purgestat    /usr/libexec/sendmail/sendmail
```

Ez azt jelenti, hogy amikor az itt felsorolt általános parancsok közül lefuttatjuk valamelyiket (például magát a `sendmail` parancsot), akkor a rendszer magától meghívja a `sendmail` néven szereplő wrapper programot, amely pedig a `mailer.conf` alapján kideríti, hogy az adott esetben a `/usr/libexec/sendmail/sendmail` hívására van szükség. Ez a rendszer megkönnyíti az alapértelmezett `sendmail` funkciók helyében lefuttatandó binárisok átállítását.

Így tehát, ha a `/usr/local/supermailer/bin/sendmail-compatible` állományt akarjuk futtatni a megszokott **sendmail** helyében, akkor az `/etc/mail/mailer.conf` állományt a következőképpen kell módosítanunk:

```
sendmail      /usr/local/kevdenclevelező/bin/sendmail-compatible
send-mail     /usr/local/kevdenclevelező/bin/sendmail-compatible
mailq        /usr/local/kevdenclevelező/bin/mailq-compatible
newaliases   /usr/local/kevdenclevelező/bin/newaliases-compatible
hoststat     /usr/local/kevdenclevelező/bin/hoststat-compatible
purgestat    /usr/local/kevdenclevelező/bin/purgestat-compatible
```

28.4.5. A művelet befejezése

Ahogy a céljainknak megfelelően mindent beállítottunk, akkor vagy egyszerűen leállítjuk a **sendmail** neve alatt futó programokat és helyettük elindítjuk az új szoftverhez tartozókat, vagy csak újraindítjuk a gépet. Az újraindítással mellesleg ellenőrizhetjük azt is, hogy jól állítottuk be a rendszerünket és az új levélküldő tényleg elindul a rendszerünkkel együtt.

28.5. A hibák elhárítása

1. Miért kell teljes hálózati neveket megadni a gépemen?

Előfordulhat, hogy a hivatkozni kívánt gép valójában egy másik tartományban szerepel. Például, ha az `ize.mize.edu` gépen vagyunk és a `vagyis` nevű gépet akarjunk innen elérni a `mize.edu` tartományban, akkor a teljes hálózati névvel, vagyis a `vagyis.mize.edu` néven kell rá hivatkoznunk, nem pedig egyszerűen csak `vagyis` néven.

Régebben egyébként ezt a BSD-típusú BIND névfeloldók megengedték. A FreeBSD jelenlegi változatai azonban már olyan **BIND** verziót tartalmaznak, amelyek alapértelmezés szerint már nem engedik a tartományunkon kívüli relatív nevek használatát. Tehát a `vagyis` vagy a `vagyis.ize.mize.edu` gép lesz, vagy a legfelső, gyökér tartományban keresi a rendszer.

Ez eltér a korábbi viselkedéstől, ahol a keresés folytatódott a `vagyis.mize.edu` és `vagyis.edu` tartományokban is. Az RFC 1535 elolvasásából ki fog derülni, hogy miért nem vált be ez a gyakorlat és hogy miért tekinthető még akár biztonsági résznek is.

Ezt a problémát egyébként megoldhatjuk annyival, hogy az `/etc/resolv.conf` állományba a

```
search ize.mize.edu mize.edu
```

sor helyett a

```
domain ize.mize.edu
```

sort írjuk be. Arra viszont ügyeljünk, hogy a keresési rend ne lépje át a “helyi és nyilvános adminisztráció között meghúzó határt”, ahogy azt az RFC 1535 nevezi.

2. A **sendmail** szerint “a levél a saját farkába harap”

Ezt a **sendmail** gyakran ismértelt kérdései között a következőképpen válaszolták meg:

A következő hibaüzenetet kapom:

```
553 MX list for tartomány.net points back to felé.tartomány.net
554 felhasználó@tartomány.net... Local configuration error
```

Hogyan oldható meg ez a probléma?

Azt kértük, hogy a tartományba (például `tartomány.net`) küldött levél az MX rekord felhasználásával egy adott gépre legyen átirányítva (ebben az esetben ez a `felé.tartomány.net`), de a továbbítást végző gép nem ismeri fel magát a `tartomány.net` címen. Vegyük fel a `tartomány.net` tartományt az `/etc/mail/local-host-names` állományba [melyet a 8.10 előtti verziókban `/etc/sendmail.cw` állománynak hívnak] (ha a `FEATURE(use_cw_file)` beállítást használjuk) vagy tegyük hozzá a “`Cw tartomány.net`” sort az `/etc/mail/sendmail.cf` állományhoz.

A **sendmail** GYIK a <http://www.sendmail.org/faq/> címen található meg (angolul) és mindenképpen javasolt elolvasni, ha “fel szeretnénk piszkálni” a levelező rendszerünk beállításait.

3. Hogyan tudok levelező szervert futtatni egy betárcsázós PPP kapcsolat esetében?

Egy helyi hálózaton levő FreeBSD-s gépet akarunk tehát az internethez kapcsolni. Ez a FreeBSD-s gép lesz a helyi hálózat leveleket továbbító átjárója. A PPP kapcsolat nem dedikált.

Legalább két módon meg tudjuk oldani. Az egyik módszer szerint az UUCP használatára lesz szükségünk.

A másik módszer szerint szereznünk kell egy éjjel-nappal üzemelő internetes szervert, amely majd szolgáltatja a másodlagos MX rekordot a tartományunkhoz. Például, ha a cégünk tartománya a *cég.hu* és az internet-szolgáltatónk a *szolgáltató.net* névre beállította a tartományunkhoz a másodlagos MX rekordokat:

<i>cég.hu.</i>	MX	10	<i>cég.hu.</i>
	MX	20	<i>szolgáltató.net.</i>

Végső címzettként csak egy gépet kell megadni (az `/etc/mail/sendmail.cf` állományba a *cég.hu* címhez tegyük hozzá a `Cw cég.hu sort`).

Amikor a leveleket küldeni akaró `sendmail` megpróbál kézbesíteni, először hozzánk (*cég.hu*) próbál csatlakozni a modemes összeköttetésen keresztül. Ez valószínűleg időtúllépéssel befejeződik, mivel nem vagyunk fenn minden pillanatban a neten. A **sendmail** ekkor automatikusan a másodlagos MX rekord által megadott címre küldi a levelet, tehát a szolgáltatónkhoz (*szolgáltató.net*). Ez a másodlagos MX cím próbálja majd időlegesen elérni a gépünket és kézbesíteni a leveleket az elsődleges MX rekord által megadott gépre (*cég.hu*).

A bejelentkezéskor ezért egy hasonló szkriptet kell lefuttatnunk:

```
#!/bin/sh
# Tegyük a /usr/local/bin/pppmyisp állományba:
( sleep 60 ; /usr/sbin/sendmail -q ) &
/usr/sbin/ppp -direct pppmyisp
```

Ha készítünk egy külön bejelentkező szkriptet a felhasználók számára, akkor a `sendmail -qRcég.hu` parancsot is használhatjuk a fenti szkript helyett. Ezzel a *cég.hu* sorában található összes levél azonnal feldolgozásra kerül.

A helyzetet így lehetne még jobban pontosítani:

Az alábbi üzenet a FreeBSD Internet service provider's levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp>) archívumából származik.

```
> we provide the secondary MX for a customer. The customer connects to
> our services several times a day automatically to get the mails to
> his primary MX (We do not call his site when a mail for his domains
> arrived). Our sendmail sends the mailqueue every 30 minutes. At the
> moment he has to stay 30 minutes online to be sure that all mail is
> gone to the primary MX.
>
> Is there a command that would initiate sendmail to send all the mails
> now? The user has not root-privileges on our machine of course.
```

In the "privacy flags" section of `sendmail.cf`, there is a definition `Opgoaway,restrictgrun`

Remove `restrictgrun` to allow non-root users to start the queue processing. You might also like to rearrange the MXs. We are the 1st MX for our customers like this, and we have defined:

```
# If we are the best MX for a host, try directly instead of generating
# local config error.
OwTrue
```

That way a remote site will deliver straight to you, without trying the customer connection. You then send to your customer. Only works for "hosts", so you need to get your customer to name their mail machine "customer.com" as well as "hostname.customer.com" in the DNS. Just put an A record in the DNS for "customer.com".

Az idézet fordítása:

```
> Másodlagos MX rekordot biztosítunk az ügyfeleinknek. Az ügyfelek ezután automatikusan
> csatlakoznak naponta akár többször is a szolgáltatásunkhoz és leszedik az elsődleges MX
> rekordhoz tartozó leveleket. (Nem szólunk neki, amikor a tartományához levél
> érkezik.) A sendmail programunk minden 30 percben elküldi a sorban felhalmozódott
> leveleket. Tehát jelen pillanatban legalább 30 percig fenn kell lennie az ügyfélnek, hogy
> rendben megkapja az elsődlegesre MX rekordra.
>
> Létezik valamilyen parancs a sendmail programhoz, amellyel azonnal lekérhetjük az összes
> levelünket? A felhasználómnak természetesen nincsenek rendszergazdai jogosultságai az adott
> gépen.
```

A sendmail.cf "privacy flags" beállításai között van egy definíció, az Opgoaway, restrictgrun.

Vegyük ki innen a restrictgrun beállítást, amivel a nem root felhasználók is megindíthatják a sor feldolgozását. Valószínűleg az MX-ek átrendezésére is szükség lesz. Mi vagyunk az első MX az ilyen típusú ügyfelek számára, és ezt adtuk meg:

```
# Ha mi vagyunk a legjobb MX a levél számára, akkor ne generáljunk
# helyi beállítási hibát, hanem próbálkozzunk közvetlenül.
OwTrue
```

Ezzel már a távoli gép közvetlenül nekünk küld anélkül, hogy próbálkozna az ügyfél kapcsolatával. Ezt majd továbbküldjük az ügyfélnek. Ez csak hálózati nevek esetében működik, tehát az ügyfélünk el kell neveznie a leveleket fogadó gépét "customer.com"-nak, valamint a fel kell venni a "hostname.customer.com" címet is a DNS-be. Ehhez egyszerűen csak elegendő egy A rekordot betenni a "customer.com"-hoz.

4. Miért kapok folyton Relaying Denied hibát, amikor más gépekről küldök levelet?

A FreeBSD alapértelmezett telepítése során a **sendmail** úgy állítódik be, hogy csak arról a gépről küldhetünk vele levelet, ahol fut. Például, ha POP szerver is elérhető, akkor a felhasználók meg tudják nézni a leveleiket az iskolából, munkából vagy bármilyen más távoli helyről, de leveleket onnan továbbra sem tudnak küldeni. Általában pár pillanattal a próbálkozás után a **MAILER-DAEMON** küldeni fog egy 5.7 Relaying Denied (5.7 A továbbítás nem engedélyezett) üzenetet.

Több lehetőségünk is van ennek megkerülésére. Az a legegyszerűbb módszer, ha az internet-szolgáltatónk címét felvesszük az /etc/mail/relay-domains állományba. Például így:

```
# echo "az.internet.szolgáltató.net" > /etc/mail/relay-domains
```

Az állomány létrehozása vagy módosítása után újra kell indítanunk a **sendmail** programot. Ez remekül működik abban az esetben, ha rendszergazdák vagyunk és nem akarunk a helyi gépről levelet küldeni, vagy egy másik gépen vagy akár másik internet-szolgáltatóval akarunk valamilyen kattingatós levelező programot használni. Olyankor is nagyon hasznos lehet, amikor csak egy vagy két e-mail hozzáférést állítottunk be. Ha egyszerre több címet is fel szeretnénk venni, akkor nyissuk meg ezt az állományt a kedvenc szövegszerkesztőnkkel és írjuk be a tartományokat, soronként egyet:

```
saját.internet.szolgáltató.net
másik.internet.szolgáltató.com
felhasználók-internet.szolgáltató.ja
www.minta.org
```

Innentől kezdve a listában szereplő bármelyik gépről tudunk levelet küldeni (feltéve, hogy az adott felhasználó hozzáfér a gépünkhöz). Ezzel gyönyörűen megoldhatjuk, hogy a felhasználóink képesek legyenek távolról is levelet küldeni a rendszerünkön keresztül anélkül, hogy mások pedig szemetet küldenének át rajtunk.

28.6. Komolyabb témák

A következő szakaszban szóba kerülnek olyan komolyabb témák, mint például a levelek konfigurációja és a levelezés beállítása az egész tartomány számára.

28.6.1. Alapvető beállítások

Alapból képesnek kell lennünk leveleket küldeni külső gépekre egészen addig, amíg az `/etc/resolv.conf` állomány a megfelelő beállításokat tartalmazza vagy egy saját névszervert futtatunk. Ha szeretnénk, hogy a gépünkre érkező levelek elérjék a FreeBSD-s gépünkön futó levéltovábbító ügynököt (például a **sendmail** programot), akkor erre két módszer kínálkozik:

- Futtassunk saját névszervert és hozzunk létre magunknak egy tartományt. Például `FreeBSD.org`.
- Közvetlenül a gépünkre küldessük a leveleket. Ezt úgy tehetjük meg, ha egyből a gépünkhöz tartozó DNS névre küldetjük a leveleket. Például az `enym.FreeBSD.org` címre.

Függetlenül attól, hogy a fentiek közül melyik megoldást választjuk, a levelek csak akkor tudnak eljutni közvetlenül a gépünkre, ha állandó, statikus IP-címmel rendelkezünk (tehát nem dinamikus címmel, amit általában a betárcsázós PPP kapcsolatokhoz szoktak kiosztani). Ha tűzfal mögött vagyunk, akkor valamilyen módon felénk kell irányítani az SMTP forgalmat is. Ha közvetlenül a gépünkön akarjuk fogadni a leveleket, akkor a következő kettő közül az egyik mindenképpen kelleni fog:

- Gondoskodjunk róla, hogy a hozzánk tartozó DNS-ben (legkisebb sorszámú) MX rekord a gépünk IP-címére mutasson.
- Gondoskodjunk róla, hogy a hozzánk tartozó DNS-ben nincs semmilyen MX rekord a gépünkhöz.

A fentiek közül bármelyik elég ahhoz, hogy közvetlenül a gépünkre érkezzen meg a levél.

Próbáljuk ki:

```
# hostname
enym.FreeBSD.org
# host enym.FreeBSD.org
enym.FreeBSD.org has address 204.216.27.XX
```

Ha ezt látjuk, akkor minden gond nélkül lehet küldeni levelet a <nevem@enym.FreeBSD.org> címre (feltételezve, hogy a **sendmail** megfelelően működik az enym.FreeBSD.org címen).

Ha viszont ehhez hasonlót tapasztalunk:

```
# host enym.FreeBSD.org
enym.FreeBSD.org has address 204.216.27.XX
enym.FreeBSD.org mail is handled (pri=10) by kozpont.FreeBSD.org
```

A gépünkre (enym.FreeBSD.org) küldött összes levelet a kozpont szedi össze ugyanazon felhasználói névvel ahelyett, hogy közvetlenül a gépünkre küldeni ezeket.

Az iménti adatokat a DNS szerver határozza meg. A levelek továbbításával kapcsolatos információkat az *MX* mint Mail eXchange DNS-rekord tárolja. Ha nincs ilyen MX rekord, akkor az IP-cím alapján közvetlenül az adott géphez kerül a levél.

Például a freefall.FreeBSD.org MX rekordja hajdanán így nézett ki:

freefall	MX	30	mail.crl.net
freefall	MX	40	agora.rdrop.com
freefall	MX	10	freefall.FreeBSD.org
freefall	MX	20	who.cdrom.com

Láthatjuk, hogy a freefall esetében több MX bejegyzés is szerepel. A legalacsonyabb MX-számú gép fogja kapni az erre a címre beérkező leveleket, amennyiben elérhető. Ha valamilyen okból nem érhető el, akkor helyette ideiglenesen a többiek (melyeket néha csak “tartalék MX-eknek” neveznek) veszik át a levelet és átadják a legalacsonyabb számúnak, amint az újra elérhetővé válik.

A tartalék jelleggel megadott MX gépek akkor érnek ténylegesen valamit, ha teljesen máshonnan csatlakoznak az internethez. Az internet szolgáltató vagy egy ismerősünk gépe valószínűleg minden további nélkül segít ennek megoldásában.

28.6.2. Egy egész tartomány leveleinek kezelése

Egy levelező szerver beállításához valahogy meg kell tudnunk oldálni, hogy a különböző munkaállomásokra küldött levelek közvetlenül hozzá fussanak be. Alapvetően tehát arról lenne szó, hogy a tartományunkon (ez ebben az esetben a *.FreeBSD.org) belüli gépekre címzett levelekre ez a gép “tart igényt” és így ezek ide irányítódnak át, majd a felhasználók erről a központi levelező szerverről kapják meg a leveleiket.

Az életünk megkönnyítéséhez minden felhasználónak létrehozuk a saját *felhasználói nevét* a levelező szerveren is. Ezt az adduser(8) paranccsal gyorsan el is végezhetjük.

A levelező szerver lesz a hálózat összes munkaállomásához kirendelt levélváltó. Ezt a DNS beállításai között így adhatjuk meg:

enym.FreeBSD.org	A	204.216.27.XX	; Munkaállomás
	MX	10 kozpont.FreeBSD.org	; Levelező szerver

Ezzel lényegében az A rekord figyelmen kívül hagyásával átirányítjuk a munkaállomások számára érkező összes levelet a levelező szerverre. A levelek tehát az MX rekord által mutatott címre mennek ki.

Ezt önállóan nem tudjuk elvégezni, hacsak nem futattunk egy saját DNS szerveret. Ha nincsen vagy nem is tudunk DNS szerveret futtatni, akkor ebben a kérdésben egyeztessünk az internet-szolgáltatónkkal vagy bárkivel, aki a DNS beállításaiért felelős.

Ha virtuális e-mail címet is kezelünk, akkor a most következő információ még a hasznunkra lehet. A példa kedvéért most feltesszük, hogy a tartományunkban van egy ügyfelünk, jelen esetben az `ugyfel1.org`, és azt akarjuk, hogy az `ugyfel1.org` címére küldött levelek a saját levelező szerverünkre kerüljenek át, a `level.sajat.com` címre. A DNS-t ehhez így kell beállítani:

```
ugyfel1.org           MX           10           level.sajat.com
```

Ha csak az `ugyfel1.org` levelezését akarjuk kezelni, akkor ahhoz *nem* kell külön A rekord.

Megjegyzés: Vigyázzunk, mert az `ugyfel1.org` csak akkor pingelhető, ha létezik hozzá A rekord.

Befejezésül a levelező szerverünkön futó **sendmail** számára is fel kell tárnunk, hogy milyen tartományokhoz és/vagy hálózati nevekhez fogadjon leveleket. Ezt több módon is elvégezhetjük. A következők bármelyik megfelel erre a célra:

- A `FEATURE(use_cw_file)` használata esetén vegyük fel a címeket az `/etc/mail/local-host-names` állományba. Ha a **sendmail** 8.10 előtti változatai esetében ehhez az `/etc/sendmail.cw` állományra lesz szükségünk.
- Tegyük be a `Cwsajat.cimunk.com` sort az `/etc/sendmail.cf` vagy a **sendmail** 8.10 és későbbi változatai esetén az `/etc/mail/sendmail.cf` állományba.

28.7. SMTP és az UUCP

A FreeBSD-hez tartozó **sendmail** olyan gépek számára lett kialakítva, amelyek közvetlenül az internethez csatlakoznak. Az UUCP használatával levelező rendszerek számára egy másik konfigurációs állományt kell telepíteni a **sendmail** számára.

Az `/etc/mail/sendmail.cf` állítása kézzel egyáltalán nem könnyű. A **sendmail** 8. változata ráadásul a konfigurációs állományokat az m4(1) előfeldolgozó segítségével gyártja le, ahol a tényleges beállítások egy magasabb absztrakciós szinten jelennek meg. Az m4(1) típusú konfigurációs állományok a `/usr/share/sendmail/cf` könyvtárban találhatók. A `cf` alkönyvtárban levő `README` állomány igyekszik a felhasználót bevezetni az m4(1) alapú beállítások világába.

A `mailertable` nevű lehetőség használatával tudjuk a legjobban támogatni az UUCP protokollon keresztüli kézbesítést. Ezzel felépül egy olyan adatbázis, amelyet a **sendmail** fel tud használni a továbbítást érintő döntésekben.

Ehhez elsőként hozzuk is létre a saját `.mc` állományunkat. Ehhez a `/usr/share/sendmail/cf/cf` könyvtár tartalmaz néhány példát. Hívjuk most ezt az állományunkat `ize.mc` néven. A következő módszerrel tudjuk egy valós `sendmail.cf` állománnyá alakítani:

```
# cd /etc/mail
```

```
# make ize.cf
# cp ize.cf /etc/mail/sendmail.cf
```

Egy átlagos .mc állomány egyébként valahogy így épül fel:

```
VERSIONID(`verziószám') OSTYPE(bsd4.4)

FEATURE(accept_unresolvable_domains)
FEATURE(nocanonify)
FEATURE(mailertable, `hash -o /etc/mail/mailertable')

define(`UUCP_RELAY', saját.uucp.relay)
define(`UUCP_MAX_SIZE', 200000)
define(`confDONT_PROBE_INTERFACES')

MAILER(local)
MAILER(smtp)
MAILER(uucp)

Cw      saját.al.nev
Cw      azuucpgepneve.UUCP
```

Az `accept_unresolvable_domains`, `nocanonify` és `confDONT_PROBE_INTERFACES` lehetőségekre hivatkozó sorok megakadályozzák, hogy a levél kézbesítésében a DNS is szerepet játsszon. Az `UUCP_RELAY` az UUCP alapú kézbesítés támogatását engedélyezi. Egyszerűen csak írjunk ide egy internetes hálózati nevet, amely képes feldolgozni az .UUCP áltartomány címeit. Az esetek többségében ide az internet-szolgáltatónk levelek továbbküldéséért felelős gépe kerül.

Miután ezzel végeztünk, szükségünk lesz még az `/etc/mail/mailertable` állományra is. Ha a külvilág felé csak egyetlen összeköttetést használunk a levelekhez, akkor az alábbi pontosan megfelel:

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
.                                uucp-dom:saját.uucp.relay
```

Egy bonyolultabb példa pedig így néz ki:

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
#
horus.interface-business.de    uucp-dom:horus
.interface-business.de         uucp-dom:if-bus
interface-business.de          uucp-dom:if-bus
.heep.sax.de                   smtp8:%1
horus.UUCP                     uucp-dom:horus
if-bus.UUCP                    uucp-dom:if-bus
.                               uucp-dom:
```

Az első három sor azokat a speciális eseteket kezeli, ahol a tartomány felé küldött levelek nem az alapértelmezett úton vesszük tovább, hanem valamelyik UUCP szomszéd felé és így “le tudjuk rövidíteni” a kézbesítés útvonalát. Az ezeket követő sor dolgozza fel a helyi Ethernet tartomány felé SMTP protokollal továbbítható leveleket. Végül az UUCP szomszédokat is felsoroljuk az .UUCP áltartomány jelölése szerint, így megengedjük, hogy a *uucp-szomszéd! címzett* felülbírálja az alapértelmezett szabályokat. Az utolsó sorban mindig egyetlen pont

szerepel, ami minden másra illeszkedik, így az UUCP kézbesítés egy olyan UUCP szomszéd felé halad, amely a világ felé egy univerzális levelező átjárónak tekinthető. A `uucp-dom`: kulcsszó mögött szereplő összes csomópont nevének érvényes UUCP szomszédra kell utalnia, amelyet a `uuname` paranccsal le is tudunk ellenőrizni.

A feladatból már csak annyi maradt hátra, hogy használat előtt ezt az állományt át kell alakítani DBM adatbázis formátumba. Az ehhez szükséges parancsot érdemes `mailertable` állomány elejére bejegyzésben felírni. A `mailertable` megváltoztatásakor mindig le kell futtatni ezt a parancsot.

Utolsó jótanács: ha nem lennénk biztosak valamelyik kézbesítési útvonal működésében, ne felejtjük el a **sendmail** `-bt` beállítását. Ezzel a **sendmail** az ún. *címtesztelő módban* (address test mode) indul el. Gépeljük be, hogy `3,0`, majd írjuk be a tesztelni kívánt címet. Az utolsó sorban láthatjuk a felhasznált belső levéltovábbító ügynököt, a célgépet, amellyel ezt meghívjuk, és a (valószínűleg az átfordított) címet. Innen a **Ctrl+D** billentyűkombinációval léphetünk ki.

```
% sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 ize@pelda.com
canonicalize      input: ize @ pelda . com
...
parse            returns: $# uucp-dom $@ saját.uucp.relay $: ize < @ pelda . com . >
> ^D
```

28.8. Csak küldés beállítása

Készítette: Bill Moran.

Gyakran előfordulhat, hogy csak leveleket akarunk továbbküldeni. Mint például:

- Asztali számítógépünk van, de használni akarunk olyan programokat, mint például a `send-pr(1)`. Ehhez az internet-szolgáltatón keresztül kell továbbküldeni a levelet.
- A számítógépünk egy olyan szerver, amely nem helyben kezeli a leveleket, ezért az összeset átküldi feldolgozásra.

Szinte bármelyik levélküldő ügynök képes betölteni ezt az űrt. Sajnos eléggé bonyolult helyesen beállítani úgy egy bármire képes levélküldőt, hogy egyszerűen csak szabaduljon meg a levelektől. Ilyenkor a **sendmail** vagy a **postfix** használatával tulajdonképpen ágyúval lövünk verébre.

Továbbá, ha egy átlagos internet-hozzáféréssel rendelkezünk, adódhat, hogy a szerződés egyszerűen tiltja a “levelező szerver” futtatását.

Legegyszerűbben úgy tudjuk kielégíteni az ilyen jellegű igényeket, ha feltelepítjük a `mail/ssmtp` portot. A `root` felhasználóval adjuk ki a következő parancsokat:

```
# cd /usr/ports/mail/ssmtp
# make install replace clean
```

Telepítése után a `mail/ssmtp` portot a mindössze négysoros `/usr/local/etc/ssmtp/ssmtp.conf` állománnyal állíthatjuk be:

```
root=valodiemail@minta.com
mailhub=level.minta.com
rewriteDomain=minta.com
```

```
hostname=_GEPNEV_
```

A `root` felhasználó számára feltétlenül egy valódi e-mail címet adjuk meg. A `level.minta.com` helyére az internet-szolgáltatónk kimenő leveleket továbbító szerverét adjuk meg (bizonyos szolgáltatók ezt “kimenő levelező szervernek” vagy “SMTP szervernek” nevezik).

Ne felejtjük el **sendmail** démont sem letiltani, beleértve a kimenő levelek kezelését. Ennek részleteit lásd a 28.4.2 Szakaszban.

A `mail/ssmtp` használatánál még adhatunk meg további beállításokat is. A `/usr/local/etc/ssmtp` állományban vagy az **ssmtp** man oldalán találhatunk példákat és olvashatunk bővebben a témáról.

Az **ssmtp** ilyen fajta beállításával a számítógépünkön levő szoftverek is helyesen fognak működni, miközben nem sértjük meg az internet-szolgáltató előírásait és nem tesszük lehetővé, hogy a számítógépünkről levélszemetet küldhessenek.

28.9. Levelezés betárcsázós kapcsolattal

Ha statikus IP-címünk van, akkor az alapértelmezett beállítások tökéletesen megfelelőek számunkra. Csupán a gépünkhöz tartozó internetes címet kell megadnunk a gépünk nevének és a **sendmail** elvégzi a többit.

Ha viszont dinamikusan kiosztott IP-címmel rendelkezünk és betárcsázós PPP kapcsolaton keresztül csatlakozunk az internethez, akkor valószínűleg az internet-szolgáltató levelező szerverén van egy postaládánk. Most tegyük fel, hogy a internet-szolgáltató tartománya a `szolgaltato.net` és a felhasználói név a `felhasznalo`, a gépünk neve pedig `otthoni.bsdm`, valamint az internet-szolgáltató részéről levelezésre a `relay.szolgaltato.net` gépet használhatjuk.

A postaládánkból úgy tudjuk letölteni a leveleket, ha telepítünk hozzá egy programot. Erre a feladatra a **fetchmail** hibátlanul alkalmas, mivel több különböző protokollt ismer. Ez a program csomagként vagy a Portgyűjteményből (`mail/fetchmail`) is elérhető. Az internet-szolgáltatók erre általában a POP protokollt ajánlják fel. Ha a felhasználói PPP alkalmazást használjuk, állítsuk be az `/etc/ppp/ppp.linkup` állományt a következő módon és így a csatlakozáskor maguktól letöltődnek a leveleink:

```
MYADDR:
!bg su felhasznalo -c fetchmail
```

Ha a **sendmail** segítségével küldjük tovább a leveleket a nem helyi hozzáférések felé (ahogy azt lentebb is láthatjuk), akkor minden bizonnyal a csatlakozáskor arra is szükségünk lesz, hogy a leveleket tároló sor is feldolgozódjon. Ezt úgy oldhatjuk meg, ha az `/etc/ppp/ppp.linkup` állományba a `fetchmail` parancs után a következőt tesszük:

```
!bg su felhasznalo -c "sendmail -q"
```

Ez a példa feltételezi, hogy az `otthoni.bsdm` gépen van egy `felhasznalo` nevű felhasználónk. Az `otthoni.bsdm` gépen a `felhasznalo` felhasználói könyvtárában hozzunk létre egy `.fetchmailrc` állományt:

```
poll szolgaltato.net protocol pop3 fetchall pass TitkosJelszo
```

Ezt az állományt csak és kizárólag a `felhasznalo` olvashatja, mivel szerepel benne a hozzá tartozó `TitkosJelszo`.

Úgy tudunk a megfelelő `from:` fejléccel küldeni, ha felvilágosítjuk a **sendmail** programot, hogy ne az `<felhasznalo@otthoni.bsdm>` címet, hanem a `<felhasznalo@szolgaltato.net>` címet használja. Sőt, a

gyorsítás kedvéért a **sendmail** számára érdemes elárulni, hogy a `relay.szolgaltato.net` címen keresztül küldjön.

A munka elvégzéséhez elegendő az alábbi `.mc` állomány:

```
VERSIONID('otthoni.bsdmc 1.0')
OSTYPE(bsd4.4)dnl
FEATURE(nouucp)dnl
MAILER(local)dnl
MAILER(smtp)dnl
Cwlocalhost
Cwotthoni.bsdmc
MASQUERADE_AS('szolgaltato.net')dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(nocanonify)dnl
FEATURE(nodns)dnl
define('SMART_HOST', 'relay.szolgaltato.net')
Dmotthoni.bsdmc
define('confDOMAIN_NAME', 'otthoni.bsdmc')dnl
define('confDELIVERY_MODE', 'deferred')dnl
```

Az előző szakaszban találhatjuk meg annak a módját, hogy miként varázsoljunk ebből az `.mc` állományból egy `sendmail.cf` állományt. A `sendmail.cf` frissítése után pedig ne felejtjük el a **sendmail** újraindítását!

28.10. Az SMTP hitelesítése

Írta: James Gorham.

Levelező szerverünkön az SMTP protokoll hitelesítésének (SMTP Authentication) engedélyezése több szempontból is előnyökkel bír. Az SMTP hitelesítésének bekapcsolása egy újabb réteget képez a **sendmail** védelmében, és az olyan állandóan mozgásban levő felhasználók számára is megoldást nyújt, akik anélkül képesek használni ugyanazt a levelező szervert, hogy minden alkalommal újrakonfigurálnák a levelező kliensüket.

1. Telepítsük fel a `security/cyrus-sasl2` portot. A `security/cyrus-sasl2` port több fordítási idejű beállítást támogat. Itt most az SMTP hitelesítést fogjuk használni, ezért gondoskodjunk a LOGIN opció engedélyezéséről.
2. A `security/cyrus-sasl2` telepítés után nyissuk meg szerkesztésre a `/usr/local/lib/sasl2/Sendmail.conf` állományt (vagy ha még nem létezne, hozzuk létre), és benne vegyük fel a következő sort:

```
pwcheck_method: saslauthd
```

3. Ezt követően telepítsük a `security/cyrus-sasl2-saslauthd` portot, és tegyük bele az `/etc/rc.conf` állományba ezt a sort:

```
saslauthd_enable="YES"
```

Végezetül indítsuk el a `saslauthd` démont:

```
# /usr/local/etc/rc.d/saslauthd start
```

Ez a démon fog közvetíteni a **sendmail** és a FreeBSD `passwd` adatbázisa közti hitelesítésben. Ezzel elkerülhetjük az új felhasználói nevek és jelszavak felvételét az SMTP hitelesítés használatához, így a hozzáférések és a levelezés jelszava ugyanaz marad.

4. Most pedig írjuk hozzá az alábbi sorokat az `/etc/make.conf` állományhoz:

```
SENDMAIL_CFLAGS=-I/usr/local/include/sasl -DSASL
SENDMAIL_LDFLAGS=-L/usr/local/lib
SENDMAIL_LDADD=-lsasl2
```

Ezek a sorok állítják be a **sendmail** számára, hogy fordítás közben a `cyrus-sasl2` függvényeit használja. A **sendmail** újrafordítása előtt mindenképpen legyen fenn a `cyrus-sasl2` port.

5. A **sendmail** újrafordítását a következő parancsok végrehajtásával intézhetjük el:

```
# cd /usr/src/lib/libsmutil
# make cleandir && make obj && make
# cd /usr/src/lib/libsm
# make cleandir && make obj && make
# cd /usr/src/usr.sbin/sendmail
# make cleandir && make obj && make && make install
```

A **sendmail** fordítása esetén semmilyen problémának nem szabadna előfordulnia, kivéve ha a `/usr/src` könyvtár és a szükséges osztott könyvtárakat nem változtatjuk időközben túlságosan gyakran.

6. A **sendmail** lefordítása és újratelepítése után szerkesszük át az `/etc/mail/freebsd.mc` állományt (vagy azt az `.mc` állományt, amelyet éppen használunk). Sok rendszergazda a `hostname(1)` parancs választát használja fel az `.mc` típusú állományok egyedi elnevezéséhez). Írjuk bele a következő sorokat:

```
dn1 set SASL options
TRUST_AUTH_MECH('GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dn1
define('confAUTH_MECHANISMS', 'GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dn1
```

Ezek állítják be a **sendmail** számára a felhasználók hitelesítésére alkalmas különböző módszereket. Ha a **pwcheck** módszer helyett valami mást akarunk használni, akkor járjunk utána a dokumentációban.

7. Zárásul futassuk le a `make(1)` parancsot az `/etc/mail` könyvtárban. Így lefut az új `.mc` állományunk és létrejön egy `freebsd.cf` (vagy amilyen nevet az `.mc` állománynak megadtunk) `.cf` állomány. Ezután a `make install restart` parancs kiadásával másoltassuk át ezt a `sendmail.cf` helyére és szabályosan indítassuk újra a **sendmail** szolgáltatást. A folyamatról részletesebb tájékoztatást az `/etc/mail/Makefile` állomány tud nyújtani.

Ha eddig minden a legnagyobb rendben történt, akkor most már képesek vagyunk bejelentkezési információt is átadni a levelező kliensnek és elküldeni egy tesztüzenetet. A hibák kiszűréséhez állítsuk a **sendmail** `LogLevel` opcióját az 13 értékre és figyeljük a `/var/log/maillog` állományt.

További felvilágosításért olvassuk el a **sendmail** SMTP hitelesítéssel (<http://www.sendmail.org/~ca/email/auth.html>) foglalkozó oldalát (angolul).

28.11. Levelező kliensek

Készítette: Marc Silver.

A levelező kliens (Mail User Agent, MUA) egy olyan alkalmazás, amelyik elektronikus levelek küldésére és fogadására használható. Azonkívül, ahogy az e-mail “fejlődik” és egyre bonyolultabbá válik, a levelező kliensek is

egyre inkább erősebbé válnak abban a tekintetben, ahogy az e-maileket kezelik. Ezzel együtt a felhasználók is egyre több lehetőséget és rugalmasságot kapnak. A FreeBSD számos levelező klienst támogat, mindegyikük könnyedén telepíthető a FreeBSD Portgyűjteménye segítségével. A felhasználók választhatnak a grafikus kliensek, mint például az **evolution** vagy a **balsa** és a konzolos kliensek, például a **mutt**, **pine** vagy **mail** között, esetleg használhatják a nagyobb szervezetek részéről felkínált webes felületeket is.

28.11.1. mail

A **mail(1)** a FreeBSD alapértelmezett levelező kliense. Egy olyan konzolos alkalmazás, amelyben elérhetjük az e-mailek küldéséhez és fogadásához szükséges összes alapvető funkciót, habár a csatolmányokat csak korlátozottan képes kezelni és csak a helyi postaládákat kezeli.

Annak ellenére, hogy a **mail** ön maga nem képes kommunikálni POP vagy IMAP szerverekkel, az ilyen postaládák tartalmát egy **fetchmail**-szerű alkalmazással (lásd 28.12 Szakasz) le tudjuk tölteni a számára is elérhető helyi **mbx** állományba.

A levelek küldéséhez és fogadásához egyszerűen hívjuk be a **mail** programot a következő módon:

```
% mail
```

Ezután a `/var/mail` könyvtárban található felhasználói postaládánk tartalmát automatikusan beolvassa a **mail** segédprogram. Ha a postaláda üres, akkor a program egyből befejezi futását és közli, hogy nem talált levelet. Amikor viszont tudott beolvasni leveleket, megjelenik egy felület, ahol a beérkezett üzenetek listáját láthatjuk. Az üzenetek automatikusan sorszámozódnak, ahogy ezt az alábbi példa is szemlélteti:

```
Mail version 8.1 6/6/93.  Type ? for help.
"/var/mail/marcs": 3 messages 3 new
>N  1 root@localhost      Mon Mar  8 14:05  14/510  "proba"
   N  2 root@localhost      Mon Mar  8 14:05  14/509  "felhasznaloi hozzaferes"
   N  3 root@localhost      Mon Mar  8 14:05  14/509  "minta"
```

Az üzenetek olvasásának a **t** paranccsal kezdhetünk neki, amelyet az elolvasandó üzenet sorszáma követ. Ebben a példában az első e-mailt nyitjuk meg:

```
& t 1
Message 1:
From root@localhost  Mon Mar  8 14:05:52 2004
X-Original-To: marcs@localhost
Delivered-To: marcs@localhost
To: marcs@localhost
Subject: proba
Date: Mon,  8 Mar 2004 14:05:52 +0200 (SAST)
From: root@localhost (Charlie Root)
```

Ezt az üzenetet probabol kuldom, valaszolj ra, ha megkaptad.

Ahogy az a fenti példából is látszik, a **t** billentyű hatására az üzenet a teljes fejlécével együtt jelenik meg. Az üzenetek listáját a **h** billentyűvel hozhatjuk vissza.

Ha egy levélre válaszolni szeretnénk, akkor ezt a **mail** paranccsal is megtehetjük, vagy az **R** vagy az **r** parancsokkal. Az **R** arra utasítja a **mail** programot, hogy csak az üzenet küldőjének válaszoljon, míg az **r** hatására nem csupán a küldő, hanem az üzenet összes címzettje megkapja a válaszunkat. A parancshoz hozzátűzhetjük egy levél sorszámat

is, ekkor az adott levélre fogunk válaszolni. Miután kiadtuk a parancsot, írjuk meg a válaszunkat és új sorban kezdve zárjuk le az üzenetet egyetlen . beírásával. Valahogy így:

```
& R 1
To: root@localhost
Subject: Re: proba
```

Koszonom, megkaptam a leveledet.

```
.
EOT
```

Új levelet az **m** segítségével tudunk küldeni, ami után meg kell adnunk a címzettet. Egyszerre több címzettet is meg tudunk adni, ha a címzett helyén címeiket egy , karakterrel elválasztva soroljuk fel. Ezután a levél témája is megadható, amit végül a levél szövege követ. Az üzenetet egy új sorban megadott egyetlen . segítségével zárhatjuk le.

```
& mail root@localhost
Subject: Elsajátítottam a mail használatát
```

```
Most már én is tudok levelet írni és fogadni a mail használatával... :)
.
EOT
```

Amikor a mail segédprogramban vagyunk, a ? használatával bármikor segítséget kérhetünk, valamint a mail működésével kapcsolatban a mail(1) man oldalát érdemes felkeresni.

Megjegyzés: Ahogy azt már korábban is említettük, a mail(1) parancsot eredetileg nem készítették fel az csatolt állományok kezelésére, ezért igen gyengén bánt velük. Az újabb levelező kliensek, mint például a **mutt**, a csatolt állományokat sokkal intelligensebb módon kezelik. Ha viszont ragaszkodunk a mail használatához, akkor a `converters/mpack` port használatát érdemes megfontolnunk.

28.11.2. mutt

A **mutt** apró mérete ellenére egy igen komoly levelező kliens és remek lehetőségeket ajánl fel. Íme ízelítésképpen közülük néhány:

- Képes az üzeneteket szálakba rendezni
- Az e-mailek titkosítására és elektronikus aláírására támogatja a PGP használatát
- MIME támogatás
- Maildir támogatás
- Nagyfokú testreszabhatóság

Ezen lehetőségei révén a **mutt** ez egyik legfejlettebb levelező kliens. A **mutt** részletesebb bemutatását a <http://www.mutt.org> címen találjuk (angolul).

A **mutt** stabil változata a `mail/mutt` port használatával telepíthető fel, miközben a fejlesztés alatt levő változatot a `mail/mutt-devel` port telepíti. Miután a portot sikerült felraknunk, a **mutt** az alábbi parancs begépelésével indítható el:

```
% mutt
```

A **mutt** indulása után automatikusan beolvassa a `/var/mail` könyvtárban megtalálható felhasználói postaládát és ha lehetséges, akkor megjeleníti a tartalmát. Ha nincsen levél a felhasználó postaládájában, akkor a **mutt** a felhasználó parancsaira vár. Ezen a képen a **mutt** üzenetlistája látható:

```
q:Quit d:Del u:Undel s:Save m:Mail r:Reply g:Group ?:Help
1 N Mar 09 Super-User ( 1) proba
2 N Mar 09 Super-User ( 1) hozzaferes
3 N Mar 09 Super-User ( 1) minta

--Mutt: /var/mail/marcs [Msgs:3 New:3 1.6K]---(date/date)----- (all)---
```

A levelek elolvasásához egyszerűen csak válasszuk ki a kurzorral és nyomjuk meg az **Enter** billentyűt. Ezután a **mutt** így mutatja a levelet:

```
i:Exit -:PrevPg <Space>:NextPg v:View Attachm. d:Del r:Reply j:Next ?:Help
X-Original-To: marcs@localhost
Delivered-To: marcs@localhost
To: marcs@localhost
Subject: proba
Date: Tue, 9 Mar 2004 10:28:36 +0200 (SAST)
From: Super-User <root@localhost>

Ez egy proba uzenet. Ha megkaptad, kerlek, valaszolj!

--N - 1/1: Super-User test -- (all)
```

Ahogy azt már a `mail(1)` parancsnál is megszokhattuk, a **mutt** is lehetővé teszi, hogy vagy csak a küldőnek, vagy pedig rajta kívül még az összes címzettnek is válaszoljunk. A levél küldőjének az **r** lenyomásával tudunk válaszolni. A csoportos válaszáshoz pedig, ahol tehát a küldőn kívül a címzettek is megkapják a levelünket, a **g** billentyűt kell használni.

Megjegyzés: A **mutt** az e-mailek létrehozásához és megválaszolásához a `vi(1)` szövegszerkesztőt használja. Ezt úgy tudjuk átállítani, ha a könyvtárunkban található `.muttrc` állományban átírjuk az `editor` változót, vagy értéket adunk az `EDITOR` környezeti változónak. A **mutt** beállításáról többet a <http://www.mutt.org> címen tudhatunk meg.

Egy új levél megírásához nyomjuk le az **m** gombot. Miután elláttuk érvényes témával a levelet, a **mutt** elindítja a vi(1) szövegszerkesztőt és nekiláthatunk a levél szövegének. Amint befejeztük, mentjük el és lépünk ki a vi szerkesztőből. Ezután visszakapjuk a **mutt** felületét, ahol a küldendő e-mail összefoglalását láthatjuk. A levelet végül az **y** lenyomásával küldhetjük el. Erre a következő képen láthatunk egy példát:

```

y:Send q:Abort t:To c:CC s:Subj a:Attach file d:Descrip ?:Help
  From: Marc Silver <marcs@localhost>
  To: Super-User <root@localhost>
  Cc:
  Bcc:
  Subject: Re: proba
  Reply-To:
  Fcc:
  Security: Clear

-- Attachments
- 1 1 /tmp/mutt-bsd-c0hobscQ [text/plain, 7bit, us-ascii, 1.1K]

-- Mutt: Compose [Approx. msg size: 1.1K Atts: 1]-----

```

A **mutt** ezenkívül még rengeteg segítséget is tartalmaz, amelyet a legtöbb menüből a **?** gomb lenyomásával érhetünk el. A felső sorban mindig láthatjuk a kiadható parancsok rövid összefoglalását.

28.11.3. pine

A **pine** alapvetően a kezdő felhasználók számára íródott, de számos komolyabb lehetőséget is támogat.

FigyelemA **pine** szoftverrel kapcsolatban a múltban már rengeteg távolról kihasználható sebezhetőség látott napvilágot, és ennek köszönhetően a támadók megfelelően előkészített e-mailek segítségével tetszőleges kódot tudnak futtatni a rendszeren levő helyi felhasználókon keresztül. Noha az összes ilyen *ismert* hibát javították, de a FreeBSD biztonsági tisztje szerint a **pine** kódját biztonság szempontjából annyira hanyag módon írták, hogy további, eddig még felfedezetlen sebezhetőségeket is magában rejt. Ennek megfelelően tehát a **pine** használata mindenkinek csak saját felelősségre javasolt.

A **pine** jelenlegi verziója a mail/pine4 porton keresztül telepíthető. A telepítés lezajlása után a **pine** a következő paranccsal indítható:

```
% pine
```

A **pine** első futtatása során egy üdvözlő üzenetet és egy rövid bemutatkozást jelenít meg, valamint a **pine** fejlesztői arra kéri a felhasználókat, hogy küldjenek nekik egy névtelen üzenetet, amiből le tudják szűrni mennyien használják a kliensüket. A névtelen üzenet elküldéséhez a **Enter** lenyomásával járulhatunk hozzá vagy az **E** használatával enélkül tudunk kilépni a képernyőről. Ezt az üdvözlő képernyőt itt láthatjuk:

```

PINE 4.58  GREETING TEXT                                     No Messages

<<<This message will appear only once>>>

Welcome to Pine ... a Program for Internet News and Email

We hope you will explore Pine's many capabilities. From the Main Menu,
select Setup/Config to see many of the options available to you. Also
note that all screens have context-sensitive help text available.

SPECIAL REQUEST: This software is made available world-wide as a public
service of the University of Washington in Seattle. In order to justify
continuing development, it is helpful to have an idea of how many people
are using Pine. Are you willing to be counted as a Pine user? Pressing
Return will send an anonymous (meaning, your real email address will not
be revealed) message to the Pine development team at the University of
Washington for purposes of tallying.

Pine is a trademark of the University of Washington.

[ALL of greeting text]
? Help      [E] Exit this greeting      [P] PrevPage  [Z] Print
[Ret] [Be Counted!]                  [SpC] NextPage

```

A felhasználó ezután a főmenübe kerül, ahol a kurzorbillentyűkkel minden gond nélkül tudunk mozogni. Ebben a főmenüben a levelek megírására, a leveleket tároló könyvtárak tallózására vagy éppen a címjegyzék karbantartására gyorsbillentyűket is használhatunk. A főmenü alatt szerepel az adott menüben végrehajtható feladatokhoz tartozó gyorsbillentyűk rövid felsorolása.

A **pine** alapértelmezés szerint az **inbox** könyvtárat nyitja meg. A bennelévő üzenetek listájának megtekintéséhez nyomjuk a **I** gombot vagy válasszuk ki a lentihez hasonló módon a **MESSAGE INDEX** menüpontot:

```

PINE 4.58  MAIN MENU                                         Folder: INBOX  3 Messages

?  HELP          - Get help using Pine
C  COMPOSE MESSAGE - Compose and send a message
I  MESSAGE INDEX  - View messages in current folder
L  FOLDER LIST    - Select a folder to view
A  ADDRESS BOOK   - Update address book
S  SETUP          - Configure Pine Options
Q  QUIT          - Leave the Pine program

Copyright 1989-2003. PINE is a trademark of the University of Washington.

? Help      [E] OTHER CMDS  [P] [Index]  [P] PrevCmd      [R] RelNotes
[0]          [Z]           [N] NextCmd    [X] KBLock

```

Az üzenetek listájában az adott könyvtárban található üzenetek láthatjuk, és köztük a kurzorbillentyűkkel mozoghatunk. A kiemelt üzenet az **Enter** lenyomásával olvasható el.

```

PINE 4.58  MESSAGE INDEX                               Folder: INBOX  Message 1 of 3 ANS
A  1 Mar  9 Super-User                                (471) proba
A  2 Mar  9 Super-User                                (479) hozzaferes
A  3 Mar  9 Super-User                                (473) minta

? Help  < FldrList  P PrevMsg  - PrevPage  D Delete  R Reply
0 OTHER CMDS > [ViewMsg] N NextMsg  Spc NextPage  U Undelete  F Forward

```

A lenti képen egy ilyen példa üzenetet láthatunk a **pine** programban. A rendelkezésünkre álló gyorsbillentyűk ilyenkor is a képernyő alján megjelennek referenciaként. Ilyen gyorsbillentyű többek közt az **r** gomb, amelynek hatására a klienssel megválaszolhatjuk a éppen látható üzenetet.

```

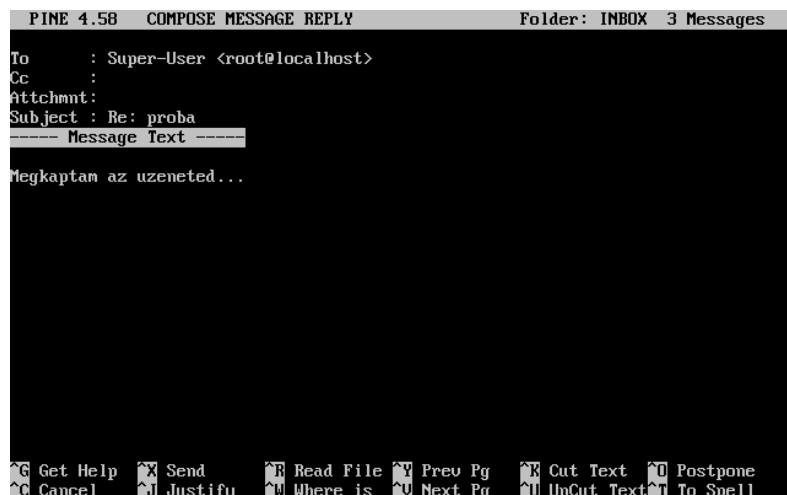
PINE 4.58  MESSAGE TEXT                               Folder: INBOX  Message 1 of 3 ALL ANS
Date: Tue,  9 Mar 2004 10:28:36 +0200 (SAST)
From: Super-User <root@localhost>
To: marcs@localhost
Subject: proba

Ez egy proba uzenet. Ha megkaptad, kerlek, ualaszolj!

[ALL of message]
? Help  < MsgIndex  P PrevMsg  - PrevPage  D Delete  R Reply
0 OTHER CMDS > ViewAtch N NextMsg  Spc NextPage  U Undelete  F Forward

```

A **pine** kliensen belül a **pico** szövegszerkesztő segítségével tudunk megválaszolni egy e-mailt, amely alpból a **pine** mellé települ. A **pico** megkönnyíti a navigációt az üzenetekben és sokkal elnézőbb a kezdő felhasználókkal, mint például a vi(1) vagy a mail(1). Ha befejeztük a választ, az üzenetet a **Ctrl+X** billentyűkombinációval tudjuk elküldeni. A **pine** erre megerősítést fog kérni.



A **pine** alkalmazás a főmenüből elérhető **SETUP** menüpont meghívásával szabható testre. A további részleteket a <http://www.washington.edu/pine> oldalon találhatjuk (angolul).

28.12. A fetchmail használata

Írta: Marc Silver.

A **fetchmail** egy mindentudó IMAP és POP kliens, amely lehetővé teszi a felhasználók számára, hogy automatikusan töltsenek le leveleket távoli IMAP és POP szerverekről és lementsék azokat a helyi postaládáikba. Így a levelek sokkal könnyebben elérhetőek. A **fetchmail** a `mail/fetchmail` port segítségével telepíthető, és számos lehetőséget ajánl fel, többek közt:

- A POP3, APOP, KPOP, IMAP, ETRN és az ODMR protokollok ismerete.
- Képes SMTP használatával levelet továbbítani, és ennek révén a szűrés, továbbküldés és az álnevek használata a megszokott módon működik.
- Démonként futtatva képes adott időközönként ellenőrizni a frissen érkező üzeneteket.
- Képes egyszerre több postaládát is kezelni, majd ezek tartalmát a beállításainak megfelelően továbbküldeni a különböző helyi felhasználóknak.

Noha a **fetchmail** összes lehetőségének aprólékos bemutatása meghaladná ennek a leírásnak a kereteit, azért szót kerítünk néhány alapvető funkciójára. A **fetchmail** segédprogramnak a megfelelő működéshez egy `.fetchmailrc` nevű konfigurációs állományra van szüksége. Ez az állomány tárolja a szerverekre vonatkozó, valamint a bejelentkezéshez szükséges információkat. Az állomány kényes tartalmára tekintettel azt javasoljuk, hogy csak a tulajdonosának engedélyezzük az olvasását:

```
% chmod 600 .fetchmailrc
```

Az alább ismertetésre kerülő `.fetchmailrc` állományban azt láthatjuk, ahogy egyetlen felhasználó postaládáját érjük el a POP protokoll használatával. Arra utasítja a **fetchmail** programot, hogy csatlakozzon a `levelezes.com` címre a `joska` felhasználóval és az `xxx` jelszóval. Ebben a példában feltételezzük, hogy a `joska` nevű felhasználó létezik a rendszerünkben is.

```
poll levelez.es.com protocol pop3 username "joska" password "XXX"
```

A következő példában több POP és IMAP szerverhez csatlakozunk és ahol lehet, több helyi felhasználónak irányítjuk át a leveleket:

```
poll levelez.es.com proto pop3:
user "joska", with password "XXX", is "jozsi" here;
user "andrea", with password "XXXX";
poll levelez.es2.net proto imap:
user "jani", with password "XXXXX", is "hardstuff" here;
```

A **fetchmail** program a `-d` beállítás megadásával démonként is elindítható, amely után meg kell adni (másodpercekben) azt az időközt, aminek elteltével a **fetchmail** lekérdi a `.fetchmailrc` állományban felsorolt szervereket. Az alábbi példában a **fetchmail** 600 másodpercenként kéri el a leveleket:

```
% fetchmail -d 600
```

A **fetchmail** további lehetőségeiről és működéséről a <http://fetchmail.berlios.de/> oldalon olvashatunk (angolul).

28.13. A procmail használata

Írta: Marc Silver.

A **procmail** segédprogram egy hihetetlenül erős alkalmazás, mellyel a beérkező leveleinket tudjuk szűrni. A felhasználók számára olyan "szabályok" megadását teszi lehetővé, amelyekre aztán a rendszer illeszti a bejövő leveleket, és az eredménynek megfelelően elvégez bizonyos feladatokat vagy átirányítja a levelet más postaládákba és/vagy e-mail címekre. A **procmail** a `mail/procmail` porttal telepíthető fel. Miután ez sikerült, akár közvetlenül be is építhetjük a legtöbb levelező kliensbe. Erről az adott levelező kliens dokumentációjában olvashatunk többet. A **procmail** úgy is integrálható, ha a felvesszük a következő sort a **procmail** szolgáltatára igényt tartó felhasználó könyvtárában található `.forward` állományba:

```
"|exec /usr/local/bin/procmail || exit 75"
```

A következő szakaszban láthatjuk a **procmail** néhány alapvető szabályát, valamint ezek rövid leírását. Ezeket a szabályokat a `.procmailrc` állományba kell beleírni, amely szintén a felhasználó könyvtárában leledzik.

Ezen szabályok többsége a `procmailex(5)` man oldalon is olvasható.

A `<felhasznalo@levelez.es.com>` címről érkező leveleket irányítsuk át a `<jocim@levelez.es2.com>` külső címre:

```
:0
* ^From.*felhasznalo@levelez.es.com
! jocim@levelez.es2.com
```

Minden 1000 byte-nál kisebb levelet küldjük át a `<jocim@levelez.es2.com>` külső címre:

```
:0
* < 1000
! jocim@levelez.es2.com
```

Küldjük át az összes `<masik@levelez.es.com>` címre küldött levelet a másik postaládába:

```
:0
* ^Tomasik@levelezes.com
masik
```

Küldjük az összes olyan levelet a /dev/null eszközre, amelyek a témájában szerepel a “Spam” szó:

```
:0
^Subject:.*Spam
/dev/null
```

Egy hasznos szabály, amellyel el tudjuk kapni a FreeBSD.org levelezési listáiról érkező leveleket és el tudjuk raktározni ezeket a saját postaládájukba:

```
:0
* ^Sender:.owner-freebsd-\([^@]+\>@FreeBSD.ORG
{
    LISTNAME=${MATCH}
    :0
    * LISTNAME??^\([^@]+
    FreeBSD-${MATCH}
}
```

Fejezet 29. Hálózati szerverek

Átdolgozta: Murray Stokely.

29.1. Áttekintés

Ebben a fejezetben a UNIX típusú rendszerekben leggyakrabban alkalmazott hálózati szolgáltatások közül fogunk néhányat bemutatni. Ennek során megismerjük a hálózati szolgáltatások különböző típusainak telepítését, beállítását, tesztelését és karbantartását. A fejezet tartalmát folyamatosan példákkal igyekszünk illusztrálni.

A fejezet elolvasása során megismerjük:

- hogyan dolgozzunk az **inetd** démonnal;
- hogyan állítsuk be a hálózati állományrendszereket;
- hogyan állítsunk be egy hálózati információs szervert a felhasználói hozzáférések megosztására;
- hogyan állítsuk be automatikusan a hálózati hozzáférésünket a DHCP használatával;
- hogyan állítsunk be névfeloldó szervereket;
- hogyan állítsuk be az **Apache** webszervert;
- hogyan állítsuk be az állományok átviteléért felelős (FTP) szervert;
- a **Samba** használatával hogyan állítsunk be Windows-os kliensek számára állomány- és nyomtatószervert;
- az NTP protokoll segítségével hogyan egyeztessük az időt és dátumot, hogyan állítsunk be egy időszervert;
- a szabványos naplózó démon, a `syslogd` beállítását hálózati keresztüli naplózásra.

A fejezet elolvasásához ajánlott:

- az `/etc/rc` szkriptek alapjainak ismerete;
- az alapvető hálózati fogalmak ismerete;
- a külső szoftverek telepítésének ismerete (4 fejezet).

29.2. Az inetd “szuperszerver”

Készítette: Chern Lee. A FreeBSD 6.1-RELEASE változatához igazította: A FreeBSD Dokumentációs Projekt.

29.2.1. Áttekintés

Az `inetd(8)` demont gyakran csak “internet szuperszerverként” nevezik, mivel a helyi szolgáltatások kapcsolatainak kezeléséért felelős. Amikor az **inetd** fogad egy csatlakozási kérelmet, akkor eldönti róla, hogy ez melyik programhoz tartozik és elindít egy példányt belőle, majd átadja neki a socketet (az így meghívott program a szabvány bemenetéhez, kimenetéhez és hibajelzési csatornájához kapja meg a socket leíróit). Az **inetd** használatával úgy tudjuk csökkenteni a rendszerünk terhelését, hogy a csak alkalmanként meghívott szolgáltatásokat nem futtatjuk teljesen független önálló módban.

Az **inetd** démonot elsősorban más démonok elindítására használjuk, de néhány triviális protokollt közvetlenül is képes kezelni, mint például a **chargen**, **auth** és a **daytime**.

Ebben a fejezetben az **inetd** beállításának alapjait foglaljuk össze mind parancssoros módban, mind pedig az `/etc/inetd.conf` konfigurációs állományon keresztül.

29.2.2. Beállítások

Az **inetd** működése az `rc(8)` rendszeren keresztül inicializálható. Az `inetd_enable` ugyan alaphól a `NO` értéket veszi fel, vagyis tiltott, de a **sysinstall** használatával már akár a telepítés során bekapcsolható attól függően, hogy a felhasználó milyen konfigurációt választott. Ha tehát a:

```
inetd_enable="YES"
```

vagy

```
inetd_enable="NO"
```

sort tesszük az `/etc/rc.conf` állományba, akkor azzal az **inetd** démonot indíthatjuk el vagy tilthatjuk le a rendszer indítása során. Az

```
# /etc/rc.d/inetd rcvar
```

paranccsal lekérdezhetjük a pillanatnyilag érvényes beállítást.

Emellett még az **inetd** démonnak az `inetd_flags` változón keresztül különböző parancssori paramétereket is át tudunk adni.

29.2.3. Parancssori paraméterek

Hasonlóan a legtöbb szerverhez, az **inetd** viselkedését is befolyásolni tudjuk a parancssorban átadható különböző paraméterekkel. Ezek teljes listája a következő:

```
inetd [-d] [-l] [-w] [-W] [-c maximum] [-C arány] [-a cím | név] [-p állomány] [-R arány] [-s maximum] [konfigurációs állomány]
```

Ezek a paraméterek az `/etc/rc.conf` állományban az `inetd_flags` segítségével adhatóak meg az **inetd** részére. Alapértelmezés szerint az `inetd_flags` értéke `-wW -C 60`, ami az **inetd** által biztosított szolgáltatások TCP protokollon keresztüli wrappelését kapcsolja be, illetve egy IP-címről nem engedi a felkínált szolgáltatások elérését percenként hatvannál többször.

A kezdő felhasználók örömmel nyugtázzhatják, hogy ezeket az alapbeállításokat nem szükséges módosítaniuk. A későbbiekben majd fény derül arra, hogy a kiszolgálás gyakoriságának szabályozása remek védekezést nyújthat túlzottan nagy mennyiségű kapcsolódási kérelem ellen. A megadható paraméterek teljes listája az `inetd(8)` man oldalán olvasható.

`-c maximum`

Az egyes szolgáltatásokhoz egyszerre felépíthető kapcsolatok alapértelmezett maximális számát adja meg. Alaphól ezt a démon nem korlátozza. A `max-child` beállítással ez akár szolgáltatásonként külön is megadható.

-C arány

Korlátozza, hogy egyetlen IP-címről alapból hányszor hívhatóak meg az egyes szolgáltatások egy percen belül. Ez az érték alapból korlátlan. A `max-connections-per-ip-per-minute` beállítással ez szolgáltatásonként is definiálható.

-R arány

Megadja, hogy egy szolgáltatást egy perc alatt mennyiszer lehet meghívni. Ez az érték alapértelmezés szerint 256. A 0 megadásával eltöröljük ezt a típusú korlátozást.

-s maximum

Annak maximumát adja meg, hogy egyetlen IP-címről egyszerre az egyes szolgáltatásokat mennyiszer tudjuk elérni. Alapból ez korlátlan. Szolgáltatásonként ezt a `max-child-per-ip` paraméterrel tudjuk felülbírálni.

29.2.4. Az `inetd.conf` állomány

Az **inetd** beállítását az `/etc/inetd.conf` konfigurációs állományon keresztül végezhetjük el.

Amikor az `/etc/inetd.conf` állományban módosítunk valamit, az **inetd** démon a következő paranccsal meg kell kérnünk, hogy olvassa újra:

Példa 29-1. Az `inetd` konfigurációs állományának újraolvasása

```
# /etc/rc.d/inetd reload
```

A konfigurációs állomány minden egyes sora egy-egy démonról ír le. A megjegyzéseket egy “#” jel vezeti be. Az `/etc/inetd.conf` állomány bejegyzéseinek formátuma az alábbi:

```
szolgáltatás-neve
socket-típusa
protokoll
{wait|nowait}[ /max-child[ /max-connections-per-ip-per-minute[ /max-child-per-ip]]]
felhasználó[:csoport][ /bejelentkezési-osztály]
szerver-program
szerver-program-paramétere
```

Az IPv4 protokollt használó `ftpd(8)` démon bejegyzése például így néz ki:

```
ftp      stream  tcp      nowait  root    /usr/libexec/ftpd      ftpd -l
```

szolgáltatás-neve

Ez az adott démon által képviselt szolgáltatást nevezi meg, amelynek szerepelnie kell az `/etc/services` állományban. Ez határozza meg, hogy az **inetd** milyen porton figyelje a beérkező kapcsolatokat. Ha egy új szolgáltatást hozunk létre, akkor azt először az `/etc/services` állományba kell felvennünk.

csatlakozás-típusa

Ennek az értéke `stream`, `dgram`, `raw`, vagy `seqpacket` lehet. A `stream` típust használja a legtöbb kapcsolat-orientált TCP démon, miközben a `dgram` típus az UDP szállítási protokollt alkalmazó démonok esetében használatos.

protokoll

Valamelyik a következők közül:

Protokoll	Magyarázat
tcp, tcp4	TCP IPv4
udp, udp4	UDP IPv4
tcp6	TCP IPv6
udp6	UDP IPv6
tcp46	TCP IPv4 és v6
udp46	UDP IPv4 és v6

{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]

A `wait|nowait` beállítás mondja meg, hogy az **inetd** démonból meghívott démon saját maga képes-e kezelni kapcsolatokat. A `dgram` típusú kapcsolatok esetében egyértelműen a `wait` beállítást kell használni, miközben a `stream` esetén, ahol általában több szálon dolgozunk, a `nowait` megadása javasolt. A `wait` hatására általában egyetlen démonnak adunk át több socketet, míg a `nowait` minden sockethez egy újabb példányt indít el.

Az **inetd** által indítható példányokat a `max-child` megadásával korlátozhatjuk. Ha tehát például az adott démon számára legfeljebb példány létrehozását engedélyezzük, akkor a `nowait` után `/10` beállítást kell megadnunk. A `/0` használatával korlátlan mennyiségű példányt engedélyezhetünk.

A `max-child` mellett még további két másik beállítás jöhet számításba az egyes démonok által kezelhető kapcsolatok maximális számának korlátozásában. A `max-connections-per-ip-per-minute` az egyes IP-címekről befutó lekezelhető kapcsolatok percenkénti számát szabályozza, így például ha itt a tízes értéket adjuk meg, akkor az adott szolgáltatáshoz egy IP-címről percenként csak tízszer férhetünk hozzá. A `max-child-per-ip` az egyes IP-címekhez egyszerre elindítható példányok számára ír elő egy korlátot. Ezek a paraméterek segítenek megóvni rendszerünket az erőforrások akaratos vagy akaratlan kimerítésétől és a DoS (Denial of Service) típusú támadásoktól.

Ebben a mezőben a `wait` vagy `nowait` valamelyikét kötelező megadni. A `max-child`, `max-connections-per-ip-per-minute` és `max-child-per-ip` paraméterek ellenben elhagyhatóak.

A `stream` típusú több szálon futó démonok a `max-child`, `max-connections-per-ip-per-minute` vagy `max-child-per-ip` korlátozása nélkül egyszerűen csak így adhatóak meg: `nowait`.

Ha ugyanezt a demont tíz kapcsolatra lekorlátozzuk, akkor a következőt kell megadnunk: `nowait/10`.

Amikor pedig IP-címenként 20 kapcsolatot engedélyezünk percenként és mindössze 10 példányt, akkor: `nowait/10/20`.

Az iménti beállítások a `fingerd(8)` démon alapértelmezett paramétereinél is megtalálhatóak:

```
finger stream tcp nowait/3/10 nobody /usr/libexec/fingerd fingerd -s
```

Végezetül engedélyezzük 100 példányt, melyek közül IP-címenként 5 használható: `nowait/100/0/5`.

felhasználó

Ezzel azt a felhasználót adjuk meg, akinek a nevében az adott démon futni fog. Az esetek túlnyomó részében a démonokat a `root` felhasználó futtatja. Láthatjuk azonban, hogy biztonsági okokból bizonyos démonok a `daemon` vagy a legkevesebb joggal rendelkező `nobody` felhasználóval futnak.

szerver-program

A kapcsolat felépülésekor az itt teljes elérési úttal megadott démon indul el. Ha ezt a szolgáltatást maga az **inetd** belsőleg valósítja meg, akkor ebben a mezőben az `internal` értéket adjuk meg.

szerver-program-paraméterei

Ez a `szerver-program` beállítással együtt működik, és ebben a mezőben a démon meghívásakor alkalmazandó paramétereket tudjuk rögzíteni, amelyet a démon nevével kezdünk. Ha a démont a parancssorból a `sajátdémon -d` paranccsal hívnánk meg, akkor a `sajátdémon -d` lesz `szerver-program-paraméterei` beállítás helyes értéke is. Természetesen, ha a démon egy belsőleg megvalósított szolgáltatás, akkor ebben a mezőben is az `internal` fog megjelenni.

29.2.5. Védelem

Attól függően, hogy a telepítés során mit választottunk, az **inetd** által támogatott szolgáltatások egyes része talán alapból engedélyezett is. Amennyiben egy adott démont konkrétan nem használunk, akkor érdemes megfontolni a letiltását. A kérdéses démon sorába tegyünk egy “#” jelet az `/etc/inetd.conf` állományba, majd olvassuk újra az `inetd` beállításait. Egyes démonok, mint például az **fingerd** használata egyáltalán nem ajánlott, mivel a támadók számára hasznos információkat tudnak kiszivároztatni.

Más démonok nem ügyelnek a védelemre, és a kapcsolatokhoz rendelt lejárat idejük túlságosan hosszú vagy éppen nincs is. Ezzel a támadónak lehetősége van lassú kapcsolatokkal leterhelni az adott démont, ezáltal kimeríteni a rendszer erőforrásait. Ha úgy találjuk, hogy túlságosan sok az ilyen kapcsolat, akkor jó ötletnek bizonyulhat a démonok számára a `max-connections-per-ip-per-minute`, `max-child` vagy `max-child-per-ip` korlátozások elrendelése.

Alapértelmezés szerint a TCP kapcsolatok wrappelése engedélyezett. A `hosts_access(5)` man oldalon találhatjuk meg az **inetd** által meghívható különféle démonok TCP-alapú korlátozásainak lehetőségeit.

29.2.6. Egyéb lehetőségek

A **daytime**, **time**, **echo**, **discard**, **chargen** és **auth** szolgáltatások feladatainak mindegyikét maga az **inetd** is képes ellátni.

Az **auth** szolgáltatás a hálózati keresztül azonosítást teszi lehetővé és bizonyos mértékig beállítható. A többi egyszerűen csak kapcsoljuk ki vagy be.

A témában az `inetd(8)` man oldalán tudunk még jobban elmerülni.

29.3. A hálózati állományrendszer (NFS)

Átdolgozta és javította: Tom Rhodes. Írta: Bill Swingle.

A FreeBSD több állományrendszert ismer, köztük a hálózati állományrendszert (Network File System, NFS) is. Az NFS állományok és könyvtárak megosztását teszi lehetővé a hálózaton keresztül. Az NFS használatával a felhasználók és a programok képesek majdnem úgy elérni a távoli rendszereken található állományokat, mintha helyben léteznének.

Íme az NFS néhány legjelentősebb előnye:

- A helyi munkaállomások kevesebb tárterületet használnak, mivel a közös adatokat csak egyetlen számítógépen tároljuk és megosztjuk mindenki között.
- A felhasználóknak nem kell a hálózat minden egyes gépén külön felhasználói könyvtárral rendelkezniük. Ezek ugyanis az NFS segítségével akár egy szerveren is beállíthatók és elérhetővé tehetők a hálózaton keresztül.
- A különböző háttértárak, mint például a floppy lemezek, CD-meghajtók és Zip® meghajtók a hálózaton több számítógép között megoszthatók. Ezzel csökkenteni tudjuk a hálózatunkban szükséges cserélhető lemezes eszközök számát.

29.3.1. Ahogy az NFS működik

Az NFS legalább két fő részből rakható össze: egy szerverből és egy vagy több kliensből. A kliensek a szerver által megosztott adatokhoz képesek távolról hozzáférni. A megfelelő működéshez mindössze csak néhány programot kell beállítani és futtatni.

A szervernek a következő démonokat kell működtetnie:

Démon	Leírás
nfsd	Az NFS démon, amely kiszolgálja az NFS kliensektől érkező kéréseket.
mountd	Az NFS csatlakoztató démonja, amely végrehajtja az nfsd(8) által átküldött kéréseket.
rpcbind	Ez a démon lehetővé teszi az NFS kliensek számára, hogy fel tudják deríteni az NFS szerver által használt portot.

A kliensen is futnia kell egy démonnak, amelynek a neve **nfsiod**. Az **nfsiod** démon az NFS szerver felől érkező kéréseket szolgálja ki. A használata teljesen opcionális, csupán a teljesítményt hivatott javítani, de a normális és helyes működéshez nincs rá szükségünk. Az nfsiod(8) man oldalán erről többet is megtudhatunk.

29.3.2. Az NFS beállítása

Az NFS beállítása viszonylag egyértelműen adja magát. A működéséhez szükséges programok automatikus elindítása csupán néhány apró módosítást igényel az `/etc/rc.conf` állományban.

Az NFS szerveren gondoskodjunk róla, hogy az alábbi beállítások szerepeljenek az `/etc/rc.conf` állományban:

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

A **mountd** magától el fog indulni, ha az NFS szervert engedélyezzük.

A kliensen a következő beállítást kell felvennünk az `/etc/rc.conf` állományba:

```
nfs_client_enable="YES"
```

Az `/etc/exports` állomány adja meg, hogy az NFS milyen állományrendszereket exportáljon (vagy másképpen szólva "osszon meg"). Az `/etc/exports` állományban tehát a megosztani kívánt állományrendszereket kell szerepeltetnünk, és azt, hogy melyik számítógépekkel tudjuk ezeket elérni. A gépek megnevezése mellett a hozzáférésre további megszorításokat írhatunk fel. Ezek részletes leírását az `exports(5)` man oldalon találjuk meg.

Lássunk néhány példát az `/etc/exports` állományban megjelenő bejegyzésekre:

A most következő példákban az állományrendszerek exportálásának finomságait igyekszünk érzékeltetni, noha a konkrét beállítások gyakran a rendszerünkötől és a hálózati konfigurációtól függenek. Például, ha a `/cdrom` könyvtárat akarjuk három gép számára megosztani, akik a szerverrel megegyező tartományban találhatók (ezért nem is kell megadnunk a tartományt) vagy mert egyszerűen megtalálhatók az `/etc/hosts` állományunkban. Az `-ro` beállítás az exportált állományrendszereket írásvédetté teszi. Ezzel a beállítással a távoli rendszerek nem lesznek képesek módosítani az exportált állományrendszer tartalmát.

```
/cdrom -ro gép1 gép2 gép3
```

A következő sorban a `/home` könyvtárat három gép számára osztjuk meg, melyeket IP-címekkel adtunk meg. Ez olyan helyi hálózat esetén hasznos, ahol nem állítottunk be névfeloldást. Esetleg a belső hálózati neveket az `/etc/hosts` állományban is tárolhatjuk. Ezzel utóbbival kapcsolatban a `hosts(5)` man oldalt érdemes fellapoznunk. Az `-alldirs` beállítás lehetővé teszi, hogy az alkönyvtárak is csatlakozási pontok lehessenek. Más szóval, nem fogja csatlakoztatni az alkönyvtárakat, de megengedi a kliensek számára, hogy csak azokat a könyvtárakat csatlakoztassák, amelyeket kell vagy amelyekre szükségünk van.

```
/home -alldirs 10.0.0.2 10.0.0.3 10.0.0.4
```

A következő sorban az `/a` könyvtárat úgy exportáljuk, hogy az állományrendszerhez két különböző tartományból is hozzá lehessen férni. A `-maproot=root` beállítás hatására a távoli rendszer `root` felhasználója az exportált állományrendszeren szintén `root` felhasználóként fogja írni az adatokat. Amennyiben a `-maproot=root` beállítást nem adjuk meg, akkor a távoli rendszeren hiába `root` az adott felhasználó, az exportált állományrendszeren nem lesz képes egyetlen állományt sem módosítani.

```
/a -maproot=root gep.minta.com doboz.haz.org
```

A kliensek is csak a megfelelő engedélyek birtokában képesek elérni a megosztott állományrendszereket. Ezért a klienst ne felejtsük el felvenni a szerver `/etc/exports` állományába.

Az `/etc/exports` állományban az egyes sorok az egyes állományrendszerekre és az egyes gépekre vonatkoznak. A távoli gépek állományrendszerenként csak egyszer adhatóak meg, és csak egy alapértelmezett bejegyzésük lehet. Például tegyük fel, hogy a `/usr` egy önálló állományrendszer. Ennek megfelelően az alábbi bejegyzések az `/etc/exports` állományban érvénytelenek:

```
# Nem használható, ha a /usr egy állományrendszer:
/usr/src kliens
/usr/ports kliens
```

Egy állományrendszerhez, vagyis itt a `/usr` partícióhoz, két export sort is megadtunk ugyanahhoz a kliens nevű géphez. Helyesen így kell megoldani az ilyen helyzeteket:

```
/usr/src /usr/ports kliens
```

Az adott géphez tartozó egy állományrendszerre vonatkozó exportoknak mindig egy sorban kell szerepelniük. A kliens nélkül felírt sorok egyetlen géphez tartozónak fognak számítani. Ezzel az állományrendszerek megosztását tudjuk szabályozni, de legtöbbek számára nem jelent gondot.

Most egy érvényes exportlista következik, ahol a `/usr` és az `/exports` mind helyi állományrendszerek:

```
# Osszuk meg az src és ports könyvtárakat a kliens01 és kliens02 részére, de csak a
# kliens01 férhessen hozzá rendszeradminisztrátori jogokkal:
/usr/src /usr/ports -maproot=root kliens01
/usr/src /usr/ports kliens02
```

```
# A kliensek az /exports könyvtárban teljes joggal rendelkeznek és azon belül
# bármit tudnak csatlakoztatni. Rajtuk kívül mindenki csak írásvédetten képes
# elérni az /exports/obj könyvtárat:
/exports -alldirs -maproot=root      kliens01 kliens02
/exports/obj -ro
```

A **mountd** démonnal az `/etc/exports` állományt minden egyes módosítása után újra be kell olvasatni, mivel a változtatásaink csak így fognak érvényesülni. Ezt megcsinálhatjuk úgy is, hogy küldünk egy HUP (hangup, avagy felfüggesztés) jelzést a már futó démonnak:

```
# kill -HUP `cat /var/run/mountd.pid`
```

vagy meghívjuk a `mountd rc(8)` szkriptet a megfelelő paraméterrel:

```
# /etc/rc.d/mountd onereoad
```

Az 11.7 Szakaszban tudhatunk meg részleteket az `rc` szkriptek használatáról.

Ezek után akár a FreeBSD újraindításával is aktiválhatjuk a megosztásokat, habár ez nem feltétlenül szükséges. Ha `root` felhasználóként kiadjuk a következő parancsokat, akkor azzal minden szükséges programot elindítunk.

Az NFS szerveren tehát:

```
# rpcbind
# nfsd -u -t -n 4
# mountd -r
```

Az NFS kliensen pedig:

```
# nfsiod -n 4
```

Ezzel most már minden készen áll a távoli állományrendszer csatlakoztatására. A példákban a szerver neve `szerver` lesz, valamint a kliens neve `kliens`. Ha csak ideiglenesen akarunk csatlakoztatni egy állományrendszert vagy egyszerűen csak ki akarjuk próbálni a beállításainkat, a kliensen `root` felhasználóként az alábbi parancsot hajtjuk végre:

```
# mount szerver:/home /mnt
```

Ezzel a szerveren található `/home` könyvtárat fogjuk a kliens `/mnt` könyvtárba csatlakoztatni. Ha mindent jól beállítottunk, akkor a kliensen most már be tudunk lépni az `/mnt` könyvtárba és láthatjuk a szerveren található állományokat.

Ha a számítógép indításával automatikusan akarunk hálózati állományrendszereket csatlakoztatni, akkor vegyük fel ezeket az `/etc/fstab` állományba. Erre íme egy példa:

```
szerver:/home    /mnt    nfs      rw      0        0
```

Az `fstab(5)` man megtalálhatjuk az összes többi beállítást.

29.3.3. Zárolások

Bizonyos alkalmazások (például a **mutt**) csak akkor működnek megfelelően, ha az állományokat a megfelelő módon zárolják. Az NFS esetében az **rpc.lockd** használható az ilyen zárolások megvalósítására. Az engedélyezéséhez mind

a szerveren és a kliensen vegyük fel a következő sort az `/etc/rc.conf` állományba (itt már feltételezzük, hogy az NFS szervert és klienst korábban beállítottuk):

```
rpc_lockd_enable="YES"
rpc_statd_enable="YES"
```

A következő módon indíthatjuk el:

```
# /etc/rc.d/lockd start
# /etc/rc.d/statd start
```

Ha nincs szükségünk valódi zárolásra az NFS kliensek és az NFS szerver között, akkor megcsinálhatjuk azt is, hogy az NFS kliensen a `mount_nfs(8)` programnak az `-L` paraméter átadásával csak helyileg végzünk zárolást. Ennek további részleterről a `mount_nfs(8)` man oldalon kaphatunk felvilágosítást.

29.3.4. Gyakori felhasználási módok

Az NFS megoldását a gyakorlatban rengeteg esetben alkalmazzák. Ezek közül most felsoroljuk a legelterjedtebbeket:

- Több gép között megosztunk egy telepítőlemezt vagy más telepítőeszközt. Ez így sokkal olcsóbb és gyakorta kényelmes megoldás abban az esetben, ha egyszerre több gépre akarjuk ugyanazt a szoftvert telepíteni.
- Nagyobb hálózatokon sokkal kényelmesebb lehet egy központi NFS szerver használata, ahol a felhasználók könyvtárait tároljuk. Ezek a felhasználói könyvtárak aztán megoszthatóak a hálózaton keresztül, így a felhasználók mindig ugyanazt a könyvtárat kapják függetlenül attól, hogy milyen munkaállomásról is jelentkeztek be.
- Több géppel is képes így osztozni az `/usr/ports/distfiles` könyvtáron. Ezen a módon sokkal gyorsabban tudunk portokat telepíteni a gépekre, mivel nem kell külön mindegyikre letölteni az ehhez szükséges forrásokat.

29.3.5. Automatikus csatlakoztatás az amd használatával

Készítette: Wylie Stilwell. Újrírta: Chern Lee.

Az `amd(8)` (automatikus csatlakoztató démon, az automatic mounter daemon) önműködően csatlakoztatja a távoli állományrendszereket, amikor azokon belül valamelyik állományhoz vagy könyvtárhoz próbálunk hozzáférni. Emellett az **amd** az egy ideje már inaktív állományrendszereket is automatikusan leválasztja. Az **amd** használata egy remek alternatívát kínál az általában az `/etc/fstab` állományban megjelenő állandóan csatlakoztatott állományrendszerekkel szemben.

Az **amd** úgy működik, hogy kapcsolódik egy NFS szerver `/host` és `/net` könyvtáraihoz. Amikor egy állományt akarunk elérni ezeken a könyvtárakon belül, az **amd** kikeresi a megfelelő távoli csatlakoztatást és magától csatlakoztatja. A `/net` segítségével egy IP-címről tudunk exportált állományrendszereket csatlakoztatni, miközben a `/host` a távoli gép hálózati neve esetében használatos.

Ha tehát a `/host/izemize/usr` könyvtárban akarunk elérni egy állományt, akkor az **amd** démonnak ahhoz először az `izemize` nevű gépről exportált `/usr` könyvtárat kell csatlakoztatnia.

Példa 29-2. Egy exportált állományrendszer csatlakoztatása az amd használatával

Egy távoli számítógép által rendelkezésre bocsátott megosztásokat a `showmount` paranccsal tudjuk lekérdezni. Például az `izemize` gépen elérhető exportált állományrendszereket így láthatjuk:

```
% showmount -e izemize
Exports list on izemize:
/usr                10.10.10.0
/a                  10.10.10.0
% cd /host/izemize/usr
```

Ahogy a példában látjuk is, a `showmount` parancs a `/usr` könyvtárat mutatja megosztásként. Amikor tehát belépünk a `/host/izemize/usr` könyvtárba, akkor **amd** magától megpróbálja feloldani az `izemize` hálózati nevet és csatlakoztatni az elérni kívánt exportált állományrendszert.

Az **amd** az indító szkripteken keresztül az `/etc/rc.conf` alábbi beállításával engedélyezhető:

```
amd_enable="YES"
```

Emellett még az `amd_flags` használatával további paraméterek is átadható az **amd** felé. Alapértelmezés szerint az `amd_flags` tartalmaz az alábbi:

```
amd_flags="-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map"
```

Az `/etc/amd.map` állomány adja meg az exportált állományrendszerek alapértelmezett beállításait. Az `/etc/amd.conf` állományban az **amd** további lehetőségeit konfigurálhatjuk..

Ha többet is szeretnénk tudni a témáról, akkor az `amd(8)` és az `amd.conf(5)` man oldalakat javasolt elolvasnunk.

29.3.6. Problémák más rendszerek használatakor

Készítette: John Lind.

Némely PC-s ISA buszos Ethernet kártyákra olyan korlátozások érvényesek, melyek komoly hálózati problémák keletkezéséhez vezethetnek, különösen az NFS esetében. Ez a nehézség nem FreeBSD-függő, de a FreeBSD rendszereket is érinti.

Ez gond általában majdnem mindig akkor merül fel, amikor egy (FreeBSD-s) PC egy hálózatba kerül többek közt a Silicon Graphic és a Sun Microsystems által gyártott nagyteljesítményű munkaállomásokkal. Az NFS csatlakoztatása és bizonyos műveletek még hibátlanul végrehajthatók, azonban hirtelen a szerver látszólag nem válaszol többet a kliens felé úgy, hogy a többi rendszertől folyamatosan dolgozza felfele a kéréseket. Ez a kliens rendszeren tapasztalható csak, amikor a kliens FreeBSD vagy egy munkaállomás. Sok rendszeren egyszerűen rendesen le sem lehet állítani a klienst, ha a probléma egyszer már felütötte a fejét. Egyedüli megoldás gyakran csak a kliens újraindítása marad, mivel az NFS-ben kialakult helyzetet máshogy nem lehet megoldani.

Noha a "helyes" megoldás az lenne, ha beszereznénk egy nagyobb teljesítményű és kapacitású kártyát a FreeBSD rendszer számára, azonban egy jóval egyszerűbb kerülőút is található a kielégítő működés eléréséhez. Ha a FreeBSD rendszer képviseli a *szerver*t, akkor a kliensnél adjuk meg a `-w=1024` beállítást is a csatlakoztatásnál. Ha a FreeBSD rendszer a *kliens* szerepét tölti be, akkor az NFS állományrendszert az `-r=1024` beállítással csatlakoztassuk róla. Ezek a beállítások az `fstab` állomány negyedik mezőjében is megadhatóak az automatikus csatlakoztatáshoz, vagy manuális esetben a `mount(8)` parancsnak a `-o` paraméterrel.

Hozzá kell azonban tennünk, hogy létezik egy másik probléma, amit gyakran ezzel tévesztenek össze, amikor az NFS szerverek és kliensek nem ugyanabban a hálózatban találhatók. Ilyen esetekben mindenképpen *győződjünk meg róla*, hogy az útválasztók rendesen továbbküldik a működéshez szükséges UDP információkat, különben nem sokat tudunk tenni a megoldás érdekében.

A most következő példákban a gyorsvonat lesz a nagyteljesítményű munkaállomás (felület) neve, illetve a freebsd pedig a gyengébb teljesítményű Ethernet kártyával rendelkező FreeBSD rendszer (felület) neve. A szerveren az /osztott nevű könyvtárat fogjuk NFS állományrendszerként exportálni (lásd exports(5)), amelyet majd a /projekt könyvtárba fogunk csatlakoztatni a kliensen. Minden esetben érdemes lehet még megadnunk a hard vagy soft, illetve bg opciókat is.

Ebben a példában a FreeBSD rendszer (freebsd) lesz a kliens, és az /etc/fstab állományában így szerepel az exportált állományrendszer:

```
gyorsvonat:/osztott /projekt nfs rw,-r=1024 0 0
```

És így tudjuk manuálisan csatlakoztatni:

```
# mount -t nfs -o -r=1024 gyorsvonat:/osztott /projekt
```

Itt a FreeBSD rendszer lesz a szerver, és a gyorsvonat /etc/fstab állománya így fog kinézni:

```
freebsd:/osztott /projekt nfs rw,-w=1024 0 0
```

Manuálisan így csatlakoztathatjuk az állományrendszert:

```
# mount -t nfs -o -w=1024 freebsd:/osztott /projekt
```

Szinte az összes 16 bites Ethernet kártya képes működni a fenti írási vagy olvasási korlátozások nélkül is.

A kíváncsibb olvasók számára eláruljuk, hogy pontosan miért is következik be ez a hiba, ami egyben arra is magyarázatot ad, hogy miért nem tudjuk helyrehozni. Az NFS általában 8 kilobyte-os “blokkokkal” dolgozik (habár kisebb méretű darabkákat is tud készíteni). Mivel az Ethernet által kezelt legnagyobb méret nagyjából 1500 byte, ezért az NFS “blokkokat” több Ethernet csomagra kell osztani — még olyankor is, ha ez a program felsőbb rétegeiben oszthatlan egységként látszik — ezt aztán fogadni kell, összerakni és *nyugtázni* mint egységet. A nagyteljesítményű munkaállomások a szabvány által még éppen megengedett szorossággal képesek ontani magukból az egy egységhez tartozó csomagokat, közvetlenül egymás után. A kisebb, gyengébb teljesítményű kártyák esetében azonban az egymáshoz tartozó, később érkező csomagok ráfutnak a korábban megkapott csomagokra még pontosan azelőtt, hogy elérnék a gépet, így az egységek nem állíthatóak össze vagy nem nyugtázhatóak. Ennek eredményeképpen a munkaállomás egy adott idő múlva megint próbálkozik, de ismét az egész 8 kilobyte-os blokkot küldi el, ezért ez a folyamat a végtelenségig ismétlődik.

Ha a küldendő egységek méretét az Ethernet által kezelt csomagok maximális mérete alá csökkentjük, akkor biztosak lehetünk benne, hogy a teljes Ethernet csomag egyben megérkezik és nyugtázódik, így elkerüljük a holtponthoz.

A nagyteljesítményű munkaállomások természetesen továbbra is küldhetnek a PC-s rendszerek felé túlfutó csomagokat, de egy jobb kártyával az ilyen túlfutások nem érintik az NFS által használt “egységeket”. Amikor egy ilyen túlfutás bekövetkezik, az érintett egységet egyszerűen újra elküldik, amelyet a rákövetkező alkalommal nagy valószínűséggel már tudunk rendesen fogadni, összerakni és nyugtázni.

29.4. Hálózati információs rendszer (NIS/YP)

Írta: Bill Swingle. Írta: Eric Ogren és Udo Erdelhoff.

29.4.1. Mi ez?

A hálózati információs szolgáltatást (Network Information Service, avagy NIS) a Sun Microsystems fejlesztette ki a UNIX (eredetileg SunOS) rendszerek központosított karbantartásához. Mostanra már lényegében ipari szabvánnyá nőtte ki magát, hiszen az összes nagyobb UNIX-szerű rendszer (a Solaris, HP-UX, AIX®, Linux, NetBSD, OpenBSD, FreeBSD stb.) támogatja a NIS használatát.

A NIS régebben sárga oldalak (Yellow Pages) néven volt ismert, de a különböző jogi problémák miatt később ezt a Sun megváltoztatta. A régi elnevezést (és a yp rövidítést) azonban még napjainkban is lehet néhol látni.

Ez egy RPC alapján működő, kliens/szerver felépítésű rendszer, amely az egy NIS tartomány belül levő számítógépek számára teszi lehetővé ugyanazon konfigurációs állományok használatát. Segítségével a rendszergazda a NIS klienseket a lehető legkevesebb adat hozzáadásával, eltávolításával vagy módosításával képes egyetlen helyről beállítani.

Hasonló a Windows NT® tartományaihoz, és habár a belső implementációt tekintve már akadnak köztük jelentős eltérések is, az alapvető funkciók szintjén mégis összevethetőek.

29.4.2. A témához tartozó fogalmak és programok

A NIS telepítése számos fogalom és fontos felhasználói program kerül elő FreeBSD-n, akár egy NIS szervert akarunk beállítani, akár csak egy NIS klienst:

Fogalom	Leírás
NIS tartománynév	A NIS központi szerverei és az összes hozzájuk tartozó kliens (beleértve az alárendelt szervereket) rendelkezik egy NIS tartománynévvel. Hasonló a Windows NT által használt tartománynevekhez, de a NIS tartománynevei semmilyen kapcsolatban nem állnak a névfeloldással.
rpcbind	Az RPC (Remote Procedure Call, a NIS által használt egyik hálózati protokoll) engedélyezéséhez lesz rá szükségünk. Ha az rpcbind nem fut, akkor sem NIS szerver, sem pedig NIS klienst nem tudunk működtetni.
ypbind	A NIS klienst "köti össze" a hozzá tartozó NIS szerverrel. A NIS tartománynevet a rendszertől veszi, és az RPC használatával csatlakozik a szerverhez. Az ypbind a NIS környezet kliens és szerver közti kommunikációjának magját alkotja. Ha az ypbind leáll a kliens gépén, akkor nem tudjuk elérni a NIS szerveret.
ypserv	Csak a NIS szervereken szabad futnia, mivel ez maga a NIS szerver programja. Ha az ypserv(8) leáll, akkor a szerver nem lesz képes tovább kiszolgálni a NIS kéréseket (szerencsére az alárendelt szerverek képesek átvenni ezeket). A NIS bizonyos változatai (de nem az, amelyik a FreeBSD-ben is megjelenik) nem próbálnak meg más szerverekhez csatlakozni, ha bedöglik az aktuális használt szerver. Ezen gyakran egyedül csak a szerver képviselő program (vagy akár az egész szerver) újraindítása segíthet, illetve az ypbind újraindítása a kliensen.

Fogalom	Leírás
<code>rpc.yppasswdd</code>	Ez egy olyan program, amelyet csak a NIS központi szerverein kell csak futtatni. Ez a démon a NIS kliensek számára a NIS jelszavaik megváltoztatását teszi lehetővé. Ha ez a démon nem fut, akkor a felhasználók csak úgy tudják megváltoztatni a jelszavukat, ha bejelentkeznek a központi NIS szerverre.

29.4.3. Hogyan működik?

A NIS környezetekben háromféle gép létezik: a központi szerverek, az alárendelt szerverek és a kliensek. A szerverek képezik a gépek konfigurációs információinak központi tárhelyét. A központi szerverek tárolják ezen információk hiteles másolatát, míg ezt az alárendelt szerverek redundánsan tükrözik. A kliensek a szerverekre támaszkodnak ezen információk beszerzéséhez.

Sok állomány tartalma megosztható ezen a módon. Például a `master.passwd`, a `group` és `hosts` állományokat meg szokták osztani NFS-en. Amikor a kliensen futó valamelyik programnak olyan információra lenne szüksége, amely általában ezekben az állományokban nála megtalálható lenne, akkor helyette a NIS szerverhez fordul.

29.4.3.1. A gépek típusai

•

A központi NIS szerver. Ez a szerver, amely leginkább a Windows NT elsődleges tartományvezérlőjéhez hasonlítható tartja karban az összes, NIS kliensek által használt állományt. A `passwd`, `group`, és összes többi ehhez hasonló állomány ezen a központi szerveren található meg.

Megjegyzés: Egy gép akár több NIS tartományban is lehet központi szerver. Ezzel a lehetőséggel viszont itt most nem foglalkozunk, mivel most csak egy viszonylag kis méretű NIS környezetet feltételezünk.

•

Az alárendelt NIS szerverek. A Windows NT tartalék tartományvezérlőihez hasonlítanak, és az alárendelt NIS szerverek feladata a központi NIS szerveren tárolt adatok másolatainak karbantartása. Az alárendelt NIS szerverek a redundancia megvalósításában segítenek, aminek leginkább a fontosabb környezetekben van szerepe. Emellett a központi szerver terhelésének kiegyenlítését is elvégzik. A NIS kliensek elsőként mindig ahhoz a NIS szerverhez csatlakoznak, amelytől először választ kapnak, legyen akár az egy alárendelt szerver.

•

A NIS kliensek. A NIS kliensek, hasonlóan a Windows NT munkaállomásokhoz, a NIS szerveren (amely a Windows NT munkaállomások esetében a tartományvezérlő) keresztül jelentkeznek be.

29.4.4. A NIS/YP használata

Ebben a szakaszban egy példa NIS környezetet állítunk be.

29.4.4.1. Tervezés

Tegyük fel, hogy egy aprócska egyetemi labor rendszergazdái vagyunk. A labor, mely 15 FreeBSD-s gépet tudhat magáénak, jelen pillanatban még semmilyen központosított adminisztráció nem létezik. Mindegyik gép saját `/etc/passwd` és `/etc/master.passwd` állománnyal rendelkezik. Ezeket az állományokat saját kezűleg kell szinkronban tartani. Tehát ha most felveszünk egy felhasználót a laborhoz, akkor az `adduser` parancsot mind a 15 gépen ki kell adni. Egyértelmű, hogy ez így nem maradhat, ezért úgy döntöttük, hogy a laborban NIS-t fogunk használni, és két gépet kinevezünk szervernek.

Az iméntieknek megfelelően a labor most valahogy így néz ki:

A gép neve	IP-cím	A gép szerepe
ellington	10.0.0.2	központi NIS
coltrane	10.0.0.3	alárendelt NIS
basie	10.0.0.4	tanszéki munkaállomás
bird	10.0.0.5	kliensgép
cli[1-11]	10.0.0.[6-17]	a többi kliensgép

Ha még nincs tapasztalatunk a NIS rendszerek összeállításában, akkor először jó ötlet lehet végiggondolni, miként is akarjuk kialakítani. A hálózatunk méretétől függetlenül is akadnak olyan döntések, amelyeket mindenképpen meg kell hoznunk.

29.4.4.1.1. A NIS tartománynév megválasztása

Ez nem az a “tartománynév”, amit megszokhattunk. Ennek a pontos neve “NIS tartománynév”. Amikor a kliensek kérnek valamilyen információt, akkor megadják annak a NIS tartománynak a nevét is, amelynek részei. Így tud egy hálózaton több szerver arról dönteni, hogy melyikük melyik kérést válaszolja meg. A NIS által használt tartománynévre tehát inkább úgy érdemes gondolni, mint egy valamilyen módon összetartozó gépek közös nevére.

Előfordul, hogy egyes szervezetek az interneten is nyilvántartott tartományneveket választják NIS tartománynévnek. Ez alapvetően nem ajánlott, mivel a hálózati problémák felderítése közben félreértéseket szülhet. A NIS tartománynévnek a hálózatunkon belül egyedinek kell lennie, és lehetőleg minél jobban írja le az általa csoportba sorolt gépeket. Például a Kis Kft. üzleti osztályát tegyük a “kis-uzlet” NIS tartományba. Ebben a példában most a `proba-tartomany` nevet választottuk.

A legtöbb operációs rendszer azonban (köztük a SunOS) a NIS tartománynevet használja internetes tartománynévként is. Ha a hálózatunkon egy vagy több ilyen gép is található, akkor a NIS tartomány nevének az internetes tartománynevet *kell* megadnunk.

29.4.4.1.2. A szerverek fizikai elvárásai

Nem árt néhány dolgot fejben tartani, amikor a NIS szervernek használt gépet kiválasztjuk. Az egyik ilyen szerencsétlen dolog az a szintű függőség, ami a NIS kliensek felől megfigyelhető a szerverek felé. Ha egy kliens nem tudja a NIS tartományon belül felvenni a kapcsolatot valamelyik szerverrel, akkor az a gép könnyen megbízhatatlanná válhat. Felhasználói- és csoportinformációk nélkül a legtöbb rendszer egy időre le is merevedik. Ennek figyelembevételével tehát olyan gépet kell szervernek választanunk, amelyet nem kell gyakran újraindítani, és nem végzünk rajta semmilyen komoly munkát. A célnak legjobban megfelelő NIS szerverek valójában olyan gépek, amelyek egyedüli feladata csak a NIS kérések kiszolgálása. Ha a hálózatunk nem annyira leterhelt, akkor még a NIS

szerver mellett más programokat is futtathatunk, de ne feledjük, hogy ha a NIS szolgáltatás megszűnik, akkor az az összes NIS kliensen éreztetni fogja kedvezőtlen hatását.

29.4.4.2. A NIS szerverek

A NIS rendszerben tárolt összes információ általános példánya egyetlen gépen található meg, amelyet a központi NIS szervernek hívunk. Az információk tárolására szánt adatbázis pedig NIS táblázatoknak (NIS map) nevezzük. FreeBSD alatt ezek a táblázatok a `/var/yp/tartománynév` könyvtárban találhatóak, ahol a *tartománynév* a kiszolgált NIS tartományt nevezi meg. Egyetlen NIS szerver egyszerre akár több tartományt is kiszolgálhat, így itt több könyvtár is található, minden támogatott tartományhoz egy. Minden tartomány saját, egymástól független táblázatokkal rendelkezik.

A központi és alárendelt NIS szerverek az `ypserv` démon segítségével dolgozzák fel a NIS kéréseket. Az `ypserv` felelős a NIS kliensektől befutó kérések fogadásáért, és a kért tartomány valamint táblázat nevéből meghatározza az adatbázisban tárolt állományt, majd innen visszaküldi a hozzá tartozó adatot a kliensnek.

29.4.4.2.1. A központi NIS szerver beállítása

A központi NIS szerver beállítása viszonylag magától értetődő, de a nehézségét az igényeink szabják meg. A FreeBSD alapból támogatja a NIS használatát. Ezért mindössze annyit kell tennünk, hogy a következő sorokat betesszük az `/etc/rc.conf` állományba, és a FreeBSD gondoskodik a többitől.

1.

```
nisdomainname="proba-tartomany"
```

Ez a sor adja meg a hálózati beállítások (vagy például az újraindítás) során a NIS tartomány nevét, amely a korábbiak szerint itt most a `proba-tartomany`.

2.

```
nis_server_enable="YES"
```

Ezzel utasítjuk a FreeBSD-t, hogy a hálózati alkalmazások következő indításakor a NIS szerver is aktiváljon.

3.

```
nis_yppasswdd_enable="YES"
```

Ezzel engedélyezzük az `rpc.yppasswdd` demont, amely a korábban említettek szerint lehetővé teszi a felhasználók számára, hogy a közvetlenül a kliensekről változtassák meg a NIS jelszavukat.

Megjegyzés: A konkrét NIS beállításainktól függően további bejegyzések felvételére is szükségünk lehet. Erre később még az olyan NIS szervereknél, amelyek egyben NIS kliensek, vissza fogunk térni.

Miután ezeket beállítottuk, rendszeradminisztrátorként adjuk ki az `/etc/netstart` parancsot. Az `/etc/rc.conf` állományban szereplő adatok alapján mindent beállít magától. Még mielőtt inicializálnánk a NIS táblázatokat, indítsuk el manuálisan az **ypserv** demont:

```
# /etc/rc.d/ypserv start
```

29.4.4.2.2. A NIS táblázatok inicializálása

A NIS táblázatok lényegében a `/var/yp` könyvtárban tárolt adatbázisok. A központi NIS szerver `/etc` könyvtárában található konfigurációs állományokból állítódnak elő, egyetlen kivétellel: ez az `/etc/master.passwd` állomány. Ennek megvan a maga oka, hiszen nem akarjuk a `root` és az összes többi fontosabb felhasználóhoz tartozó jelszót az egész NIS tartománnyal megosztani. Ennek megfelelően a NIS táblázatok inicializálásához a következőt kell tennünk:

```
# cp /etc/master.passwd /var/yp/master.passwd
# cd /var/yp
# vi master.passwd
```

El kell távolítanunk az összes rendszerszintű (`bin`, `tty`, `kmem`, `games`, `stb`), és minden olyan egyéb hozzáférést, amelyeket nem akarjuk közvetíteni a NIS kliensek felé (például a `root` és minden más nullás, vagyis rendszeradminisztrátori azonosítóval ellátott hozzáférést).

Megjegyzés: Gondoskodjunk róla, hogy az `/var/yp/master.passwd` állomány sem a csoport, sem pedig bárki más számára nem olvasható (600-as engedély)! Ennek beállításához használjuk az `chmod` parancsot, ha szükséges.

Ha végeztünk, akkor már tényleg itt az ideje inicializálni NIS táblázatainkat. A FreeBSD erre egy `ypinit` nevű szkriptet ajánl fel (erről a saját man oldalán tudhatunk meg többet). Ez a szkript egyébként a legtöbb UNIX típusú operációs rendszeren megtalálható, de nem az összesen. A Digital UNIX/Compaq Tru64 UNIX rendszereken ennek a neve `ypsetup`. Mivel most a központi NIS szerver táblázatait hozzuk létre, azért az `ypinit` szkriptnek át kell adnunk a `-m` opciót is. A NIS táblázatok előállításánál feltételezzük, hogy a fentebb ismertetett lépéseket már megtettük, majd kiadjuk ezt a parancsot:

```
ellington# ypinit -m proba-tartomany
Server Type: MASTER Domain: proba-tartomany
Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.
Do you want this procedure to quit on non-fatal errors? [y/n: n] n
Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
At this point, we have to construct a list of this domains YP servers.
rod.darktech.org is already known as master server.
Please continue to add any slave servers, one per line. When you are
done with the list, type a <control D>.
master server   : ellington
next host to add: coltrane
next host to add: ^D
The current list of NIS servers looks like this:
ellington
coltrane
Is this correct? [y/n: y] y
```

[.. a táblázatok generálása ..]

NIS Map update completed.
ellington has been setup as an YP master server without any errors.

Az üzenetek fordítása:

```
A szerver típusa: KÖZPONTI, tartomány: proba-tartomány
Az YP szerver létrehozásához meg kell válaszolni néhány kérdést az
eljárás megkezdése előtt.
Szeretnénk, ha az eljárás megszakadna a nem végzetes hibák esetén is? [i/n: n] n
Rendben, akkor ne felejtsük el manuálisan kijavítani a hibát, ha
valamivel gond lenne. Ha nem tesszük meg, akkor előfordulhat, hogy
valami nem fog rendesen működni. Most össze kell állítanunk egy listát
a tartomány YP szervereiről.
Jelenleg a rod.darktech.org a központi szerver.
Kérjünk, adjon meg további alárendelt szervereket, soronként egyet.
Amikor ezt befejeztük, a <control D> lenyomásával tudunk
kilépni.
központi szerver : ellington
következő gép    : coltrane
következő gép    : ^D
A NIS szerverek listája jelenleg a következő:
ellington
coltrane
Ez megfelelő? [i/n: i] i

[ .. a táblázatok generálása .. ]

A NIS táblázatok sikeresen frissültek.
Az elligon szervert minden hiba nélkül sikerült központi szerverként
beállítani.
```

Az ypinit a /var/yp/Makefile.dist állományból létrehozza a /var/yp/Makefile állományt. Amennyiben ez létrejött, az állomány feltételezi, hogy csak FreeBSD-s gépek részvételével akarunk kialakítani egy egyszerveres NIS környezetet. Mivel a proba-tartomány még egy alárendelt szervert is tartalmaz, ezért át kell írunk a /var/yp/Makefile állományt:

```
ellington# vi /var/yp/Makefile
```

Ezt a sort kell megjegyzésbe tennünk:

```
NOPUSH = "True"
```

(ha még nem lenne úgy).

29.4.4.2.3. Az alárendelt NIS szerverek beállítása

Az alárendelt NIS szerverek beállítása még a központiál is egyszerűbb. Jelentkezzünk be az alárendelt szerverre és az eddigieknek megfelelően írjuk át az /etc/rc.conf állományt. Az egyetlen különbség ezúttal csupán annyi lesz, hogy az ypinit lefuttatásakor a -s opciót kell megadnunk (mint slave, vagyis alárendelt). A -s opció használatához a központi NIS szerver nevét is át kell adnunk, ezért a konkrét parancs valahogy így fog kinézni:

```
coltrane# ypinit -s ellington proba-tartomány
```

```
Server Type: SLAVE Domain: test-domain Master: ellington
```

Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.

Do you want this procedure to quit on non-fatal errors? [y/n: n] **n**

Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
There will be no further questions. The remainder of the procedure
should take a few minutes, to copy the databases from ellington.
Transferring netgroup...

```
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byuser...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byhost...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring group.bygid...
ypxfr: Exiting: Map successfully transferred
Transferring group.byname...
ypxfr: Exiting: Map successfully transferred
Transferring services.byname...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.byname...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.byname...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring netid.byname...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring ypservers...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byname...
ypxfr: Exiting: Map successfully transferred
```

coltrane has been setup as an YP slave server without any errors.
Don't forget to update map ypservers on ellington.

Most már lennie kell egy `/var/yp/proba-tartomany` nevű könyvtárunknak is. A központi NIS szerver táblázatainak másolata itt fognak tárolódni. Ezeket soha ne felejtjük el frissen tartani. Az alárendelt szervereken a következő `/etc/crontab` bejegyzések pontosan ezt a feladatot látják el:

```
20      *      *      *      *      root    /usr/libexec/ypxfr passwd.byname
21      *      *      *      *      root    /usr/libexec/ypxfr passwd.byuid
```

Ez a két sor gondoskodik róla, hogy az alárendelt szerverek ne felejtsek el egyeztetni a táblázataikat a központi szerver táblázataival. Ezek a bejegyzések nem nélkülözhetetlenek a megfelelő működéshez, mivel a központi szerver automatikusan feltölti az alárendelt szerverekre a létrejött változásokat. Mivel azonban a jelszavak létfontosságúak a szervertől függő rendszerek számára, ezért ajánlott explicit módon is előírni a frissítést. Ez a forgalmasabb hálózatokon nagyobb jelentőséggel bír, mivel ott a táblázatok frissítése nem mindig fejeződik be rendesen.

Most pedig futassuk le a `/etc/netstart` parancsot az alárendelt szervereken is, amivel így elindul a NIS szerver.

29.4.4.3. A NIS kliensek

A NIS kliens az `ypbind` démon segítségével egy kötésnek (`bind`) nevezett kapcsolatot épít ki egy adott NIS szerverrel. Az `ypbind` ellenőrzi a rendszer alapértelmezett tartományát (ezt a `domainname` paranccsal állítottunk be), majd RPC kéréseket kezd szórni a helyi hálózaton. Ezek a kérések annak a tartománynak a nevét tartalmazzák, amelyhez az `ypbind` megpróbál kötést létrehozni. Ha az adott tartomány kiszolgálására beállított szerver észleli ezeket a kéréseket, akkor válaszol az `ypbind` démonnak, amely pedig feljegyzi a szerver címét. Ha több szerver is elérhető (például egy központi és több alárendelt), akkor az `ypbind` az elsőként válaszoló címét fogja rögzíteni. Innentől kezdve a kliens közvetlenül ennek a szervernek fogja küldeni a NIS kéréseit. Az `ypbind` időnként “megpingeli” a szervert, hogy meggyőződjön az elérhetőségéről. Ha az `ypbind` egy adott időn belül nem kap választ a ping kéréseire, akkor megszünteti a kötést a tartományhoz és nekilát keresni egy másik szervert.

29.4.4.3.1. A NIS kliensek beállítása

Egy FreeBSD-s gépet NIS kliensként meglehetősen egyszerűen lehet beállítani.

- Nyissuk meg az `/etc/rc.conf` állományt és a NIS tartománynév beállításához, valamint az `yypbind` elindításához a következőket írjuk bele:

```
nisdomainname="proba-tartomany"  
nis_client_enable="YES"
```
- A NIS szerveren található jelszavak importálásához távolítsuk el az összes felhasználói hozzáférést az `/etc/master.passwd` állományunkból és a `vipw` segítségével adjuk hozzá az alábbi sort az állomány végéhez:

```
+:::~::~:
```

Megjegyzés: Ez a sor beenged bárkit a rendszerünkre, akinek a NIS szervereken van érvényes hozzáférése. A NIS klienseket ezzel a sorral sokféle módon tudjuk állítani. A hálózati csoportokról szóló szakaszban találunk majd erről több információt. A téma mélyebb megismeréséhez az *O'Reilly Managing NFS and NIS* című könyvét ajánljuk.

Megjegyzés: Legalább helyi hozzáférést (vagyis amit nem NIS-en keresztül importálunk) azonban mindenképpen hagyjunk meg az `/etc/master.passwd` állományunkban, és ez a hozzáférés legyen a `wheel` csoport tagja. Ha valami gond lenne a NIS használatával, akkor ezen a hozzáférésen keresztül tudunk a gépre távolról bejelentkezni, majd innen `root` felhasználóra váltva megoldani a felmerült problémákat.

3. A NIS szerverről az összes lehetséges csoport-bejegyzést az `/etc/group` állományban így tudjuk importálni:

```
+ : * : :
```

Miután elvégeztük ezeket a lépéseket, képesek leszünk futtatni az `ypcat passwd` parancsot, és látni a NIS szerver jelszavakat tartalmazó táblázatát.

29.4.5. A NIS biztonsága

Általában tetszőleges távoli felhasználó küldhet RPC kéréseket az `ypserv(8)` számára és kérheti le a NIS táblázatok tartalmát, feltéve, hogy ismeri a tartomány nevét. Az ilyen hitelesítés nélküli műveletek ellen az `ypserv(8)` úgy védekezik, hogy tartalmaz egy “`securenets`” nevű lehetőséget, amellyel az elérhetőségüket tudjuk leszűkíteni gépek egy csoportjára. Az `ypserv(8)` indításakor ezeket az információkat a `/var/yp/securenets` állományból próbálja meg betölteni.

Megjegyzés: Az elérési útvonala megadható a `-p` opció használatával. Ez az állomány olyan bejegyzéseket tartalmaz, amelyekben egy hálózati cím és tőle láthatatlan karakterekkel elválasztva egy hálózati maszk szerepel. A “`#`” karakterrel kezdődő sorokat megjegyzésnek nyilvánítjuk. Egy minta `securenets` állomány valahogy így nézne ki:

```
# Engedélyezzük önmagunkról a csatlakozást -- kell!
127.0.0.1      255.255.255.255
# Engedélyezzük a 192.168.128.0 hálózatról érkező csatlakozásokat:
192.168.128.0 255.255.255.0
# Engedélyezzük a laborban található 10.0.0.0 és 10.0.15.255 közti
# címekkel rendelkező gépek csatlakozását:
10.0.0.0      255.255.240.0
```

Ha az `ypserv(8)` olyan címről kap kérést, amely illeszkedik az előírt címek valamelyikére, akkor a szokásos módon feldolgozza azt. Ellenkező esetben a kérést figyelmen kívül hagyja és egy figyelmeztetést vesz fel hozzá a naplóba. Ha a `/var/yp/securenets` állomány nem létezik, akkor az `ypserv` tetszőleges gépről engedélyezi a csatlakozást.

Az `ypserv` lehetőséget ad a Wietse Venema által fejlesztett **TCP Wrapper** csomag használatára is. Ezzel a rendszergazda a `/var/yp/securenets` állomány helyett a **TCP Wrapper** konfigurációs állományai alapján képes szabályozni az elérhetőséget.

Megjegyzés: Miközben mind a két módszer nyújt valamilyen fajta védelmet, de a privilegizált portok teszteléséhez hasonlóan az “IP álcázásával” (IP spoofing) sebezhetőek. Ezért az összes NIS-hez tartozó forgalmat tűzfallal kell blokkolni.

Az `/var/yp/securenets` állományt használó szerverek nem képesek az elavult TCP/IP implementációkat használó érvényes klienseket rendesen kiszolgálni. Egyes ilyen implementációk a címben a géphez tartozó

biteket nullára állítják az üzenetszóráshoz, és/vagy ezért az üzenetszóráshoz használt cím kiszámításakor nem tudja észleli a hálózati maszkot. A legtöbb ilyen probléma megoldható a kliens konfigurációjának megváltoztatásával, míg más problémák megoldása a kérdéses kliensek nyugdíjazását kívánják meg, vagy a `/var/yp/securenets` használatának elhagyását.

Egy régebbi TCP/IP implementációval üzemelő szerveren pedig a `/var/yp/securenets` állomány használata kifejezetten rossz ötlet, és a hálózatunk nagy részében képes használhatatlanná tenni a NIS funkcióit.

A **TCP Wrapper** csomag alkalmazása a NIS szerverünk válaszadáshoz szükséges idejét is segít csökkenteni. Az ilyenkor jelentkező plusz késlekedés melleleg elég nagy lehet ahhoz, hogy a klienseknél időtúllépés következzen be, különösen a terheltebb hálózatokon vagy a lassú NIS szerverek esetében. Ha egy vagy több kliensünk is ilyen tüneteket mutat, akkor érdemes a kérdéses kliens rendszereket alárendelt NIS szerverekké alakítani és önmagukhoz rendelni.

29.4.6. Egyes felhasználók bejelentkezésének megakadályozása

A laborunkban van egy *basie* nevű gép, amely a tanszék egyetlen munkaállomása. Ezt a gépet nem akarjuk kivenni a NIS tartományból, de a központi NIS szerver `passwd` állománya mégis egyaránt tartalmazza a hallgatók és az oktatók eléréseit. Mit lehet ilyenkor tenni?

Adott felhasználók esetében le tudjuk tiltani a bejelentkezést a gépen még olyankor is, ha léteznek a NIS adatbázisában. Ehhez mindössze a kliensen az `/etc/master.passwd` állomány végére be kell tennünk egy *-felhasználónév* sort, ahol a *felhasználónév* annak a felhasználónak a neve, akit nem akarunk beengedni a gépre. Ezt leginkább a `vipw` használatán keresztül érdemes megtennünk, mivel a `vipw` az `/etc/master.passwd` állomány alapján végez némi ellenőrzést, valamint a szerkesztés befejeztével magától újragenerálja a jelszavakat tároló adatbázist. Például, ha a *bill* nevű felhasználót ki akarjuk tiltani a *basie* nevű gépről, akkor:

```
basie# vipw
[vegyük fel a -bill sort a végére, majd lépünk ki]
vipw: rebuilding the database...
vipw: done

basie# cat /etc/master.passwd

root:[jelszó]:0:0:0:0:The super-user:/root:/bin/csh
toor:[jelszó]:0:0:0:0:The other super-user:/root:/bin/sh
daemon:*:1:1:0:0:Owner of many system processes:/root:/sbin/nologin
operator:*:2:5:0:0:System &:/sbin/nologin
bin:*:3:7:0:0:Binaries Commands and Source,,:/sbin/nologin
tty:*:4:65533:0:0:Tty Sandbox:/sbin/nologin
kmem:*:5:65533:0:0:KMem Sandbox:/sbin/nologin
games:*:7:13:0:0:Games pseudo-user:/usr/games:/sbin/nologin
news:*:8:8:0:0:News Subsystem:/sbin/nologin
man:*:9:9:0:0:Mister Man Pages:/usr/share/man:/sbin/nologin
bind:*:53:53:0:0:Bind Sandbox:/sbin/nologin
uucp:*:66:66:0:0:UUCP pseudo-user:/var/spool/uucppublic:/usr/libexec/uucp/uucico
xten:*:67:67:0:0:X-10 daemon:/usr/local/xten:/sbin/nologin
pop:*:68:6:0:0:Post Office Owner:/nonexistent:/sbin/nologin
nobody:*:65534:65534:0:0:Unprivileged user:/nonexistent:/sbin/nologin
+:+++++
-bill
```

basie#

29.4.7. A hálózati csoportok alkalmazása

Készítette: Udo Erdelhoff.

Az előző szakaszban ismertetett módszer viszonylag jól működik olyan esetekben, amikor nagyon kevés felhasználóra és/vagy számítógépre kell alkalmaznunk speciális megszorításokat. A nagyobb hálózatokban szinte *biztos*, hogy elfelejtünk kizárni egyes felhasználókat az érzékeny gépekről, vagy az összes gépen egyenként kell ehhez a megfelelő beállításokat elvégezni, és ezzel lényegében elvesztjük a NIS legfontosabb előnyét, vagyis a *központosított* karbantarthatóságot.

A NIS fejlesztői erre a problémára a *hálózati csoportokat* létrehozásával válaszoltak. A céljuk és működésük szempontjából leginkább a UNIX-os állományrendszerekben található csoportokhoz mérhetőek. A legnagyobb eltérés a numerikus azonosítók hiányában mutatkozik meg, valamint a hálózati csoportokat a felhasználókon kívül további hálózati csoportok megadásával is ki lehet alakítani.

A hálózati csoportok a nagyobb, bonyolultabb, többszáz felhasználós hálózatok számára jöttek létre. Egy részről ez nagyon jó dolog, különösen akkor, ha egy ilyen helyzettel kell szembenéznünk. Másrészről ez a mértékű bonyolultság szinte teljesen lehetetlenné teszi a hálózati csoportok egyszerű bemutatását. A szakasz további részében használt példa is ezt a problémát igyekszik illusztrálni.

Tételezzük fel, hogy laborunkban a NIS sikeres bevezetése felkeltette a főnökeink figyelmét. Így a következő feladatunk az lett, hogy terjesszük ki a NIS tartományt az egyetemen található néhány másik gépre is. Az alábbi két táblázatban az új felhasználók és az új számítógép neveit találjuk, valamint a rövid leírásukat.

Felhasználók nevei	Leírás
alpha, beta	az IT tanszék hétköznapi dolgozói
charlie, delta	az IT tanszék újdonsült dolgozói
echo, foxtrott, golf, ...	átlagos dolgozók
able, baker, ...	ösztöndíjasok

Gépek nevei	Leírás
haboru, halal, ehseg, szennyezes	A legfontosabb szervereink. Csak az IT tanszék dolgozói férhetnek hozzájuk.
buszkeseg, kapzsisag, irigyseg, harag, bujasag, lustasag	Kevésbé fontos szerverek. Az IT tanszék összes tagja el tudja érni ezeket a gépeket.
egy, ketto, három, negy, ...	Átlagos munkaállomások. Egyedül csak a <i>valódi</i> dolgozók jelentkezhetnek be ezekre a gépekre.
szemetes	Egy nagyon régi gép, semmi értékes adat nincs rajta. Akár még az ösztöndíjasok is nyúzhatják.

Ha ezeket az igényeket úgy próbáljuk meg teljesíteni, hogy a felhasználókat egyenként blokkoljuk, akkor minden rendszer `passwd` állományába külön fel kell vennünk a *-felhasználó* sorokat a letiltott felhasználókhoz. Ha csak egyetlen bejegyzést is kihagyunk, akkor könnyen bajunk származhat belőle. Ez a rendszer kezdeti beállítása során még talán nem okoz gondot, de az új felhasználókat *biztosan* el fogjuk felejteni felvenni a megfelelő csoportokba.

Elvégre Murphy is optimista volt.

A hálózati csoportok használata ilyen helyzetekben számos előnyt rejt. Nem kell az egyes felhasználókat külön felvenni, egy felhasználót felveszünk valamelyik csoportba vagy csoportokba, és a csoportok összes tagjának egyszerre tudjuk tiltani vagy engedélyezni a hozzáféréseket. Ha hozzáadunk egy új gépet a hálózatunkhoz, akkor mindössze a hálózati csoportok bejelentkezési korlátozásait kell beállítani. Ha új felhasználót veszünk fel, akkor a felhasználót kell vennünk egy vagy több hálózati csoportba. Ezek a változtatások függetlenek egymástól, és nincs szükség “minden felhasználó és minden gép összes kombinációjára”. Ha a NIS beállításainkat előzetesen körültekintően megterveztük, akkor egyetlen központi konfigurációs állományt kell módosítani a gépek elérésének engedélyezéséhez vagy tiltásához.

Az első lépés a hálózati csoportokat tartalmazó NIS táblázat inicializálása. A FreeBSD ypinit(8) programja alapértelmezés szerint nem hozza létre ezt a táblázatot, de ha készítünk egy ilyet, akkor a NIS implementációja képes kezelni. Egy ilyen üres táblázat elkészítéséhez ennyit kell begépelni:

```
ellington# vi /var/yp/netgroup
```

Ezután elkezdhetjük felvenni a tartalmát. A példánk szerint legalább négy hálózati csoportot kell csinálnunk: az IT dolgozóinak, az IT új dolgozóinak, a normál dolgozóknak és az ösztöndíjasoknak.

```
IT_DOLG      ( ,alpha,proba-tartomany)    ( ,beta,proba-tartomany)
IT_UJDOLG    ( ,charlie,proba-tartomany)  ( ,delta,proba-tartomany)
FELHASZNALO  ( ,echo,proba-tartomany)     ( ,foxtrott,proba-tartomany) \
              ( ,golf,proba-tartomany)
OSZTONDIJAS  ( ,able,proba-tartomany)     ( ,baker,proba-tartomany)
```

Az IT_DOLG, IT_UJDOLG stb. a hálózati csoportok nevei lesznek. Minden egyes zárójelezett csoport egy vagy több felhasználói hozzáférést tartalmaz. A csoportokban szereplő három mező a következő:

1. Azon gépek neve, amelyekre a következő elemek érvényesek. Ha itt nem adunk meg neveket, akkor a bejegyzés az összes gépre vonatkozik. Ha megadjuk egy gép nevét, akkor jutalmunk a teljes sötétség, a rettegetés és totális megtérbolyodás.
 2. A csoporthoz tartozó hozzáférés neve.
 3. A hozzáféréshez kapcsolódó NIS tartomány. A csoportba más NIS tartományokból is át tudunk hozni hozzáféréseket, ha netalán éppen olyan szerencsétlenek lennénk, hogy több NIS tartományt is felügyelnünk kell.
- A mezők mindegyike tartalmazhat dőszókerakaraktereket. Erről részletesebben a netgroup(5) man oldalon olvashatunk.

Megjegyzés: A hálózati csoportoknak lehetőleg ne adjunk 8 karakternél hosszabb nevet, különösen abban az esetben, ha a NIS tartományban más operációs rendszereket is használunk. A nevekben eltérnek a kis- és nagybetűk. Ha a hálózati csoportokat nevét nagybetűkkel írjuk, akkor könnyen különbséget tudunk tenni a felhasználók, gépek és hálózati csoportok nevei között.

Egyes (nem FreeBSD alapú) NIS kliensek nem képesek kezelni a nagyon sok bejegyzést tartalmazó hálózati csoportokat. Például a SunOS néhány korábbi verziója fennakad rajta, ha egy hálózati csoport 15 *bejegyzésnél* többet tartalmaz. Az ilyen korlátozások alól úgy tudunk kibújni, ha 15 felhasználónként újabb hálózati csoportokat hozunk létre, amelyekkel az eredeti hálózati csoportot építjük fel:

```
NAGYCSP1    ( ,joe1,tartomany) ( ,joe2,tartomany) ( ,joe3,tartomany) [...]
NAGYCSP2    ( ,joe16,tartomany) ( ,joe17,tartomany) [...]
NAGYCSP3    ( ,joe31,tartomany) ( ,joe32,tartomany)
NAGYCSOPORT NAGYCSP1 NAGYCSP2 NAGYCSP3
```

Ugyanez a folyamat javasolt olyan esetekben is, ahol 225 felhasználónál többre lenne szükség egyetlen hálózati csoporton belül.

Az így létrehozott új NIS táblázat szétküldése meglehetősen könnyű feladat:

```
ellington# cd /var/yp
ellington# make
```

Ez a parancs létrehoz három NIS táblázatot: `netgroup`, `netgroup.byhost` és `netgroup.byuser`. Az `ypcat(1)` paranccsal ellenőrizni is tudjuk az új NIS táblázatainkat:

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

Az első parancs kimenete a `/var/yp/netgroup` állomány tartalmára emlékeztethet minket. A második parancsnak nincs semmilyen kimenete, hacsak nem adtunk meg valamilyen gépfüggő hálózati csoportot. A harmadik parancs a hálózati csoportokat listázza ki a felhasználókhöz.

A kliensek beállítása tehát nagyon egyszerű. A `haboru` nevű szerver beállításához indítsuk el a `vipw(8)` programot, és cseréljük a

```
+:::~::~:
```

sort erre:

```
+@IT_DOLG:::~::~:
```

Innentől kezdve kizárólag csak az `IT_DOLG` csoportban található felhasználók fognak bekerülni a `haboru` jelszó adatbázisába, és csak ezek a felhasználók tudnak ide bejelentkezni.

Sajnos ez a korlátozás a parancsértelmező `~` funkciójára és összes olyan rutinra is vonatkozik, amelyet a felhasználói nevek és azok numerikus azonosító között képez le. Más szóval a `cd ~felhasználó` parancs nem fog működni, és az `ls -l` parancs kimenetében a felhasználói nevek helyett csak numerikus azonosítók jelennek meg, továbbá a `find . -user joe -print` No such user (Nincs ilyen felhasználó) hibát fog visszaadni. Ez úgy tudjuk megjavítani, ha úgy importáljuk a szerverre az összes felhasználó bejegyzését, hogy közben tiltjuk a hozzáférésüket.

Ehhez vegyünk fel egy újabb sort az `/etc/master.passwd` állományba. A sor valahogy így fog kinézni:

```
+:::~::~:/sbin/nologin, amely annyit tesz, hogy "importáljuk az összes bejegyzést, de a hozzájuk tartozó parancsértelmező a /sbin/nologin legyen". A passwd állományban tetszőleges mező tartalmát le tudjuk úgy cserélni, ha megadunk neki egy alapértelmezett értéket az /etc/master.passwd állományban.
```

Figyelem Vigyázzunk, hogy a `+:::~::~:/sbin/nologin` sort az `+@IT_DOLG:::~::~:` sor után írjuk. Ha nem így teszünk, akkor a NIS-ből importált összes felhasználói hozzáférés a `/sbin/nologin` parancsértelmezőt kapja.

Miután elvégeztük ezt a változtatást, minden újabb dolgozó felvétele után csupán egyetlen táblázatot kell megváltoztatnunk. Ugyanezt a taktikát követhetjük a kevésbé fontosabb szerverek esetében is, hogy ha a helyi `/etc/master.passwd` állományukban a korábbi `+:::~::~:` bejegyzést valami ilyesmivel helyettesítjük:

```
+@IT_DOLG:::
+@IT_UJDOLG:::
+:::/sbin/nologin
```

Az egyszerű munkaállomások esetében pedig ezekre a sorokra lesz szükségünk:

```
+@IT_DOLG:::
+@FELHASZNALOK:::
+:::/sbin/nologin
```

Minden remekül üzemel egészen addig, amíg néhány hét múlva ismét változik a házirend: az IT tanszékre ösztöndíjasok érkeznek. Az IT ösztöndíjasai a munkaállomásokat és a kevésbé fontosabb szervereket tudják használni. Az új IT dolgozók már a központi szerverekre is bejelentkezhetnek. Így tehát létrehozunk egy új hálózati csoportot `IT_OSZTONDIJAS` néven, majd felvesszük ide az új IT ösztöndíjasokat, és nekilátunk végigzongorázni az összes gép összes konfigurációs állományát... Ahogy azonban egy régi mondás is tartja: “A központosított tervezésben ejtett hibák teljes káoszhoz vezetnek”.

A NIS az ilyen helyzeteket úgy igyekszik elkerülni, hogy megengedi újabb hálózati csoportok létrehozását más hálózati csoportokból. Egyik ilyen lehetőség a szerep alapú hálózati csoportok kialakítása. Például, ha a fontosabb szerverek bejelentkezési korlátozásai számára hozzunk létre egy `NAGYSRV` nevű csoportot, valamint egy másik hálózati csoportot `KISSRV` néven a kevésbé fontosabb szerverekhez, végül `MUNKA` néven egy harmadik hálózati csoportot a munkaállomásokhoz. Mindegyik ilyen hálózati csoport tartalmazza azokat a csoportokat, amelyek engedélyezik a gépek elérését. A hálózati csoportok leírását tartalmazó NIS táblázat most valahogy így fog kinézni:

```
NAGYSRV  IT_DOLG  IT_UJDOLG
KISSRV   IT_DOLG  IT_UJDOLG  IT_OSZTONDIJAS
MUNKA    IT_DOLG  IT_OSZTONDIJAS  FELHASZNALOK
```

A bejelentkezési megszorítások ilyen típusú megadása viszonylag jól működik, hogy ha azonos korlátozások alá eső gépek csoportjait akarjuk felírni. Bánatunk ez a kivétel, és nem a szabály. Az esetek nagy többségében ugyanis a bejelentkezésre vonatkozó korlátozásokat gépenként kell egyesével megadni.

A hálózati csoportok gépfüggő megadása tehát az iménti házirendhez társuló igények kielégítésének egyik módja. Ebben a forgatókönyvben az `/etc/master.passwd` állomány minden számítógépen két “+”-os sorral kezdődik. Közülük az első a gépen engedélyezett hozzáféréseket tartalmazó hálózati csoportra vonatkozik, a második pedig az összes többi hozzáféréshez az `/sbin/nologin` parancsértelmezőt kapcsolja hozzá. Itt jó ötlet, ha a gép nevének “VÉGIG-NAGYBETŰS” változatát adjuk meg a hozzá tartozó hálózati csoport nevének:

```
+@GÉPNÉV:::
+:::/sbin/nologin
```

Miután elvégeztük ezt a feladatot minden egyes gépen, az `/etc/master.passwd` állomány helyi változatait soha többé nem kell módosítanunk. Az összes többi változtatást a NIS táblázaton keresztül tudjuk keresztül vinni. Íme a felvázolt forgatókönyvhöz tartozó hálózati csoportok kiépítésének egyik lehetséges változata, egy-két finomsággal kiegészítve:

```
# Először a felhasználók csoportjait adjuk meg:
IT_DOLG      ( ,alpha,proba-tartomany)    ( ,beta,proba-tartomany)
IT_UJDOLG    ( ,charlie,proba-tartomany)        ( ,delta,proba-tartomany)
TANSZ1       ( ,echo,proba-tartomany)    ( ,foxtrott,proba-tartomany)
TANSZ2       ( ,golf,proba-taromany)      ( ,hotel,proba-tartomany)
TANSZ3       ( ,india,proba-taromany)     ( ,juliet,proba-tartomany)
IT_OSZTONDIJAS ( ,kilo,proba-tartomany)         ( ,lima,proba-tartomany)
```

```

D_OSZTONDIJAS    (,able,proba-tartomany)    (,baker,proba-tartomany)
#
# Most pedig hozzunk létre csoportokat szerepek szerint:
FELHASZNALOK      TANSZ1    TANSZ2            TANSZ3
NAGYSRV           IT_DOLG   IT_UJDOLG
KISSRV            IT_DOLG   IT_UJDOLG      IT_OSZTONDIJAS
MUNKA             IT_DOLG   IT_OSZTONDIJAS FELHASZNALOK
#
# Következzenek a speciális feladatokhoz tartozó csoportok:
# Az echo és a golf tudja elérni a vírusvédelemért felelős gépet:
VEDELEM           IT_DOLG   (,echo,proba-tartomany) (,golf,proba-tartomany)
#
# Gép alapú hálózati csoportok
# A fő szervereink:
HABORU            NAGYSRV
EHSEG             NAGYSRV
# Az india nevű felhasználó hozzá szeretné ehhez férni:
SZENNYEZES        NAGYSRV   (,india,proba-tartomany)
#
# Ez valóban fontos és komolyan szabályoznunk kell:
HALAL             IT_DOLG
#
# Az előbb említett vírusvédelmi gép:
EGY               VEDELEM
#
# Egyetlen felhasználóra korlátozzuk le ezt a gépet:
KETTO             (,hotel,proba-tartomany)
# [...és itt folytatódik a többi csoporttal]

```

Ha a felhasználói hozzáféréseinket valamilyen adatbázisban tároljuk, akkor a táblázat első részét akár az adatbázis lekérdezéseink keresztül is elő tudjuk állítani. Ezzel a módszerrel az új felhasználók automatikusan hozzáférnek a gépekhez.

Legyünk viszont óvatosak: nem mindig javasolt gépeken alapuló hálózati csoportokat készíteni. Ha a hallgatói laborokba egyszerre több tucat vagy akár több száz azonos konfigurációjú gépet telepítünk, akkor a gép alapú csoportok helyett inkább szerep alapú csoportokat építsünk fel, mivel így a NIS táblázatok méretét egy elfogadható méreten tudjuk tartani.

29.4.8. Amit feltétlenül észben kell tartanunk

Még mindig akad néhány olyan dolog, amit másképpen kell csinálnunk azután, hogy most már NIS környezetben vagyunk.

- Amikor egy új felhasználót akarunk felvenni a laborba, akkor *csak* a központi NIS szerverre kell felvennünk, és *újra kell generáltatnunk a NIS táblázatokat*. Ha ezt elfelejtjük megtenni, akkor az új felhasználó a központi NIS szerveren kívül sehova sem lesz képes bejelentkezni. Például, ha fel akarjuk venni a `jsmith` nevű felhasználót a laborba, akkor ezt kell tennünk:

```

# pw useradd jsmith
# cd /var/yp
# make proba-tartomany

```

Vagy a `pw useradd jsmith` parancs helyett az `adduser jsmith` parancsot is használhatjuk.

- *A rendszergazdai szintű hozzáféréseket ne tároljuk a NIS táblázatokban.* Olyan gépekre egyáltalán ne is küldjünk olyan karbantartáshoz használt hozzáféréseket, amelynek a felhasználói hivatalosan nem is férhetnének hozzájuk.
- *A központi NIS szervert és az alárendelt szervereket óvjuk minél jobban, és igyekezzünk minimalizálni a kieséseiket.* Ha valaki feltöri vagy egyszerűen csak kikapcsolja ezeket a gépeket, akkor ezzel lényegében mindenkit megakadályoz abban, hogy be tudjon jelentkezni a laborban.

Ezek a központosított vezérlésű rendszerek legfőbb gyengeségei. Ha nem védjük kellően a NIS szervereinket, akkor azzal nagyon ellenséget szerezhetünk magunknak!

29.4.9. Kompatibilitás a NIS első változatával

A FreeBSD-ben megtalálható **ypserv** szolgáltatás valamennyire képes ellátni a NIS első változatát használó klienseket is. A FreeBSD NIS implementációja csak a NIS v2 protokollt használja, azonban mivel más implementációk kompatibilisek kívánnak maradni a régebbi rendszerekkel, ismerik a v1 protokollt is. Az ilyen rendszerekhez tartozó **ypbind** démonok még olyankor is megpróbálnak v1-es NIS szerverekhez kötést létrehozni, amikor valójában nincs is rá szükségük (és gyakran még akkor is ilyet keresnek, amikor az üzenetükre már válaszolt egy v2-es szerver). Hozzátennénk, hogy bár az **ypserver** ezen változata a normál klienshívásokat képes feldolgozni, a táblázatok már nem tudja átküldeni a v1-es klienseknek. Ebből következik, hogy a központi vagy alárendelt szerverek nem tudnak együttműködni olyan NIS szerverekkel, amelyek csak a v1-es protokollt beszélik. Szerencsére ilyen szervereket manapság már alig használnak.

29.4.10. NIS szerverek, melyek egyben NIS kliensek

Óvatosan kell bánnunk az **ypserv** elindításával olyan többszerveres tartományokban, ahol a szerverek maguk is NIS kliensek. Alapvetően nincs abban semmi kivetnivaló, ha a szervereket saját magukhoz kötjük ahelyett, hogy engednénk nekik a kötési kérések küldését és így egymáshoz kötnénk ezeket. Különös hibák tudnak származni olyan helyzetekben, amikor az egyik szerver leáll, miközben a többiek pedig függenek tőle. Végül is ilyenkor minden kliens szépen kivárja a szükséges időt, aztán megpróbál más szerverekhez kötődni, de az itt fellépő késlekedés jelentős mennyiségű lehet, és ez a hibajelenség ismét fennállhat, mivel előfordulhat, hogy a szerverek megint egymáshoz kapcsolódnak.

A klienst úgy tudjuk egy adott szerverhez kötni, ha az **ypbind** parancsot a `-s` beállítással indítjuk. Ha mindezt nem akarjuk manuálisan megtenni a NIS szerver minden egyes újraindításakor, akkor vegyük fel a következő sorokat az `/etc/rc.conf` állományba:

```
nis_client_enable="YES" # elindítjuk a klienst is
nis_client_flags="-s NIS tartomány,server"
```

Részletesebb lásd az **ypbind(8)** man oldalát.

29.4.11. A jelszavak formátuma

A NIS rendszerek kiépítése során az emberek leggyakrabban a jelszavak formátumával kapcsolatban tapasztalnak nehézségeket. Ha a szerverünk DES titkosítású jelszavakat használ, akkor csak olyan klienseket fog tudni támogatni, amelyek szintén így kódolják ezeket. Például, ha a hálózaton vannak Solaris rendszerű NIS klienseink, akkor szinte biztos, hogy DES titkosítást kell használnunk.

A szerverek és a kliensek által használt formátumokat az `/etc/login.conf` állományba tekintve deríthetjük ki. Ha a gépek többségén a DES titkosítást látjuk, akkor a `default` osztálynak egy ilyen bejegyzést kell tartalmaznia:

```
default:\
    :passwd_format=des:\
    :copyright=/etc/COPYRIGHT:\
    [a többi most nem mutatjuk]
```

A `passwd_format` tulajdonság további lehetséges értékei lehetnek a `blf` és az `md5` (melyek rendre a Blowfish és MD5 titkosítású jelszavakat adják meg).

Ha változtattunk valamit az `/etc/login.conf` állományban, akkor a bejelentkezési tulajdonságok adatbázisát is újra kell generálni, melyet `root` felhasználóként a következő módon tehetünk meg:

```
# cap_mkdb /etc/login.conf
```

Megjegyzés: Az `/etc/master.passwd` állományban jelenlevő jelszavak formátuma azonban nem frissítődik egészen addig, amíg a felhasználók a bejelentkezési adatbázis újragenerálása *után* meg nem változtatják a jelszavaikat.

Úgy tudjuk még biztosítani, hogy a jelszavak megfelelő formátumban kódolódjanak, ha az `/etc/auth.conf` állományban megkeressük a `crypt_default` sort, amelyben a választható jelszóformátumok felhasználásai sorrendjét találhatjuk meg. Itt tehát mindössze annyit kell tennünk, hogy a kiszemelt formátumot a lista elejére tesszük. Például, ha a DES titkosítású jelszavakat akarunk használni, akkor ez a bejegyzés így fog kinézni:

```
crypt_default    =      des blf md5
```

Ha a fenti lépéseket követjük az összes FreeBSD alapú NIS szervernél és kliensnél, akkor biztosra mehetünk abban, hogy a hálózatunkon belül ugyanazt a jelszóformátumot fogják használni. Ha gondunk akadna a NIS kliensek hitelesítésével, akkor itt érdemes kezdeni a hiba felderítését. Ne felejtjük: ha egy NIS szervert egy heterogén hálózatba akarunk telepíteni, akkor valószínűleg az összes rendszeren a DES titkosítást kell választani, mivel általában ez a közös nevező ebben a tekintetben.

29.5. A hálózat automatikus beállítása (DHCP)

Írta: Greg Sutter.

29.5.1. Mi az a DHCP?

A Dinamikus állomáskonfigurációs protokoll, avagy Dynamic Host Configuration Protocol (DHCP) annak eszközeit írja le, hogy egy rendszer miként tud csatlakozni egy hálózathoz és miként tudja azon belül megszerezni a kommunikációhoz szükséges információkat. A FreeBSD 6.0 előtti változatai az ISC (Internet Systems Consortium, vagyis az internetes rendszerkonzorcium) által kidolgozott DHCP kliens (`dhclient(8)`) implementációját tartalmazzák. A későbbi verziókban pedig az OpenBSD 3.7 verziójából átvett `dhclient` paranccsal dolgozhatunk. Ebben a szakaszban a `dhclient` parancsra vonatkozó összes információ egyaránt érvényes az ISC és az OpenBSD által fejlesztett DHCP kliensekre. A DHCP szerver az ISC-től származik.

29.5.2. Mivel foglalkozik ez a szakasz

Ebben a szakaszban az ISC és az OpenBSD DHCP klienseinek kliens- és szerver oldali komponenseit mutatjuk be. A kliens oldali program neve a `dhclient`, amely a FreeBSD részeként érkezik, és a szerver oldali elem pedig a `net/isc-dhcp31-server` porton keresztül érhető el. A lentebb említett hivatkozások mellett a témában még a `dhclient(8)`, `dhcp-options(5)` és a `dhclient.conf(5)` man adhatnak bővebb felvilágosítást a témában.

29.5.3. Ahogyan működik

Amikor a `dhclient`, vagyis a DHCP kliens elindul egy kliensgépen, akkor a hálózaton üzenetszórással próbálja meg elkérni a konfigurációjához szükséges adatokat. Alapértelmezés szerint ezek a kérések a 68-as UDP porton keresztül mennek. A szerver ezekre a 67-es UDP porton válaszol, ahol visszaad a kliensnek egy IP-címet és a hálózat használatához szükséges további információkat, mint például a hálózati maszkot, az alapértelmezett átjáró és a névfeloldásért felelős szerverek címét. Az összes ilyen jellegű adat egy DHCP “bérlet” (lease) formájában érkezik meg, amely csak egy adott ideig érvényes (ezt a DHCP szerver karbantartója állítja be). Így a hálózaton a kliens nélküli IP-címeket egy idő után automatikusan visszanyerjük.

A DHCP kliensek rengeteg információt képesek elkérni a szervertől. Ezek teljes listáját a `dhcp-options(5)` man oldalán olvashatjuk el.

29.5.4. Használat a FreeBSD-n belül

A FreeBSD teljes egészében tartalmazza az ISC vagy az OpenBSD DHCP kliensét, a `dhclient` programot (attól függően, hogy a FreeBSD melyik változatát használjuk). A DHCP kliensek támogatása a telepítőben és az alaprendszerben is megtalálható, és ezzel mentesülünk minden konkrét hálózati beállítás alól a DHCP szervereket alkalmazó hálózatokon. A `dhclient` a FreeBSD 3.2 változata óta megtalálható a rendszerben.

DHCP használatát a `sysinstall` is lehetővé teszi. Amikor egy hálózati felületet a `sysinstall` programon belül állítunk be, akkor a második kérdés mindig ez szokott lenni: “Do you want to try DHCP configuration of the interface?” (“Megpróbáljuk DHCP használatával beállítani a felületet?”) Ha erre igennel válaszolunk, akkor azzal lényegében a `dhclient` parancsot indítjuk el, és ha mindez sikerrel zárul, akkor szinte magától kitöltődik az összes hálózati beállításunk.

A DHCP használatához két dolgot kell beállítanunk a rendszerünkön:

- Gondoskodjunk róla, hogy a `bpf` eszköz része a rendszermagunknak. Ha még nem lenne benne, akkor a rendszermag beállításait tartalmazó állományba vegyük fel a `device bpf` sort és fordítsuk újra a rendszermagot. A rendszermagok fordításáról a 8 fejezetben tudhatunk meg többet.

A `bpf` eszköz alpból megtalálható a `GENERIC` rendszermagokban, így ha ezt használjuk, akkor nem kell saját verziót készítenünk a DHCP használatához.

Megjegyzés: Azok számára viszont, akik biztonsági szempontból aggódnak a rendszerük miatt, meg kell említenünk, hogy a `bpf` egyben az az eszköz, amely a csomagok lehallgatását is lehetővé teszi (habár az ilyeneket `root` felhasználóként lehet csak elindítani). A `bpf` *kell* a DHCP használatához, azonban ha nagyon fontos nekünk a rendszerünk biztonsága, akkor a `bpf` eszközt érdemes kivennünk a rendszermagból, ha még pillanatnyilag nem használunk ilyet.

- Az `/etc/rc.conf` állományunkat az alábbiak szerint kell módosítani:

```
ifconfig_fxp0="DHCP"
```

Megjegyzés: Az `fxp0` eszközt ne felejtsük el kicserélni arra a felületre, amelyet automatikusan akarunk beállítani. Ennek mikéntje a 11.8 Szakaszban olvasható.

Ha a `dhclient` a rendszerünkben máshol található, vagy egyszerűen csak további beállításokat akarunk átadni a `dhclient` parancsnak, akkor adjuk meg a következőt is (változtassuk meg igényeink szerint):

```
dhclient_program="/sbin/dhclient"
dhclient_flags=""
```

A DHCP szerver, a **dhcpcd** a `net/isc-dhcp31-server` port részeként érhető el. Az a port tartalmazza az ISC DHCP szerverét és a hozzá tartozó dokumentációt.

29.5.5. Állományok

- `/etc/dhclient.conf`

A `dhclient` működéséhez szükség lesz egy konfigurációs állományra, aminek a neve `/etc/dhclient.conf`. Ez az állomány általában csak megjegyzéseket tartalmaz, mivel az alapértelmezett értékek többnyire megfelelőek. Ezt a konfigurációs állományt a `dhclient.conf(5)` man oldal írja le.

- `/sbin/dhclient`

A `dhclient` statikusan linkelt és az `/sbin` könyvtárban található. A `dhclient(8)` man oldal tud róla részletesebb felvilágosítást adni.

- `/sbin/dhclient-script`

A `dhclient-script` a FreeBSD-ben levő DHCP kliens konfigurációs szkriptje. Működését a `dhclient-script(8)` man oldal írja le, de a felhasználók részéről semmilyen módosítást nem igényel.

- `/var/db/dhclient.leases`

A DHCP kliens az érvényes bérleteket tartja nyilván ezekben az állományban és naplóként használja. A `dhclient.leases(5)` man oldal ezt valamivel bővebben kifejti.

29.5.6. További olvasnivalók

A DHCP protokoll működését az RFC 2131 (<http://www.freesoft.org/CIE/RFC/2131/>) mutatja be. A témához kapcsolódóan itt (<http://www.dhcp.org/>) tudunk még leírásokat találni.

29.5.7. A DHCP szerverek telepítése és beállítása

29.5.7.1. Miről szól ez a szakasz

Ebben a szakaszban arról olvashatunk, hogy miként kell egy FreeBSD típusú rendszert DHCP szervernek beállítani, ha az ISC (internetes rendszerkonzorcium) DHCP szerverét használjuk.

Ez a szerver nem része a FreeBSD-nek, ezért a szolgáltatás elindításához először fel kell raknunk a `net/isc-dhcp31-server` portot. A Portgyűjtemény használatára vonatkozóan a 4 fejezet lehet segítségünkre.

29.5.7.2. A DHCP szerver telepítése

Ha a FreeBSD rendszerünket DHCP szerverként akarjuk beállítani, akkor ehhez elsőként a `bpf(4)` eszköz jelenlétét kell biztosítani a rendszermagban. Ehhez vegyük fel a `device bpf` sort a rendszermagunk beállításait tartalmazó állományba, majd fordítsuk újra a rendszermagot. A rendszermag lefordításáról a 8 fejezetben olvashatunk.

A `bpf` eszköz a FreeBSD-hez alapból adott `GENERIC` rendszermag része, ezért a DHCP használatához nem kell feltétlenül újat fordítanunk.

Megjegyzés: A biztonsági szempontok miatt aggódó felhasználók részére megjegyezzük, hogy a `bpf` eszköz egyben a csomagok lehallgatását is lehetővé teszi (habár az ilyen témájú programok futtatásához megfelelő jogokra is szükség van). A `bpf` használata *kötelező* a DHCP működtetéséhez, de ha nagyon kényesek vagyunk a biztonságot illetően, akkor minden olyan esetben, amikor nem használjuk ki ezt a lehetőséget, távolítsuk el a rendszermagból.

A következő lépésben át kell szerkesztenünk a mintaként mellékelt `dhcpd.conf` állományt, amelyet a `net/isc-dhcp31-server` port rakott fel. Ez alapértelmezés szerint a `/usr/local/etc/dhcpd.conf.sample` néven található meg, és mielőtt bármit is változtatnánk rajta, másoljuk le `/usr/local/etc/dhcpd.conf` néven.

29.5.7.3. A DHCP szerver beállítása

A `dhcpd.conf` az alhálózatokat illetve a gépeket érintő deklarációkat tartalmazza, és talán a legkönnyebben a következő példa alapján mutatható be:

```
option domain-name "minta.com";❶
option domain-name-servers 192.168.4.100;❷
option subnet-mask 255.255.255.0;❸

default-lease-time 3600;❹
max-lease-time 86400;❺
ddns-update-style none;❻

subnet 192.168.4.0 netmask 255.255.255.0 {
    range 192.168.4.129 192.168.4.254;❼
    option routers 192.168.4.1;❽
}

host mailhost {
    hardware ethernet 02:03:04:05:06:07;❾
    fixed-address levelezes.minta.com;(10)
}
```

- ❶ Ez a beállítás adja meg a kliensek számára az alapértelmezett keresési tartományt (search domain). A `resolv.conf(5)` tud ezzel kapcsolatban részletesebb információkat adni.
- ❷ Ez a beállítás adja meg a kliensek által használt névfeloldó szerverek vesszővel elválasztott felsorolását.

- ③ A kliensekhez tartozó hálózati maszk.
- ④ A kliens egy adott időre kérhet bérleti jogot, egyébként a szerver dönt a bérlet lejáratí idejéről (másodpercekben).
- ⑤ Ez az a maximális idő, amennyire a szerver hajlandó bérbe adni IP-címet. A kliens ugyan hosszabb időre is kérheti és meg is kapja, de legfeljebb csak `max-lease-time` másodpercig lesz érvényes.
- ⑥ Ez a beállítás határozza meg, hogy a DHCP szervernek frissítse-e a névoldási információkat a bérletek elfogadásánál vagy visszamondásánál. Az ISC implementációjánál ez a beállítás *kötelező*.
- ⑦ Ezzel adjuk meg milyen tartományból tudunk IP-címeket kiosztani a kliensek számára. A kezdő címet is beleértve, innen fogunk kiutalni egyet a klienseknek.
- ⑧ A kliensek felé elküldött alapértelmezett átjáró címe.
- ⑨ A gép hardveres MAC-címe (így a DHCP szerver képes felismerni a kérés küldőjét).
- (10) Ennek megadásával a gépek mindig ugyanazt az IP-címet kapják. Itt már megadhatunk egy hálózati nevet, mivel a bérlethez tartozó információk visszaküldése előtt maga a DHCP szerver fogja feloldani a gép nevét.

Miután befejeztük a `dhcpcd.conf` módosítását, a DHCP szerver az `/etc/rc.conf` állományban tudjuk engedélyezni, vagyis tegyük bele a következőt:

```
dhcpcd_enable="YES"
dhcpcd_ifaces="dc0"
```

A `dc0` felület nevét helyettesítsük annak a felületnek (vagy whitespace karakterekkel elválasztott felületeknek) a nevével, amelyen keresztül a DHCP szerver várni fogja a kliensek kéréseit.

Ezután a következő parancs kiadásával indítsuk el a szerveret:

```
# /usr/local/etc/rc.d/isc-dhcpd start
```

Amikor a jövőben valamit változtatunk a konfigurációs állományon, akkor ezzel kapcsolatban fontos megemlíteni, hogy ha csak egy `SIGHUP` jelzést küldünk a **dhcpcd** démonnak, akkor az a többi démonról eltérően önmagában még *nem* eredményezi a konfigurációs adatok újraolvasását. Helyette a `SIGTERM` jelzéssel kell leállítani a programot, majd újraindítani a fenti paranccsal.

29.5.7.4. Állományok

- `/usr/local/sbin/dhcpcd`

A **dhcpcd** statikusan linkelt és a `/usr/local/sbin` könyvtárban található. A porttal együtt felkerülő `dhcpcd(8)` man oldal ad részletesebb útmutatást **dhcpcd** használatáról.

- `/usr/local/etc/dhcpcd.conf`

Mielőtt a **dhcpcd** megkezdhetné működését, egy konfigurációs állományra is szükségünk lesz, amely a `/usr/local/etc/dhcpcd.conf`. Ez az állomány tartalmazza az összes olyan információt, ami kell a kliensek megfelelő kiszolgálásához valamint a szerver működéséhez. Ez a konfigurációs állomány porthoz tartozó `dhcpcd.conf(5)` man oldalon kerül ismertetésre.

- `/var/db/dhcpcd.leases`

A DHCP szerver ebben az állományba tartja nyilván a kiadott bérleteket, egy napló formájában. A porthoz kapcsolódó `dhcpcd.leases(5)` man oldalon erről többet is megtudhatunk.

- `/usr/local/sbin/dhcrelay`

A **dhcrelay** állománynak olyan komolyabb környezetekben van szerepe, ahol a DHCP szerver a kliensektől érkező kéréseket egy másik hálózaton található DHCP szerverhez továbbítja. Ha szükség lenne erre a lehetőségre, akkor telepítsük fel a `net/isc-dhcp31-relay` portot. A porthoz tartozó `dhcrelay(8)` man oldal ennek részleteit taglalja.

29.6. Névfeloldás (DNS)

Készítette: Chern Lee, Tom Rhodes, és Daniel Gerzo.

29.6.1. Áttekintés

A FreeBSD alapértelmezés szerint a BIND (Berkeley Internet Name Domain) egyik verzióját tartalmazza, amely a névfeloldási (Domain Name System, DNS) protokoll egyik elterjedt implementációja. A DNS protokollon keresztül tudunk az IP-címekhez neveket rendelni és fordítva. Például a `www.FreeBSD.org` névre a FreeBSD Projekt webszerverének IP-címét kapjuk meg, miközben a `ftp.FreeBSD.org` pedig a hozzá tartozó FTP szerver IP-címét fogja visszaadni. Ehhez hasonlóan a fordítottja is megtörténhet, vagyis egy IP-címhez is kérhetjük a hálózati névfeloldását. A névfeloldási kérések kiszolgálásához nem feltétlenül szükséges névszerver futtatni a rendszerünkön.

A FreeBSD jelen pillanatban alpból a BIND9 névszervert tartalmazza. A benne szereplő változata több biztonsági javítást, új állományrendszeri kiosztást és automatizált `chroot(8)` beállítást is magában foglal.

Az interneten keresztüli névfeloldást legfelső szintű tartományoknak (Top Level Domain, TLD) nevezett hitelesített tövek némileg bonyolult rendszerén alapszik, valamint más egyéb olyan névszervereken, amelyek további egyéni információkat tárolnak és táraznak.

A BIND fejlesztését jelenleg az Internet Systems Consortium (<http://www.isc.org/>) felügyeli.

29.6.2. Alapfogalmak

A leírás megértéséhez be kell mutatnunk néhány névfeloldással kapcsolatos fogalmat.

Fogalom	Meghatározás
Közvetlen névfeloldás (forward DNS)	A hálózati nevek leképezése IP-címekre.
ős (origin)	Egy adott zóna állományban szereplő tartományra vonatkozik.
named , BIND	A FreeBSD-n belüli BIND névszerver különböző megnevezései.
Névfeloldó (resolver)	Az a program a rendszerben, amelyhez a hálózaton levő gépek a zónák adatainak elérésével kapcsolatban fordulnak.
Inverz névfeloldás (reverse DNS)	Az IP-címek leképezése hálózati nevekre.
Gyökérzóna (root zone)	Az interneten található zónák hierarchiájának töve. Minden zóna ebbe a gyökérzónába esik, ahhoz hasonlóan, ahogy egy állományrendszerben az állományok a gyökérkönyvtárba.

Fogalom	Meghatározás
Zóna (zone)	Egy különálló tartomány, altartomány vagy a névfeloldás azon része, amelyet egyazon fennhatóság alatt tartanak karban.

Példák zónákra:

- A gyökérzónára a leírásokban általában `.` néven szoktak hivatkozni.
- A `org.` egy legfelső szintű tartomány (TLD) a gyökérzónán belül.
- A `minta.org.` a `org.` TLD tartomány alatti zóna.
- A `1.168.192.in-addr.arpa` egy olyan zóna, amelyek a `192.168.1.*` IP-címtartományban szereplő összes címet jelöli.

Mint láthatjuk, a hálózati nevek balról kiegészülve pontosodnak. Tehát például a `minta.org.` sokkal pontosabb meghatározás, mint a `org.`, ahogy az `org.` magánál a gyökérzónánál jelent többet. A hálózati nevek felosztása leginkább egy állományrendszerhez hasonlítható, például a `/dev` könyvtár a gyökéren belül található, és így tovább.

29.6.3. Miért érdemes névszervert futtatni

A névszerverek általában két alakban jelennek meg. Egyikük a hitelesített névszerver, a másikuk a gyorsítótárazó névszerver.

Egy hitelesített névszerverre akkor van szükségünk, ha:

- a világ többi része felé akarunk hiteles névfeloldási információkat szolgáltatni;
- regisztráltunk egy tartományt (például `minta.org`) és az alatta levő hálózati nevekhez is szeretnénk IP-címeket rendeltetni;
- a IP-címtartományunkban szükség van inverz névfeloldási bejegyzésekre (amely IP-címből ad meg hálózati nevet) is;
- a kérések teljesítéséhez egy tartalék avagy második, alárendelt (slave) névszerver kell.

A gyorsítótárazó névszerverre akkor van szükségünk, ha:

- egy helyi névfeloldó szerver felhasználásával fel akarjuk gyorsítani az egyébként a külső névszerver felé irányuló kérések kiszolgálását.

Amikor valaki lekérdezi a `www.FreeBSD.org` címét, akkor a névfeloldó először általában a kapcsolatot rendelkezésre bocsátó internet-szolgáltató névszerverét kérdezi meg és onnan kapja meg a választ. Egy helyi, gyorsítótárazó névszerver használata esetén azonban egy ilyen kérést csak egyszer kell kiadni a külső névszervernek. Ezután már minden további ilyen kérés el sem hagyja a belső hálózatunkat, mivel a válasz szerepel a gyorsítótárban.

29.6.4. Ahogyan működik

FreeBSD alatt a BIND démon nyilvánvaló okokból **named** néven érhető el.

Állomány	Leírás
<code>named(8)</code>	A BIND démon.

Állomány

```
rndc(8)
/etc/namedb
/etc/namedb/named.conf
```

Leírás

A névszerver vezérlő segédprogram.
A BIND által kezelt zónák adatait tároló könyvtár.
A démon konfigurációs állománya.

Attól függően, hogy miként állítjuk be az adott zónát a szerveren, a hozzá tartozó állományok a `/etc/namedb` könyvtárban belül a `master`, `slave` vagy `dynamic` alkönyvtárban foglalnak helyet. Az itt tárolt állományokban levő névfeloldási információk alapján válaszol a névszerver a felé intézett kérésekre.

29.6.5. A BIND elindítása

Mivel a BIND alaphoz elérhető a rendszerben, viszonylag könnyen be tudjuk állítani.

A **named** alapértelmezett beállítása szerint egy `chroot(8)` környezetben futó egyszerű névfeloldást végző szerver, amely a helyi IPv4 interfészen (127.0.0.1) fogadja a kéréseket. Ezzel a beállítással a következő parancson keresztül tudjuk elindítani:

```
# /etc/rc.d/named onestart
```

Ha engedélyezni akarjuk a **named** demont minden egyes rendszerindításkor, tegyük a következő sort az `/etc/rc.conf` állományba:

```
named_enable="YES"
```

Értelemszerűen az `/etc/namedb/named.conf` tele van olyan beállítási lehetőségekkel, amelyek meghaladják ennek a leírásnak a kereteit. Ha viszont kíváncsiak vagyunk a FreeBSD-ben a **named** indításához használt beállításokra, akkor az `/etc/defaults/rc.conf` állományban nézzük meg `named_*` változókat és olvassuk át az `rc.conf(5)` man oldalt. Emellett még a 11.7 Szakaszt is hasznos lehet elolvasni.

29.6.6. A konfigurációs állományok

A **named** beállításait tartalmazó állományok pillanatnyilag az `/etc/namedb` könyvtárban találhatóak és hacsak nem egy egyszerű névfeloldóra tartunk igényt, akkor a használata előtt módosítanunk is kell. Itt ejtjük meg a beállítások nagy részét.

29.6.6.1. /etc/namedb/named.conf

```
// $FreeBSD$
//
// Részletesebb leírást a named.conf(5) és named(8) man oldalakon, valamint
// a /usr/share/doc/bind9 könyvtárban találhatunk.
//
// Ha egy hitelesített szervert akarunk beállítani, akkor igyekezzünk
// a névfeloldás összes finom részletével pontosan tisztában lenni.
// Ugyanis még a legkisebb hibákkal is egyrészt elvághatunk gépeket az
// internet-lérésétől, vagy másrészt felesleges forgalmat tudunk
// generálni
//
```

```
options {  
    // A chroot könyvtárhoz relatív elérési út, amennyiben létezik  
    directory      "/etc/namedb";  
    pid-file       "/var/run/named/pid";  
    dump-file      "/var/dump/named_dump.db";  
    statistics-file "/var/stats/named.stats";  
  
// Ha a named démont csak helyi névfeloldóként használjuk, akkor ez  
// egy biztonságos alapbeállítás. Ha viszont a named démon az egész  
// hálózatunkat is kiszolgálja, akkor ezt a beállítást tegyük  
// megjegyzésbe, vagy adjuk meg egy rendes IP-címet, esetleg  
// töröljük ki.  
listen-on        { 127.0.0.1; };  
  
// Ha rendszerünkön engedélyezett az IPv6 használata, akkor a helyi  
// névfeloldó használatához ezt a sort vegyük ki a megjegyzésből.  
// A hálózatunk többi részéről pedig úgy lehet elérni, ha itt megadunk  
// egy IPv6 címet, vagy az "any" kulcsszót.  
listen-on-v6     { ::1; };  
  
// Az alábbi zónákat már a lentebb található üres zónák eleve lefedik.  
// Ha tehát a lenti üres zónákat kivesszük a konfigurációból, akkor  
// ezeket a sorokat is tegyük megjegyzésbe.  
disable-empty-zone "255.255.255.255.IN-ADDR.ARPA";  
disable-empty-zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA";  
disable-empty-zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA";  
  
// Ha a szolgáltatónk névszervert is elérhetővé tett számunkra, akkor  
// itt adjuk meg annak az IP-címét és engedélyezzük az alábbi sort.  
// Ezzel egyben kihasználjuk a gyorsítóátvitelt is, így mérsékeljünk az  
// internet felé mozgó névfeloldásokat.  
/*  
forwarders {  
    127.0.0.1;  
};  
*/  
  
// Ha a 'forwarders' rész nem üres, akkor alapértelmezés szerint a  
// 'forward first' értékkel rendelkezik. Ekkor a kérést a helyi szerver  
// kapja abban az esetben, amikor a 'forwarders' részben megadott  
// serverek nem tudják megválaszolni. Emellettt a névszerverben a  
// következő sor hozzáadásával letilthatjuk, hogy önmagától ne  
// kezdeményezzen kéréseket:  
// forward only;  
  
// Ha a kérések továbbítását az /etc/resolv.conf állományban megadott  
// bejegyzések mentén szeretnénk automatikusan konfigurálni, akkor vegyük  
// ki a megjegyzésből az alábbi sort és adjuk hozzá az /etc/rc.conf  
// állománynak a name_auto_forward=yes sort. Emellettt használható még a  
// named_auto_forward_only beállítás is (amely fentebb leírt funkciót  
// valósítja meg).  
include "/etc/namedb/auto_forward.conf";
```

Ahogy arról a megjegyzésekben is szó esik, úgy tudjuk aktiválni a gyorsítótárat, ha megadjuk a forwarders beállítását. Normális körülmények között a névszerver az interneten az egyes névszervereket rekurzívan fogja keresni egészen addig, amíg meg nem találja a keresett választ. Az iménti beállítás engedélyezésével azonban először a szolgáltató névszerverét (vagy az általa kijelölt névszervert) fogjuk megkérdezni, a saját gyorsítótárából. Ha a szolgáltató kérdéses névszervere egy gyakran használt, gyors névszerver, akkor ezt érdemes bekapcsolnunk.

Figyelem! A 127.0.0.1 megadása *nem* működik. Mindenképpen írjuk át a szolgáltatónk névszerverének IP-címére.

```
/*
    A BIND legújabb változataiban alapértelmezés szerint minden egyes
    kimenő kérésnél más, véletlenszerűen választott UDP portot
    használnak, ezáltal jelentős mértékben csökkenthető a gyorsítótár
    meghamisíthatóságának (cache poisoning) esélye. Javasoljuk
    mindenkinek, hogy használják ki ezt a lehetőséget és eszerint
    állítsák be a tűzfalakat.

    Ha nem sikerül a tűzfalat hozzáigazítani ehhez a
    viselkedéshez AKKOR ÉS CSAK IS AKKOR engedélyezzük a lenti
    beállítást. Alkalmazásával sokkal kevésbé lesz ellenálló a
    névszerver a különböző hamisítási kísérletekkel szemben,
    ezért lehetőség szerint kerüljük el.

    Az NNNNN helyére egy 49160 és 65530 közti számot kell
    beírunk.
*/
// query-source address * port NNNNN;
};

// Ha engedélyezzük a helyi névszerveret, akkor az /etc/resolv.conf
// állományban első helyen megadni a 127.0.0.1 címet. Sőt, az
// /etc/rc.conf állományból se felejtjük ki.

// A hagyományos "root-hints" megoldás. Használjuk ezt VAGY a lentebb
// megadott alárendelt zónákat.
zone "." { type hint; file "named.root"; };

/*
    Több szempontból is előnyös, ha a következő zónákat alárendeljük a
    gyökér névfeloldó szervereknek:
    1. A helyi felhasználók kéréseit gyorsabban tudjuk feloldalni.
    2. A gyökérszerverek felé nem megy semmilyen hamis forgalom.
    3. A gyökérszerverek meghibásodása vagy elosztott DoS támadás
        esetén rugalmasabban tudunk reagálni.

    Másfelől azonban ez a módszer a "hints" állomány alkalmazásával
    szemben több felügyeletet igényel, mivel figyelniünk kell, nehogy
    egy váratlan meghibásodás működésképtelenné tegye a
    szerverünket. Ez a megoldás leginkább a sok klienst kiszolgáló
    névszerverek esetén bizonyulhat jövedelmezőbbnek. Óvatosan
    bányjunk vele!
```

A módszer alkalmazásához vegyük ki a megjegyzésből a következő bejegyzéseket és tegyük megjegyzésbe a fenti hint zónát.

```
*/

zone "." {
    type slave;
    file "slave/root.slave";
    masters {
        192.5.5.241;    // F.ROOT-SERVERS.NET.
    };
    notify no;
};

zone "arpa" {
    type slave;
    file "slave/arpa.slave";
    masters {
        192.5.5.241;    // F.ROOT-SERVERS.NET.
    };
    notify no;
}

zone "in-addr.arpa" {
    type slave;
    file "slave/in-addr.arpa.slave";
    masters {
        192.5.5.241;    // F.ROOT-SERVERS.NET.
    };
    notify no;
};
*/

/*    Az alábbi zónák helyi kiszolgálásával meg tudjuk akadályozni, hogy
    a belőlük indított kérések elhagyják a hálózatunkat és a elérjük
    a gyökér névfeloldó szervereket. Ez a megközelítés két komoly
    előnnyel rendelkezik:
    1. A helyi felhasználók kéréseit gyorsabban tudjuk
        megválaszolni.
    2. A gyökérszerverek felé nem továbbítódik semmilyen hamis
        forgalom.
*/

// RFC 1912
zone "localhost"          { type master; file "master/localhost-forward.db"; };
zone "127.in-addr.arpa"    { type master; file "master/localhost-reverse.db"; };
zone "255.in-addr.arpa"    { type master; file "master/empty.db"; };

// A helyi IPv6 címek részére létrehozott RFC 1912-szerű zóna
zone "0.ip6.arpa"         { type master; file "master/localhost-reverse.db"; };

// "Ez" a hálózat (RFC 1912 és 3330)
zone "0.in-addr.arpa"      { type master; file "master/empty.db"; };

// Magáncélú hálózatok (RFC 1918)
```

```

zone "10.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "16.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "17.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "18.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "19.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "20.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "21.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "22.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "23.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "24.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "25.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "26.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "27.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "28.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "29.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "30.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "31.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "168.192.in-addr.arpa" { type master; file "master/empty.db"; };

// Helyi link/APIPA (RFC 3330 és 3927)
zone "254.169.in-addr.arpa" { type master; file "master/empty.db"; };

// Dokumentációs próbahálózat (RFC 3330)
zone "2.0.192.in-addr.arpa" { type master; file "master/empty.db"; };

// Útválasztási teljesítmény tesztelésére (RFC 3330)
zone "18.198.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "19.198.in-addr.arpa"  { type master; file "master/empty.db"; };

// Az IANA részére fentartott - a régi E osztályú címtér
zone "240.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "241.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "242.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "243.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "244.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "245.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "246.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "247.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "248.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "249.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "250.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "251.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "252.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "253.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "254.in-addr.arpa"      { type master; file "master/empty.db"; };

// Hozzárendelés nélküli IPv6-címek (RFC 4291)
zone "1.ip6.arpa"            { type master; file "master/empty.db"; };
zone "3.ip6.arpa"            { type master; file "master/empty.db"; };
zone "4.ip6.arpa"            { type master; file "master/empty.db"; };
zone "5.ip6.arpa"            { type master; file "master/empty.db"; };
zone "6.ip6.arpa"            { type master; file "master/empty.db"; };
zone "7.ip6.arpa"            { type master; file "master/empty.db"; };

```

```

zone "8.ip6.arpa"           { type master; file "master/empty.db"; };
zone "9.ip6.arpa"           { type master; file "master/empty.db"; };
zone "a.ip6.arpa"           { type master; file "master/empty.db"; };
zone "b.ip6.arpa"           { type master; file "master/empty.db"; };
zone "c.ip6.arpa"           { type master; file "master/empty.db"; };
zone "d.ip6.arpa"           { type master; file "master/empty.db"; };
zone "e.ip6.arpa"           { type master; file "master/empty.db"; };
zone "0.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "1.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "2.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "3.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "4.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "5.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "6.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "7.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "8.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "9.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "a.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "b.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "0.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "1.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "2.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "3.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "4.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "5.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "6.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "7.e.f.ip6.arpa"       { type master; file "master/empty.db"; };

// IPv6 ULA (RFC 4193)
zone "c.f.ip6.arpa"         { type master; file "master/empty.db"; };
zone "d.f.ip6.arpa"         { type master; file "master/empty.db"; };

// IPv6 helyi link (RFC 4291)
zone "8.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "9.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "a.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "b.e.f.ip6.arpa"       { type master; file "master/empty.db"; };

// Elavult IPv6 helyi címek (RFC 3879)
zone "c.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "d.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "e.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "f.e.f.ip6.arpa"       { type master; file "master/empty.db"; };

// Az IP6.INT már elavult (RFC 4159)
zone "ip6.int"               { type master; file "master/empty.db"; };

// FONTOS: Ne használjuk ezeket az IP-címeket, mert nem valódiak,
// csupán illusztrációs és dokumentációs célokból adtuk meg!
//
// Az alárendelt zónák beállításaira vonatkozó bejegyzések. Érdekes
// illet beállítani legalább ahhoz a zónához, amelyhez a tartományunk is
// tartozik. Az elsődleges névszerverhez tartozó IP-címet érdeklődjük meg

```

```
// az illetékes hálózati rendszergazdától.
//
// Soha ne felejtsünk el megadni zónát az inverz kereséshez! A neve az IP-cím
// tagjainak fordított sorrendjéből // származik, amelyhez hozzátoldunk még egy
// ".IN-ADDR.ARPA" (illetve IPv6 esetén ".IP6.ARPA") részt.
//
// Mielőtt nekilátnánk egy elsődleges zóna beállításának, gondoljuk
// végig, hogy tényleg a megfelelő szinten ismerjük a névfeloldás és
// a BIND működését. Gyakran ugyanis egyáltalán nem nyilvánvaló
// csapdába tudunk esni. Egy alárendelt zóna beállítása általában sokkal egyszerűbb feladat.
//
// FONTOS: Ne kövessük vakon a most következő példát :-) Helyette inkább
// valódi neveket és címeket adjunk meg.

/* Példa dinamikus zónára
key "mintaorgkulcs" {
    algorithm hmac-md5;
    secret "sf87HJqjkqh8ac87a021la==";
};
zone "minta.org" {
    type master;
    allow-update {
        key "mintaorgkulcs";
    };
    file "dynamic/minta.org";
};
*/

/* Példa inverz alárendelt zónákra
zone "1.168.192.in-addr.arpa" {
    type slave;
    file "slave/1.168.192.in-addr.arpa";
    masters {
        192.168.1.1;
    };
};
*/
```

A `named.conf` állományban tehát így adhatunk meg közvetlen és inverz alárendelt zónákat.

Minden egyes újabb kiszolgált zónához az egy új bejegyzést kell felvenni a `named.conf` állományban.

Például a `minta.org` címhez tartozó legegyszerűbb ilyen bejegyzés így néz ki:

```
zone "minta.org" {
    type master;
    file "master/minta.org";
};
```

Ez egy központi zóna, ahogy arról a `type` mező, vagyis a típusa is árulkodik. Továbbá a `file` mezőben láthatjuk, hogy a hozzá tartozó információkat az `/etc/namedb/master/minta.org` állományban tárolja.

```
zone "minta.org" {
    type slave;
```

```
file "slave/minta.org";
};
```

Az alárendelt esetben a zónához tartozó információkat a zóna központi szerverétől kapjuk meg és megadott állományban mentjük el. Ha valamiért a központi szerver leáll vagy nem érhető el, akkor az alárendelt szerver az átküldött zóna információk alapján képes helyette kiszolgálni a kéréseket.

29.6.6.2. A zóna állományok

A `minta.org` címhez tartozó példa központi zóna állomány (amely az `/etc/namedb/master/néven.org` érhető el) tartalma az alábbi:

```
$TTL 3600          ; alapértelmezés szerint 1 óra
minta.org.         IN      SOA      ns1.minta.org. admin.minta.org. (
                                2006051501      ; sorozatszám
                                10800           ; frissítés
                                3600            ; ismétlés
                                604800          ; lejárat
                                300             ; TTL negatív válasz
                                )

; névszerverek
                IN      NS      ns1.minta.org.
                IN      NS      ns2.minta.org.

; MX rekordok
                IN      MX 10    mx.minta.org.
                IN      MX 20    levelezes.minta.org.

                IN      A        192.168.1.1

; a gépek nevei
localhost       IN      A        127.0.0.1
ns1              IN      A        192.168.1.2
ns2              IN      A        192.168.1.3
mx              IN      A        192.168.1.4
levelezes        IN      A        192.168.1.5

; álnevek
www             IN      CNAME     minta.org.
```

A “.”-ra végződő hálózati nevek abszolút nevek, míg minden más “.” nélküli név az ősére vezethető vissza (tehát relatív). Például az `ns1` névből az `ns1.minta.org` keletkezik.

A zóna állományok felépítése a következő:

```
rekordnév      IN rekordtípus  érték
```

A névfeloldásban leggyakrabban alkalmazott rekordok típusai:

SOA

a zóna fennhatóságának kezdete

NS

egy hitelesített névszerver

A

egy gép címe

CNAME

egy álnév kanonikus neve

MX

levélváltó

PTR

mutató a tartománynévre (az inverz feloldás használja)

```
minta.org. IN SOA ns1.minta.org. admin.minta.org. (
                                2006051501      ; sorozatszám
                                10800             ; 3 óránként frissítsünk
                                3600              ; 1 óra után próbálkozzunk újra
                                604800            ; 1 hét után jár le
                                300 )             ; TTL negatív válasz
```

minta.org.

a tartomány neve, amely egyben a zóna őse

ns1.minta.org.

a zóna elsődleges/hitelesített névszervere

admin.minta.org.

a zónáért felelős személy neve, akinek az e-mail címét a “@” behelyettesítésével kapjuk meg. (Tehát a <admin@example.org> címből admin.example.org lesz.)

2006051501

az állomány sorozatszáma. Ezt a zóna állomány módosításakor mindig növelnünk kell. Manapság a rendszergazdák a sorozatszámot ééééhhnnvv alakban adják meg. A 2006051501 tehát azt jelenti, hogy az állományt 2006. május 15-én módosították utoljára, és a 01 pedig arra utal, hogy aznap először. A sorozatszám megadása fontos az alárendelt névszerverek számára, mivel így tudják megállapítani, hogy a zóna mikor változott utoljára.

```
IN NS          ns1.minta.org.
```

Ez egy NS bejegyzés. A zónához tartozó minden hitelesített névszervernek lennie kell legalább egy ilyen bejegyzésének.

```
localhost      IN      A      127.0.0.1
ns1             IN      A      192.168.1.2
ns2            IN      A      192.168.1.3
```

```
mx          IN      A      192.168.1.4
levelezés   IN      A      192.168.1.5
```

Az A rekord egy gép nevét adja meg. Ahogy a fenti példából is kiderül, az `ns1.minta.org` név a `192.168.1.2` címre képződik le.

```
          IN      A      192.168.1.1
```

Ez a sor `192.168.1.1` címet rendeli az aktuális őshöz, amely jelen esetünkben az `example.org`.

```
www         IN  CNAME      @
```

A kanonikus neveket tároló rekordokat általában egy gép álneveihez használjuk. Ebben a példában a `www` a “főgép” egyik álneve, amely itt éppenséggel a `minta.org` (`192.168.1.1`) tartományneve. A CNAME rekordok mellé más típusú rekordokat ugyanarra a hálózati névre soha ne adjunk meg.

```
          IN  MX    10      levelezés.minta.org.
```

Az MX rekord adja meg, hogy milyen levelező szerverek felelősek a zónába érkező levelek fogadásáért. A `levelezés.minta.org` a levelező szerver hálózati neve, ahol a 10 az adott levelező szerver prioritása.

Több levelező szerver is megadható 10-es, 20-as stb. prioritásokkal. A `minta.org` tartományon belül először mindig a legnagyobb MX prioritással rendelkező levelező szervernek próbáljuk meg továbbítani a leveleket (a legkisebb prioritási értékkel rendelkező rekord), majd ezután a második legnagyobbban stb. egészen addig, amíg a levelet tovább nem küldtük.

Az `in-addr.arpa` zóna állományok (inverz DNS) esetén ugyanez a felépítés, kivéve, hogy a PTR típusú bejegyzések szerepelnek az A és CNAME helyett.

```
$TTL 3600
```

```
1.168.192.in-addr.arpa. IN SOA ns1.minta.org. admin.minta.org. (
                        2006051501      ; sorozatszám
                        10800           ; frissítés
                        3600            ; ismétlés
                        604800          ; lejárat
                        300 )           ; TTL negatív válasz
```

```
          IN      NS      ns1.minta.org.
          IN      NS      ns2.minta.org.
```

```
1      IN      PTR      minta.org.
2      IN      PTR      ns1.minta.org.
3      IN      PTR      ns2.minta.org.
4      IN      PTR      mx.minta.org.
5      IN      PTR      levelezés.minta.org.
```

Ez az állomány írja le tehát a kitalált tartományunkon belül az IP-címek és hálózati nevek összerendelését.

Érdemes megemlíteni, hogy a PTR rekordok jobb oldalán álló nevek mindegyikének teljes hálózati névnek kell lennie (vagyis “.” karakterrel kell végződnie).

29.6.7. A gyorsítótárazó névszerver

A gyorsítótárazó névszerver az a névszerver, amely elsődleges feladata a rekurzív kérések kiszolgálása. Egyszerűen továbbítja a beérkező kéréseket, majd megjegyzi azokat, így később közvetlenül tud válaszolni.

29.6.8. Biztonság

Habár a névfeloldás szempontjából a BIND a legelterjedtebb, a biztonságosságával azért akadnak gondok. Gyakran találunk benne potenciális és kihasználható biztonsági réseket.

A FreeBSD azonban a **named** démon automatikusan egy chroot(8) környezetbe helyezi. Emellett még léteznek további más védelmi mechanizmusok is, amelyek segítségével el tudjuk kerülni a névfeloldást célzó esetleges támadásokat.

Sosem árt olvasgatni a CERT (<http://www.cert.org/>) által kiadott biztonsági figyelmeztetéseket és feliratkozni a FreeBSD security notifications levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications>) címére, hogy folyamatosan értesüljünk az interneten és a FreeBSD-ben talált különböző biztonsági hibákról.

Tipp: Ha valamilyen gondunk támadna, akkor esetleg próbálkozzunk meg a forrásaink frissítésével és a **named** újrafordításával.

29.6.9. Egyéb olvasnivalók

A BIND/**named** man oldalai: rndc(8) named(8) named.conf(5)

- Az ISC BIND hivatalos honlapja (angolul) (<http://www.isc.org/software/bind>)
- Az ISC BIND hivatalos fóruma (angolul) (<http://www.isc.org/software/guild>)
- O'Reilly DNS and BIND 5th Edition (<http://www.oreilly.com/catalog/dns5/>)
- RFC1034 - Domain Names - Concepts and Facilities (<http://www.rfc-editor.org/rfc/rfc1034.txt>)
- RFC1035 - Domain Names - Implementation and Specification (<http://www.rfc-editor.org/rfc/rfc1035.txt>)

29.7. Az Apache webservert

Készítette: Murray Stokely.

29.7.1. Áttekintés

A FreeBSD szolgálja ki a legforgalmasabb honlapok nagy részét szerte a világban. A mögöttük álló webszerverek általában az **Apache webservert** alkalmazzák. Az **Apache** használatához szükséges csomagok megtalálhatóak a FreeBSD telepítőlemezén is. Ha a FreeBSD első telepítésekor még nem telepítettük volna az **Apache** szerverét, akkor a `www/apache13` vagy `www/apache12` portból tudjuk feltenni.

Az **Apache** szervert sikeres telepítését követően be kell állítanunk.

Megjegyzés: Ebben a szakaszban az **Apache webservert** 1.3.X változatát mutatjuk be, mivel ezt használják a legtöbb esetben FreeBSD alatt. Az **Apache 2.X** rengeteg új technológiát vezetett be, de ezekkel itt most nem foglalkozunk. Az **Apache 2.X** változatával kapcsolatban keressük fel a <http://httpd.apache.org/> oldalt.

29.7.2. Beállítás

Az **Apache webservert** konfigurációs állománya FreeBSD alatt `/usr/local/etc/apache/httpd.conf` néven található. Ez az állomány egy szokványos UNIX-os szöveges konfigurációs állomány, ahol a megjegyzéseket egy `#` karakterrel vezetjük be. Az itt használható összes lehetséges beállítási lehetőség átfogó ismertetése meghaladná az egész kézikönyv határait, ezért most csak a leggyakrabban módosított direktívákat fogjuk ismertetni.

```
ServerRoot "/usr/local"
```

Ez adja meg az **Apache** számára az alapértelmezett könyvtárat. A binárisai ezen belül a `bin` és `sbin` alkönyvtárakban, a konfigurációs állományai pedig az `etc/apache` könyvtárban tárolódnak.

```
ServerAdmin saját@címünk.az.interneten
```

Erre a címre küldhetik nekünk a szerverrel kapcsolatos hibákat. Ez a cím egyes szerver által generált oldalakon jelenik meg, például hibák esetében.

```
ServerName www.minta.com
```

A `ServerName` segítségével meg tudjuk adni, hogy milyen nevet küldjön vissza a szerver a klienseknek olyankor, ha az nem egyezne meg a jelenlegivel (vagyis a `www` nevet használjuk a gépünk valódi neve helyett).

```
DocumentRoot "/usr/local/www/data"
```

A `DocumentRoot` adja meg azt a könyvtárat, ahonnan kiszolgáljuk a dokumentumokat. Alapértelmezés szerint az összes kérés erre a könyvtárra fog vonatkozni, de a szimbolikus linkek és az álnevek akár más helyekre is mutathatnak.

A változtatások végrehajtása előtt mindig is jó ötlet biztonsági másolatot készíteni az **Apache** konfigurációs állományairól. Ahogy sikerült összerakni egy számunkra megfelelő konfigurációt, készen is állunk az **Apache** futtatására.

29.7.3. Az Apache futtatása

A többi hálózati szervertől eltérően az **Apache** nem az **inetd** szuperszerverből fut. A kliensektől érkező HTTP kérések minél gyorsabb kiszolgálásának érdekében úgy állítottuk be, hogy önállóan fusson. Ehhez egy szkriptet is mellékelünk, amellyel igyekeztünk a lehető legjobban leegyszerűsíteni a szerver indítását, leállítását és újraindítását. Az **Apache** első indításához adjuk ki a következő parancsot:

```
# /usr/local/sbin/apachectl start
```

Így pedig a szervert bármikor leállíthatjuk:

```
# /usr/local/sbin/apachectl stop
```

Ha valamilyen okból megváltoztattuk volna a szerver beállításait, akkor ezen a módon tudjuk újraindítani:

```
# /usr/local/sbin/apachectl restart
```

Ha a jelenleg megnyitott kapcsolatok felbontása nélkül akarjuk újraindítani az **Apache** szerveret, akkor ezt írjuk be:

```
# /usr/local/sbin/apachectl graceful
```

Mindezekről az apachectl(8) man oldalon találunk bővebb leírást.

Amennyiben szükségünk lenne az **Apache** elindítására a rendszer indításakor, akkor a következő sort vegyünk fel az `/etc/rc.conf` állományba:

```
apache_enable="YES"
```

Az **Apache** 2.2 esetében:

```
apache22_enable="YES"
```

Amikor az **Apache** httpd nevű programjának szeretnénk további parancssori paramétereket átadni a rendszer indítása során, akkor ezeket így tudjuk megadni az `rc.conf` állományban:

```
apache_flags=" "
```

Most, miután a webszerverünk működik, a böngészőnkkel mindezt ellenőrizni is tudjuk a `http://localhost/` cím beírásával. Ilyenkor az alapértelmezés szerinti `/usr/local/www/data/index.html` állomány tartalmát láthatjuk.

29.7.4. Virtuális nevek

Az **Apache** a virtuális nevek használatának két különböző módját ismeri. Ezek közül az első módszer a név alapú virtualizáció (Name-based Virtual Hosting). Ilyenkor a kliens HTTP/1.1 fejlécéből próbálja meg a szerver megállapítani a hivatkozási nevet. Segítségével több tartomány is osztozhat egyetlen IP-címen.

Az **Apache** név alapú virtualizációjának beállításához az alábbi beállítást kell hozzátennünk a `httpd.conf` állományhoz:

```
NameVirtualHost *
```

Ha a webszerverünk neve `www.tartomany.hu`, és hozzá egy `www.valamilyenmasiktartomany.hu` virtuális nevet akarunk megadni, akkor azt a következőképpen tehetjük meg a `httpd.conf` állományon belül:

```
<VirtualHost *>
ServerName www.tartomany.hu
DocumentRoot /www/tartomany.hu
</VirtualHost>

<VirtualHost *>
ServerName www.valamilyenmasiktartomany.hu
DocumentRoot /www/valamilyenmasiktartomany.hu
</VirtualHost>
```

A címek és elérési utak helyére helyettesítsük be a használni kívánt címeket és elérési utakat.

A virtuális nevek beállításának további részleteivel kapcsolatosan keressük fel az **Apache** hivatalos dokumentációját a `http://httpd.apache.org/docs/vhosts/` címen (angolul).

29.7.5. Apache-modulok

Az alap szerver képességeinek kiegészítéséhez több különböző **Apache** modul áll rendelkezésünkre. A FreeBSD Portgyűjteménye az **Apache** telepítése mellett lehetőséget ad a népszerűbb bővítményeinek telepítésére is.

29.7.5.1. mod_ssl

A **mod_ssl** modul az OpenSSL könyvtár használatával valósít meg erős titkosítást a biztonságos socket réteg második, illetve harmadik verziójával (Secure Sockets Layer, SSL v2/v3) és a biztonságos szállítási rétegbeli (Transport Layer Security v1) protokoll segítségével. Ez a modul mindent biztosít ahhoz, hogy a megfelelő hatóságok által aláírt tanúsítványokat tudjunk kérni, és ezáltal egy védett webszervert futtassunk FreeBSD-n.

Ha még nem telepítettünk volna fel az **Apache** szervert, akkor a `www/apache13-modssl` porton keresztül a **mod_ssl** modullal együtt is fel tudjuk rakni az **Apache** 1.3.X változatát. Az SSL támogatása pedig már az **Apache** 2.X `www/apache22` porton keresztül elérhető változataiban alapértelmezés szerint engedélyezett.

29.7.5.2. Kapcsolódás nyelvekhez

Mindegyik nagyobb szkriptnyelvhez létezik egy külön Apache-modul, amelyek segítségével komplett **Apache**-modulokat tudunk készíteni az adott nyelven. Gyakran a dinamikus honlapok is így próbálják a szerverbe épített belső értelmezőn keresztül a külső értelmező indításából és benne a szkriptek lefuttatásából fakadó költségeket megspórolni, ahogy erről a következő szakaszokban olvashatunk.

29.7.6. Dinamikus honlapok

Az utóbbi évtizedben egyre több vállalkozás fordult az internet felé bevételeik és részesedéseinek növelésének reményében, amivel egyre jobban megnőtt az igény a dinamikus honlapokra is. Miközben bizonyos cégek, mint például a Microsoft, a saját fejlesztésű termékeikbe építettek be ehhez támogatást, addig a nyílt forrásokkal foglalkozó közösség sem maradt tétlen és felvette a kesztyűt. A dinamikus tartalom létrehozásához többek közt Django, Ruby on Rails, a **mod_perl** és a **mod_php** modulok használhatóak.

29.7.6.1. Django

A Django egy BSD típusú licensszel rendelkező keretrendszer, amelynek használatával nagy teljesítményű és elegáns webes alkalmazásokat tudunk gyorsan kifejleszteni. Tartalmaz egy objektum-relációs leképezőt, így az adattípusokat Python-objektumokként tudjuk leírni, és ezekhez az objektumokhoz egy sokrétű, dinamikus adatbázis hozzáférést nyújtó alkalmazásfejlesztői felületet, így a fejlesztőknek egyetlen SQL utasítást sem kell megírniuk. Találhatunk még benne továbbá egy bővíthető sablonrendszert, amelynek köszönhetően az alkalmazás belső működése elválasztható a HTML-beli megjelenésétől.

A Django működéséhez a **mod_python** modulra, az **Apache** szerverre és egy tetszőlegesen választott SQL alapú adatbázisrendszerre van szükség. A hozzá tartozó FreeBSD port mindezeket automatikusan telepíti a megadott beállítások szerint.

Példa 29-3. A Django telepítése az Apache, mod_python3 és a PostgreSQL használatával

```
# cd /usr/ports/www/py-django; make all install clean -DWITH_MOD_PYTHON3 -DWITH_POSTGRESQL
```

Miután a Django és a hozzá szükséges komponensek felkerültek rendszerünkre, hozzunk létre egy könyvtárat a leendő Django projektünknek és állítsuk be az Apache szerveret, hogy az oldalunk belül a megadott linkekre a saját alkalmazásunkat hívja meg a beágyazott Python-értelmezőn keresztül.

Példa 29-4. Az Apache beállítása a Django és mod_python használatához

A következő sort kell hozzátennünk a `httpd.conf` állományhoz, hogy az Apache bizonyos linkeket a webes alkalmazás felé irányítson át:

```
<Location "/">
    SetHandler python-program
    PythonPath "[ '/a/django/csomagok/helye/' ] + sys.path"
    PythonHandler django.core.handlers.modpython
    SetEnv DJANGO_SETTINGS_MODULE azoldalam.beallitasai
    PythonAutoReload On
    PythonDebug On
</Location>
```

29.7.6.2. Ruby on Rails

A Ruby on Rails egy olyan másik nyílt forráskódú keretrendszer, amivel lényegében egy teljes fejlesztői készletet kapunk és amelyet kifejezetten arra élezték ki, hogy segítségével a webfejlesztők sokkal gyorsabban tudjanak haladni és a komolyabb alkalmazások gyorsabb elkészítése se okozzon nekik gondot. A Portrgyűjteményből pillanatok alatt telepíthető.

```
# cd /usr/ports/www/rubygem-rails; make all install clean
```

29.7.6.3. mod_perl

Az **Apache** és Perl egyesítésén fáradozó projekt a Perl programozási nyelv és az **Apache webservert** erejének összehangolásán dolgozik. A **mod_perl** modulon keresztül Perlben vagyunk képesek modulokat készíteni az **Apache** szerverhez. Ráadásul a szerverben egy belső állandó értelmező is található hozzá, ezzel igyekeznek megspórolni a külső értelmező és a Perl indításából keletkező többletköltségeket.

A **mod_perl** több különböző módon állítható munkába. A **mod_perl** használatához nem szabad elfelejtenünk, hogy a **mod_perl** 1.0-ás verziója csak az **Apache** 1.3 változatával működik, és a **mod_perl** 2.0-ás változata pedig csak az **Apache** 2.X változataival. A **mod_perl** 1.0 a `www/mod_perl` portból telepíthető, valamint a statikusan beépített változata a `www/apache13-modperl` portban található. A **mod_perl** 2.0 a `www/mod_perl2` portból rakható fel.

29.7.6.4. mod_php

Írta: Tom Rhodes.

A PHP, vagy másik nevén “PHP, a hipertext feldolgozó” egy általános célú szkriptnyelv, amelyet kifejezetten honlapok fejlesztéséhez hoztak létre. A szabványos HTML ágyazható nyelv felépítésében a C, Java és Perl nyelveket ötvözi annak elérése érdekében, hogy ezzel segítse a fejlesztőket a dinamikusan generált oldalak minél gyorsabb megírásában.

A PHP5 támogatását úgy tudjuk hozzáadni az **Apache** webservert, ha telepítjük a `lang/php5` portot.

Ha a `lang/php5` portot most telepítjük először, akkor a vele kapcsolatos beállításokat tartalmazó `OPTIONS` menü automatikusan megjelenik. Ha ezzel nem találkozánk, mert például valamikor korábban már felraktuk volna a `lang/php5` portot, akkor a port könyvtárában következő parancs kiadásával tudjuk újra visszahozni:

```
# make config
```

A beállítások között jelöljük be az `APACHE` opciót, amelynek eredményeképpen létrejön az **Apache** webszerverhez használható **`mod_php5`** betölthető modul.

Megjegyzés: A PHP4 modul még ma is rengeteg szerver használja több különböző okból (például kompatibilitási problémák vagy a már korábban kiadott tartalom miatt). Ha tehát a **`mod_php5`** helyett inkább a **`mod_php4`** modulra lenne szükségünk, akkor a `lang/php4` portot használjuk. A `lang/php4` portnál is megtalálhatjuk a `lang/php5` fordítási idejű beállításainak nagy részét.

Az iméntiek révén települnek és beállítódnak a dinamikus PHP alkalmazások támogatásához szükséges mouldok. Az `/usr/local/etc/apache/httpd.conf` állományban ellenőrizni is tudjuk, hogy az alábbi részek megjelentek-e:

```
LoadModule php5_module          libexec/apache/libphp5.so

AddModule mod_php5.c
    <IfModule mod_php5.c>
        DirectoryIndex index.php index.html
    </IfModule>
    <IfModule mod_php5.c>
        AddType application/x-httpd-php .php
        AddType application/x-httpd-php-source .phps
    </IfModule>
```

Ahogy befejeződött a művelet, a PHP modul betöltéséhez mindösszesen az `apachectl` paranccsal kell óvatosan újraindítanunk a webszervert:

```
# apachectl graceful
```

A PHP jövőbeni frissítéseihez már nem lesz szükségünk a `make config` parancsra, mivel a korábban kiválasztott `OPTIONS` menün belüli beállításainkat a FreeBSD Portgyűjteményéhez tartozó keretrendszer automatikusan elmenti.

A PHP FreeBSD-ben megtalálható támogatása kifejezetten moduláris, ezért az alap telepítése igencsak korlátozott. A további elemek hozzáadásához a `lang/php5-extensions` portot tudjuk használni. A port egy menüvezérelt felületet nyújt a PHP különböző bővítményeinek telepítéséhez. Az egyes bővítményeket azonban a megfelelő portok használatával is fel tudjuk rakni.

Például PHP5 modulhoz úgy tudunk támogatást adni a **MySQL** adatbázis szerverhez, ha telepítjük a `databases/php5-mysql` portot.

Miután telepítettünk egy bővítményt, az **Apache** szerverrel újra be kell töltenünk a megváltozott beállításokat:

```
# apachectl graceful
```

29.8. Állományok átvitele (FTP)

Készítette: Murray Stokely.

29.8.1. Áttekintés

Az adatállomány átviteli protokoll (File Transfer Protocol, FTP) a felhasználók számára lehetőséget ad az ún. FTP szerverekre állományokat feltölteni, illetve onnan állományokat letölteni. A FreeBSD alaprendszere is tartalmaz egy ilyen FTP szerverprogramot, **ftpd** néven. Ezért FreeBSD alatt egy FTP szerver beállítása meglehetősen egyszerű.

29.8.2. Beállítás

A beállítás legfontosabb lépése, hogy eldöntsük milyen hozzáféréseken át lehet elérni az FTP szerveret. Egy hétköznapi FreeBSD rendszerben rengeteg hozzáférés a különböző démonokhoz tartozik, de az ismeretlen felhasználók számára nem kellene megengednünk ezek használatát. Az `/etc/ftpusers` állományban szerepelnek azok a felhasználók, akik semmilyen módon nem érhetik el az FTP szolgáltatást. Alapértelmezés szerint itt találhatjuk az előbb említett rendszerszintű hozzáféréseket is, de ide minden további nélkül felvehetjük azokat a felhasználókat, akiknél nem akarjuk engedni az FTP elérését.

Más esetekben előfordulhat, hogy csak korlátozni akarjuk egyes felhasználók FTP elérését. Ezt az `/etc/ftpchroot` állományon keresztül tehetjük meg. Ebben az állományban a lekorlátozni kívánt felhasználókat és csoportokat írhatjuk bele. Az `ftpchroot(5)` man oldalán olvashatjuk el ennek részleteit, ezért ennek pontos részleteit itt most nem tárgyaljuk.

Ha az FTP szerverünkhöz névtelen (anonim) hozzáférést is engedélyezni akarunk, akkor ahhoz először készítenünk kell egy `ftp` nevű felhasználót a FreeBSD rendszerünkben. A felhasználók ezután az `ftp` vagy `anonymous` nevek, valamint egy tetszőleges jelszó (ez a hagyományok szerint a felhasználó e-mail címe) használatával is képesek lesznek bejelentkezni. Az FTP szerver ezután a névtelen felhasználók esetében meghívja a `chroot(2)` rendszerhívást, és ezzel lekorlátozza hozzáférésüket az `ftp` felhasználó könyvtárára.

Két szöveges állományban adhatunk meg a becsatlakozó FTP kliensek számára üdvözlő üzeneteket. Az `/etc/ftpwelcome` állomány tartalmát még a bejelentkezés előtt látni fogják a felhasználók, a sikeres bejelentkezést követően pedig az `/etc/ftpmotd` állomány tartalmát látják. Vigyázzunk, mert ennek az állománynak már a bejelentkezési környezethez képest relatív az elérése, ezért a névtelen felhasználók esetében ez konkrétan az `~ftp/etc/ftpmotd` állomány lesz.

Ahogy beállítottuk az FTP szerveret, az `/etc/inetd.conf` állományban is engedélyeznünk kell. Itt mindössze annyira lesz szükségünk, hogy eltávolítsuk a megjegyzést jelző “#” karaktert a már meglevő **ftpd** sor elől:

```
ftp      stream  tcp      nowait  root    /usr/libexec/ftpd      ftpd -l
```

Ahogy arról már a 29-1 Példa szót ejtett, az **inetd** beállításait újra be kell olvasatnunk a konfigurációs állomány megváltoztatása után. A 29.2.2 Szakasz írja le az **inetd** engedélyezésének részleteit.

Az **ftpd** önálló szerverként is elindítható. Ehhez mindössze elegendő csak a megfelelő változót beállítani az `/etc/rc.conf` állományban:

```
ftpd_enable="YES"
```

Miután megadtuk az iménti változót, a szerver el fog indulni a rendszer következő indítása során. Szükség esetén természetesen `root` felhasználóként a következő paranccsal is közvetlenül elindítható:

```
# /etc/rc.d/ftpd start
```

Most már be is tudunk jelentkezni az FTP szerverre:

```
% ftp localhost
```

29.8.3. Karbantartás

Az **ftpd** démon a `syslog(3)` használatával naplózza az üzeneteket. Alapértelmezés szerint a rendszernaplózó démon az FTP működésére vonatkozó üzeneteket az `/var/log/xferlog` állományba írja. Az FTP naplóinak helyét az `/etc/syslog.conf` állományban tudjuk módosítani:

```
ftp.info          /var/log/xferlog
```

Legyünk körültekintőek a névtelen FTP szerverek üzemeltetésekor. Azt pedig kétszer is gondoljuk meg, hogy engedélyezzük-e a névtelen felhasználók számára állományok feltöltését, hiszen könnyen azon kaphatjuk magunkat, hogy az FTP oldalunk illegális állománycserék színterévé válik vagy esetleg valami sokkal rosszabb történik. Ha mindenképpen szükségünk lenne erre a lehetőségre, akkor állítsunk be olyan engedélyeket a feltöltött állományokra, hogy a többi névtelen felhasználó ezeket a tartalmuk tüzetes ellenőrzéséig ne is olvashassa.

29.9. Állomány- és nyomtatási szolgáltatások Microsoft Windows kliensek számára (Samba)

Készítette: Murray Stokely.

29.9.1. Áttekintés

A **Samba** egy olyan elterjedt nyílt forráskódú szoftver, ami Microsoft Windows kliensek számára tesz lehetővé állomány- és nyomtatási szolgáltatásokat. Az ilyen kliensek általa helyi meghajtóként képesek elérni a FreeBSD állományrendszerét, vagy helyi nyomtatóként a FreeBSD általt kezelt nyomtatókat.

A **Samba** csomagja általában megtalálható a FreeBSD telepítőeszközén. Ha a FreeBSD-vel együtt nem raktuk fel a **Samba** csomagját, akkor ezt később `net/samba3` port vagy csomag telepítésével pótolhatjuk.

29.9.2. Beállítás

A **Samba** konfigurációs állománya a telepítés után `/usr/local/share/examples/samba/smb.conf.default` néven található meg. Ezt kell lemásolnunk `/usr/local/etc/smb.conf` néven, amelyet aztán a **Samba** tényleges használata előtt módosítanunk kell.

Az `smb.conf` állomány a **Samba** futásához használt beállításokat tartalmazza, mint például Windows kliensek számára felkínált a nyomtatók és "megosztások" adatait. A **Samba** csomagban ezen kívül találhatunk még egy **swat** nevű webes eszközt, amellyel egyszerű módon tudjuk az `smb.conf` állományt állítgatni.

29.9.2.1. A Samba webes adminisztrációs eszköze (SWAT)

A Samba webes adminisztrációs segédeszköze (Samba Web Administration Tool, SWAT) az **inetd** démonon keresztül fut démonként. Ennek megfelelően az `/etc/inetd.conf` állományban a következő sort kell kivennünk megjegyzésből, mielőtt a **swat** segítségével megkezdénénk a **Samba** beállítását:

```
swat    stream  tcp      nowait/400    root    /usr/local/sbin/swat    swat
```

Ahogy azt a 29-1 Példa is mutatja, az **inetd** démont újra kell indítanunk a megváltozott konfigurációs állományának újbóli beolvasásához.

Miután az `inetd.conf` állományban a **swat** engedélyezésre került, a böngészőnk segítségével próbáljunk meg a `http://localhost:901` címre csatlakozni. Először a rendszer `root` hozzáféréssel kell bejelentkeznünk.

Miután sikeresen bejelentkeztünk a **Samba** beállításait tárgyaló lapra, el tudjuk olvasni a rendszer dokumentációját, vagy a **Globals** földre kattintva nekiláthatunk a beállítások elvégzésének. A **Globals** részben található opciók az `/usr/local/etc/smb.conf` állomány `[global]` szekciójában található változókat tükrözik.

29.9.2.2. Általános beállítások

Akár a **swat** eszközzel, akár a `/usr/local/etc/smb.conf` közvetlen módosításával dolgozunk, a **Samba** beállítása során a következőkkel mindenképpen össze fogunk futni:

```
workgroup
```

A szerveret elérni kívánó számítógépek által használt NT tartomány vagy munkacsoport neve.

```
netbios name
```

A **Samba** szerver NetBIOS neve. Alapértelmezés szerint ez a név a gép hálózati nevének első tagja.

```
server string
```

Ez a szöveg jelenik meg akkor, ha például a `net view` paranccsal vagy valamilyen más hálózati segédprogrammal kérdezzük le a szerver beszédesebb leírását.

29.9.2.3. Biztonsági beállítások

A `/usr/local/etc/smb.conf` állományban a két legfontosabb beállítás a választott biztonsági modell és a kliensek felhasználói jelszavainak tárolásához használt formátum. Az alábbi direktívák vezérlik ezeket:

```
security
```

Itt a két leggyakoribb beállítás a `security = share` és a `security = user`. Ha a kliensek a FreeBSD gépen található felhasználói neveiket használják, akkor felhasználói szintű védelemre van szükségünk (tehát a `user` beállításra). Ez az alapértelmezett biztonsági házirend és ilyenkor a klienseknek először be kell jelentkezniük a megosztott erőforrások eléréséhez.

A megosztás (`share`) szintű védelem esetében, a klienseknek nem kell a szerveren érvényes felhasználói névvel és jelszóval rendelkezniük a megosztott erőforrások eléréséhez. Ez volt az alapbeállítás a **Samba** korábbi változataiban.

passdb backend

A **Samba** számos különböző hitelesítési modellt ismer. A klienseket LDAP, NIS+, SQL adatbázis vagy esetleg egy módosított jelszó állománnyal is tudjuk hitelesíteni. Az alapértelmezett hitelesítési módszer a `smbpasswd`, így itt most ezzel foglalkozunk.

Ha feltesszük, hogy az alapértelmezett `smbpasswd` formátumot választottuk, akkor a **Samba** úgy fogja tudni hitelesíteni a klienseket, ha előtte létrehozzuk a `/usr/local/private/smbpasswd` állományt. Ha a Windows-os kliensekkel is el akarjuk érni a UNIX-os felhasználói hozzáféréseinket, akkor használjuk a következő parancsot:

```
# smbpasswd -a felhasználónév
```

Megjegyzés: A **Samba** a 3.0.23c verziójától kezdődően a hitelesítéshez szükséges állományokat a `/usr/local/etc/samba` könyvtárban tárolja. A felhasználói hozzáférések hozzáadására innentől már a `tdbsam` parancs használata javasolt:

```
# pdbedit -a -u felhasználónév
```

A hivatalos Samba HOGYAN (<http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/>) ezekről a beállításokról szolgál további információkkal (angolul). Viszont az itt vázolt alapok viszont már elegendőek a **Samba** elindításához.

29.9.3. A Samba elindítása

A `net/samba3` port a **Samba** irányítására egy új indító szkriptet tartalmaz. A szkript engedélyezéséhez, tehát általa a **Samba** elindításának, leállításának és újraindításának lehetővé tételéhez vegyük fel a következő sort az `/etc/rc.conf` állományba:

```
samba_enable="YES"
```

Ha még finomabb irányításra vágunk:

```
nmbd_enable="YES"
```

```
smbd_enable="YES"
```

Megjegyzés: Ezzel egyben a rendszer indításakor automatikusan be is indítjuk a **Samba** szolgáltatást.

A **Samba** a következőkkel bármikor elindítható:

```
# /usr/local/etc/rc.d/samba start
Starting SAMBA: removing stale tdb's :
Starting nmbd.
Starting smbd.
```

Az `rc` szkriptekkel kapcsolatban a 11.7 Szakaszt ajánljuk elolvasásra.

A **Samba** jelen pillanatban három különálló démonból áll. Láthatjuk is, hogy az **nmbd** és **smbd** démonokat elindította a `samba` szkript. Ha az `smb.conf` állományban engedélyeztük a **winbind** névfeloldási szolgáltatást is, akkor láthatjuk, hogy ilyenkor a **winbindd** démon is elindul.

A **Samba** így állítható le akármikor:

```
# /usr/local/etc/rc.d/samba stop
```

A **Samba** egy összetett szoftvercsomag, amely a Microsoft Windows hálózatokkal kapcsolatos széles körű együttműködést tesz lehetővé. Az általa felkínált alapvető lehetőségeken túl a többi a <http://www.samba.org> honlapon ismerhetjük meg (angolul).

29.10. Az órák egyeztetése az NTP használatával

Készítette: Tom Hukins.

29.10.1. Áttekintés

Idővel a számítógép órája hajlamos elmászni. A hálózati idő protokoll (Network Time Protocol, NTP) az egyik módja az óránk pontos tartásának.

Rengeteg internetes szolgáltatás elvárja vagy éppen előnyben részesíti a számítógép órájának pontosságát. Például egy webszervertől megkérdezhetik, hogy egy állományt adott ideje módosítottak-e. A helyi hálózatban az egyazon állományszerveren megosztott állományok ellentmondásmentes dátumozása érdekében szinte elengedhetetlen az órák szinkronizálása. Az olyan szolgáltatások, mint a `cron(8)` is komolyan építkeznek a pontosan járó rendszeróra, amikor egy adott pillanatban kell lefuttatniuk parancsokat.

A FreeBSD alapból az `ntpd(8)` NTP szerveret tartalmazza, amellyel más NTP szerverek segítségével tudjuk beállítani gépünk óráját, vagy éppen idővel kapcsolatos információkat szolgáltatni másoknak.

29.10.2. A megfelelő NTP szerverek kiválasztása

Az óránk egyeztetéséhez egy vagy több NTP szerverre lesz szükségünk. Előfordulhat, hogy a hálózati rendszergazdánk vagy az internet-szolgáltatónk már beállított egy ilyen szervert erre a célra. Ezzel kapcsolatban olvassuk el a megfelelő leírásokat. A nyilvánosan elérhető NTP szerverekről készült egy lista (<http://ntp.isc.org/bin/view/Servers/WebHome>), ahonnan könnyedén ki tudjuk keresni a számunkra leginkább megfelelő (hozzánk legközelebbi) szervert. Ne hagyjuk figyelmen kívül a szerverre vonatkozó házirendet és kérjünk engedélyt a használatához, amennyiben ez szükséges.

Több, egymással közvetlen kapcsolatban nem álló NTP szerver választásával járunk jól, ha netalán az egyikük váratlanul elérhetetlenné vagy az órája pontatlanná válna. Az `ntpd(8)` a visszakapott válaszokat intelligensen használja fel, mivel esetükben a megbízható szervereket részesíti előnyben.

29.10.3. A gépünk beállítása

29.10.3.1. Alapvető beállítások

Ha a számítógépünk indításakor akarjuk egyeztetni az óránkat, akkor erre az `ntpd(8)` nevű programot használhatjuk. Ez olyan asztali gépek számára megfelelő választás, amelyeket gyakran indítanak újra és csak időnként kell szinkronizálnunk. A legtöbb gépnek viszont az `ntpd(8)` használatára van szüksége.

Az `ntpd(8)` elindítása olyan esetekben is hasznos, ahol az `ntpd(8)` is fut. Az `ntpd(8)` az órát fokozatosan állítja, ellenben az `ntpd(8)` az eltérés mértékétől és irányától függetlenül egyszerűen átállítja a gép óráját a pontos időre.

Az `ntpd(8)` elindítását úgy tudjuk engedélyezni a rendszer indításakor, ha az `/etc/rc.conf` állományba berakjuk az `ntpd_enable="YES"` sort. Emellett még `ntpd_flags` változóban meg kell adnunk az alkalmazott beállítások mellett azokat a szervereket, amelyekkel szinkronizálni akarunk.

29.10.3.2. Általános beállítások

Az NTP az `/etc/ntp.conf` állományon keresztül állítható, amelyek felépítését az `ntp.conf(5)` man oldal tárgyalja. Íme erre egy egyszerű példa:

```
server ntplocal.minta.com prefer
server timeserver.minta.org
server ntp2a.minta.net

driftfile /var/db/ntp.drift
```

A `server` beállítás adja meg az egyeztetéshez használt szervereket, soronként egyet. Ha egy szerver mellett szerepel még a `prefer` paraméter is, ahogy azt a példában a `ntplocal.minta.com` mellett láthattuk, akkor a többivel szemben azt a szerveret fogjuk előnyben részesíteni. Az így kiemelt szervertől érkező választ abban az esetben viszont eldobjuk, hogy a többi szervertől kapott válasz jelentős mértékben eltér tőle. Minden más esetben a ő válasza lesz a mérvadó. A `prefer` paramétert általában olyan NTP szerverekhez használják, amelyek közismerten nagy pontosságúak, tehát például külön erre a célra szánt felügyeleti eszközt is tartalmaznak.

A `driftfile` beállítással azt az állományt adjuk meg, amiben a rendszeróra frekvencia eltolódásait tároljuk. Az `ntpd(8)` program ezzel ellensúlyozza automatikusan az óra természetes elmozdítását, ezáltal lehetővé téve, hogy egy viszonylag pontos időt kapjunk még abban az esetben is, amikor egy kis időre külső időforrások nélkül maradnánk.

A `driftfile` beállítással egyben azt az állományt jelöljük ki, amely az NTP szervertől kapott korábbi válaszokat tárolja. Ez az NTP működéséhez szükséges belső adatokat tartalmaz, ezért semmilyen más programnak nem szabad módosítania.

29.10.3.3. A szerverünk elérésének szabályozása

Alapértelmezés szerint az NTP szerverünket bárki képes elérni az interneten. Az `/etc/ntp.conf` állományban szereplő `restrict` beállítás segítségével azonban meg tudjuk mondani, milyen gépek érhetik el a szerverünket.

Ha az NTP szerverünk felé mindenféle próbálkozást el akarunk utasítani, akkor az `/etc/ntp.conf` állományba a következő sort kell felvennünk:

```
restrict default ignore
```

Megjegyzés: Ezzel egyben azonban a helyi beállításainkban szereplő szerverek elérését is megakadályozzuk. Ha külső NTP szerverekkel is szeretnénk szinkronizálni, akkor itt is engedélyezünk kell ezeket. Erről bővebben lásd az `ntp.conf(5)` man oldalon.

Ha csak a belső hálózatunkban levő gépek számára szeretnénk elérhetővé tenni az órák egyeztetését, de sem a szerver állapotának módosítását nem engedélyezzük, sem pedig azt, hogy a vele egyenrangú szerverekkel szinkronizáljon, akkor az iménti helyett a

```
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

sort írjuk bele, ahol a `192.168.1.0` a belső hálózatunk IP-címe és a `255.255.255.0` a hozzá tartozó hálózati maszk.

Az `/etc/ntp.conf` több `restrict` típusú beállítást is tartalmazhat. Ennek részleteiről az `ntp.conf(5)` man oldalon, az `Access Control Support` című szakaszban olvashatunk.

29.10.4. Az NTP futtatása

Úgy tudjuk az NTP szervert elindítani a rendszerünkkel együtt, ha az `/etc/rc.conf` állományban szerepeltetjük az `ntpd_enable="YES"` sort. Ha az `ntpd(8)` számára további beállításokat is át akarunk adni, akkor az `/etc/rc.conf` állományban adjuk meg az `ntpd_flags` paramétert.

Ha a gépünk újraindítása nélkül akarjuk elindítani a szerver, akkor az `ntpd` parancsot adjuk ki az `/etc/rc.conf` állományban a `ntpd_flags` változóhoz megadott paraméterekkel. Mint például:

```
# ntpd -p /var/run/ntpd.pid
```

29.10.5. Az ntpd használati időleges internet csatlakozással

Az `ntpd(8)` program megfelelő működéséhez nem szükséges állandó internet kapcsolat. Ha azonban igény szerinti tárcsázással építünk fel ideiglenes kapcsolatot, akkor érdemes letiltani az NTP forgalmát, nehogy feleslegesen aktiválja vagy tartsa életben a vonalat. Ha PPP típusú kapcsolatunk van, akkor az `/etc/ppp/ppp.conf` állományban a `filter` direktívával tudjuk ezt leszabályozni. Például:

```
set filter dial 0 deny udp src eq 123
# Nem engedjük az NTP által küldött adatoknak, hogy tárcsázást
# kezdeményezzenek:
set filter dial 1 permit 0 0
set filter alive 0 deny udp src eq 123
# Nem engedjük az NTP adatainak, hogy fenntartsák a kapcsolatot:
set filter alive 1 deny udp dst eq 123
set filter alive 2 permit 0/0 0/0
```

Mindenezekről részletesebb felvilágosítást a `ppp(8)` man oldal `PACKET FILTERING` című szakaszában és a `/usr/share/examples/ppp/` könyvtárban található példákban kaphatunk.

Megjegyzés: Egyes internet-szolgáltatók blokkolják az alacsonyabb portokat, ezáltal az NTP nem használható, mivel a válaszok nem fogják elérni a gépünket.

29.10.6. További olvasnivalók

Az NTP szerver dokumentációja HTML formátumban a `/usr/share/doc/ntp/` könyvtárban található.

29.11. Távoli gépek naplózása `syslogd` használatával

Készítette: Tom Rhodes.

A rendszernaplókkal kapcsolatos műveletek egyaránt fontosak a biztonság és a karbantartás szempontjából. Ha közepes vagy nagyobb méretű, esetleg különböző típusú hálózatokban adminisztrálunk több gépet, akkor könnyen átláthatatlanná válhat a naplók rendszeres felügyelete. Ilyen helyzetekben a távoli naplózás beállításával az egész folyamatot sokkal kényelmesebbé tehetjük.

Némileg képesek vagyunk enyhíteni a naplóállományok kezelésének terhet, ha egyetlen központi szerverre küldjük át az adatokat. Ekkor a FreeBSD alaprendszerében megtalálható alapeszközökkel, mint például a `syslogd(8)` vagy a `newsyslog(8)` felhasználásával egyetlen helyen be tudjuk állítani a naplók összegyűjtését, összefésülését és cseréjét. A most következő példa konfigurációban az A gép, a `naploszerver.minta.com` fogja gyűjteni a helyi hálózatról érkező naplóinformációkat. A B gép, a `naplokliens.minta.com` pedig a szervernek küldi a naplózandó adatokat. Éles környezetben mind a két gépnek rendelkeznie kell megfelelő DNS bejegyzésekkel, vagy legalább szerepelniük kell egymás `/etc/hosts` állományaiban. Ha ezt elmulasztjuk, a szerver nem lesz hajlandó adatokat fogadni.

29.11.1. A naplószerver beállítása

A naplószerverek olyan gépek, amelyeket úgy állítottunk be, hogy naplózási információkat tudjanak fogadni távoli számítógépekről. A legtöbb esetben így egyszerűsíteni tudunk a konfiguráción, vagy olykor egyszerűen csak hasznos, ha ezt a megoldást alkalmazzuk. Függetlenül attól, hogy miért használjuk, a továbblépés előtt néhány előkészületet meg kell tennünk.

Egy rendesen beállított naplószervernek legalább a következő követelményeknek kell eleget tennie:

- az 514-es UDP portot engedélyezni kell mind a kliensen, mind pedig a szerveren futó tűzfal szabályrendszerében;
- a `syslogd(8)` képes legyen a távoli kliens gépekről érkező üzeneteket fogadni;
- a `syslogd(8)` szervernek és az összes kliensnek rendelkeznie kell érvényes DNS (közvetlen és inverz) bejegyzésekkel vagy szerepelnie kell az `/etc/hosts` állományban.

A naplószerver beállításához mindegyik klienst fel kell vennünk az `/etc/syslog.conf` állományba, valamint meg kell adnunk a megfelelő funkciót (facility):

```
+naplokliens.minta.com
*. *      /var/log/naplokliens.log
```

Megjegyzés: A `syslog.conf(5)` man oldalán megtalálhatjuk a különböző támogatott és elérhető *funkciókat*.

Miután beállítottuk, az összes adott funkcióhoz tartozó üzenet az előbb megadott állományba (`/var/log/naplokliens.log`) fog kerülni.

A szerveren továbbá meg kell adnunk a következő sort az `/etc/rc.conf` állományban:

```
syslogd_enable="YES"
syslogd_flags="-a naplokliens.minta.com -vv"
```

Az első sorral engedélyezzük a `syslogd` elindítását a rendszerindítás során, majd a második sorral engedélyezzük, hogy a kliens naplózni tudjon a szerverre. Itt még látható a `-vv` opció, amellyel a naplózott üzenetek részletességét tudjuk növelni. Ennek nagyon fontos a szerepe a naplózási funkciók behangolásakor, mivel így a rendszergazdák pontosan láthatják milyen típusú üzenetek milyen funkcióval kerültek rögzítésre a naplóban.

Befejezésképpen hozzuk létre a naplóállományt. Teljesen mindegy, hogy erre milyen megoldást alkalmazunk, például a `touch(1)` remekül megfelel:

```
# touch /var/log/naplokliens.log
```

Ezután indítsuk újra és ellenőrizzük a `syslogd` démon:

```
# /etc/rc.d/syslogd restart
# pgrep syslog
```

Ha válaszul megkapjuk a futó démon azonosítóját, akkor sikerült újraindítanunk, elkezdhetjük a kliens beállítását. Ha valamiért nem indult volna újra a szerver, az `/var/log/messages` állományból próbáljuk meg kideríteni az okát.

29.11.2. A naplókliens beállítása

A naplókliens az a gép, amely egy helyi naplópéldány karbantartása mellett továbbküldni a naplózandó információkat egy naplószervernek.

Hasonlóan a naplószerverekhez, a klienseknek is teljesítenie bizonyos alapvető elvárásokat:

- a `syslogd(8)` démon küldjön bizonyos típusú üzeneteket a naplószervernek, amely ezeket pedig képes legyen fogadni;
- a hozzá tartozó tűzfal engedje át a forgalmat az 514-es UDP porton;
- rendelkezzen mind közvetlen, mind pedig inverz DNS bejegyzéssel, vagy szerepeljenek az `/etc/hosts` állományban.

A kliens beállítása sokkal egyszerűbb a szerverhez képest. A kliensen adjuk hozzá a következő sorokat az `/etc/rc.conf` állományhoz:

```
syslogd_enabled="YES"
syslogd_flags="-s -vv"
```

A szerver beállításaihoz hasonlóan itt is engedélyezzük a `syslogd` démon és megnöveljük a naplózott üzenetek részletességét. A `-s` kapcsolóval pedig megakadályozzuk, hogy a kliens más gépekről is hajlandó legyen naplőüzeneteket elfogadni.

A funkciók a rendszernek azon részét írják le, amelyhez létrejön az adott üzenet. Tehát például az `ftp` és `ipfw` egyaránt ilyen funkciók. Amikor keletkezik egy naplőüzenet valamelyikükhöz, általában megjelenik a nevük. A funkciókhoz tartozik még egy prioritás vagy szint is, amellyel az adott üzenet fontosságát jelzik. Ezek közül a

leggyakoribb a warning (mint “figyelmeztetés”) és info (mint “információ”). A használható funkciók és a hozzájuk tartozó prioritások teljes listáját a syslog(3) man oldalán olvashatjuk.

A naplószervert meg kell adnunk a kliens `/etc/syslog.conf` állományában. Itt a @ szimbólummal jelezzük, hogy az adatokat egy távoli szerverre szeretnénk továbbküldeni, valahogy így:

```
*.* @naploszerver.minta.com
```

Ezután a beállítás érvényesítéséhez újra kell indítanunk a syslogd démonot:

```
# /etc/rc.d/syslogd restart
```

A logger(1) használatával próbáljuk ki a kliensről a aplóüzenetek hálózaton keresztüli küldését, és küldjünk valamit a syslogd démonnak:

```
# logger "Üdvözlét a naplokliensrol"
```

A parancs kiadása után az üzenetnek mind a kliens, mind pedig a szerver `/var/log/messages` állományában meg kell jelennie.

29.11.3. Hibakeresés

Előfordulhat, hogy a naplószerver valamiért nem kapja meg rendesen az üzeneteket, ezért valamilyen módon meg kell keresnünk a hiba okát. Ez több minden lehet, de általában két leggyakoribb ok valamilyen hálózati kapcsolódási vagy DNS beállítási hiba. Ezek teszteléséhez gondoskodjunk róla, hogy a gépek kölcsönösen elérhetőek egymásról az `/etc/rc.conf` állományban megadott hálózati nevük szerint. Ha ezzel látszólag minden rendben van, akkor próbáljuk meg módosítani a `syslogd_flags` értékét az `/etc/rc.conf` állományban.

A most következő példában a `/var/log/naplokliens.log` teljesen üres, illetve a `/var/log/messages` állomány semmilyen hibára utaló okot nem tartalmaz. A hibakereséshez még több információt a `syslogd_flags` átírásával tudunk kérni:

```
syslogd_flags="-d -a naplokliens.minta.com -vv"
```

Természetesen ne felejtjük el újraindítani a szervert:

```
# /etc/rc.d/syslogd restart
```

A démon újraindítása után közvetlenül az alábbiakhoz hasonló üzenetek árasztják el a képernyőt:

```
logmsg: pri 56, flags 4, from naploszerver.minta.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from naploszerver.minta.com, msg syslogd: kernel boot file is /boot/kernel
Logging to FILE /var/log/messages
syslogd: kernel boot file is /boot/kernel/kernel
cvthname(192.168.1.10)
validate: dgram from IP 192.168.1.10, port 514, name naplokliens.minta.com;
rejected in rule 0 due to name mismatch.
```

A diagnosztikai üzeneteket végigolvasva nyilvánvaló válik, hogy azért dobja el az üzeneteket a szerver, mert nem megfelelő a gép neve. Miután átnézzük a beállításainkat, felfedezhetünk az `/etc/rc.conf` állományban egy apró hibát:

```
syslogd_flags="-d -a naploklien.minta.com -vv"
```

Láthatjuk, hogy ebben a sorban a naplokliens névnek kellene szerepelni, nem pedig a naploklien névnek. Miután elvégeztük a szükséges javításokat, indítsuk újra a szerveret és vizsgáljuk meg az eredményt:

```
# /etc/rc.d/syslogd restart
logmsg: pri 56, flags 4, from naploszerver.minta.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from naploszerver.minta.com, msg syslogd: kernel boot file is /boot/kerne
syslogd: kernel boot file is /boot/kernel/kernel
logmsg: pri 166, flags 17, from naploszerver.minta.com, msg Dec 10 20:55:02 <syslog.err> naplosze
cvthname(192.168.1.10)
validate: dgram from IP 192.168.1.10, port 514, name naplokliens.minta.com;
accepted in rule 0.
logmsg: pri 15, flags 0, from naplokliens.minta.com, msg Dec 11 02:01:28 pgj: Masodik teszt uzene
Logging to FILE /var/log/naplokliens.log
Logging to FILE /var/log/messages
```

Itt már minden üzenet rendben megérkezett és a megfelelő állományokba került (a /var/log/messages a kliensen, és a /var/log/naplokliens.log a szerveren)).

29.11.4. Biztonsági megfontolások

Mint minden hálózati szolgáltatás esetén, ilyenkor is figyelembe kell vennünk bizonyos biztonsági megfontolásokat a tényleges konfiguráció kiépítése előtt. Olykor előfordulhat, hogy a naplók különböző kényes információkat tartalmaznak, mint például a helyi rendszeren futó szolgáltatások nevei, felhasználói nevek vagy egyéb konfigurációs adatok. A kliens és a szerver között hálózaton utazó adatok viszont se nem titkosítottak, se nem jelszóval védettek. Ha titkosítást szeretnénk használni, akkor javasoljuk például a `security/stunnel` portot, amellyel egy titkosított tunnelel keresztül tudunk adatokat küldeni a hálózaton.

A helyi rendszer biztonságának szavatolása is fontos lehet. A naplók sem a használat során, sem pedig a lecserélésük után nem kerülnek titkosításra. Emiatt a helyi rendszerhez hozzáférő felhasználók kedvükre nyerhetnek ki belőlük a rendszerünket érintő konfigurációs információkat. Ezért ilyenkor nagyon fontos, hogy mindig a megfelelő engedélyeket állítsuk be a naplókra. A `newsyslog(8)` segédprogrammal be tudjuk állítani a frissen létrehozott és a lecserélt naplók engedélyeit. Tehát könnyen megakadályozhatjuk a helyi felhasználók kíváncsiságát, ha itt a naplók engedélyeit például a 600 kóddal adjuk meg.

Fejezet 30. Tűzfalak

Írta: Joseph J. Barbish. SGML formátumúra alakította és aktualizálta: Brad Davis.

30.1. Bevezetés

A tűzfalakkal a rendszerünkön keresztülfolyó bejövő és kimenő forgalmat tudjuk szűrni. A tűzfalak egy vagy több “szabályrendszer” alapján vizsgálják az éppen érkező vagy távozó hálózati csomagokat, és vagy továbbengedik ezeket vagy megállítják. A tűzfalak szabályai a csomagok egy vagy több jellemzőjét veszik szemügyre, amelyek lehetnek például a protokoll típusa, a forrás vagy cél hálózati címe, esetleg a forrás- vagy a célport.

A tűzfalak jelentős mértékben képesek gyarapítani egy gép vagy egy hálózat védelmét. Leginkább a következőkre tudjuk felhasználni:

- A belső hálózatunkban futó alkalmazások, szolgáltatások, gépek megvédésére és elszigetelésére az internetről érkező nem kívánt forgalom ellen
- A belső hálózatban levő gépek elérését tudjuk korlátozni vagy letiltani az interneten elérhető szolgáltatások felé
- A hálózati címfordítás (Network Address Translation, NAT) beállításához, ahol a belső hálózatunk privát IP-címeket használnak és egy közös kapcsolaton keresztül érik el az internetet (egyetlen IP-címmel, vagy pedig automatikusan kiosztott publikus címekkel).

A fejezet elolvasása során megismerjük:

- hogyan adjuk meg helyesen a csomagok szűrését leíró szabályokat;
- a FreeBSD-be épített tűzfalak közti különbségeket;
- hogyan állítsuk be és használjuk az OpenBSD **PF** tűzfalát;
- hogyan állítsuk be és használjuk az **IPFILTER** tűzfalat;
- hogyan állítsuk be és használjuk az **IPFW** tűzfalat.

A fejezet elolvasása előtt ajánlott:

- a FreeBSD-hez és az internethez kötődő alapvető fogalmak ismerete.

30.2. Röviden a tűzfalokról

A tűzfalak szabályrendszerait alapvetően kétféleképpen tudjuk összeállítani: “inkluzív”, vagyis megengedő, illetve “exkluzív” vagyis kizáró módon. Az exkluzív tűzfalak minden forgalmat átengednek, amiről nem rendelkeznek a tűzfal szabályai. Az inkluzív tűzfalak ennek pontosan az ellenkezőjét teszik. Csak azt a forgalmat engedik át, amiről van szabály és minden mást blokkolnak.

Az inkluzív tűzfalak alkalmazásával sokkal jobban kezünkbe tudjuk tartani a hálózatunk kimenő forgalmát, ezért leginkább az internetes szolgáltatásokat futtató rendszerek esetében bizonyulhat jobb választásnak. Emellett az internetről a hálózatunk felé irányuló forgalmat is képes szabályozni. Ekkor az egyetlen szabályra sem illeszkedő csomagokat egyszerűen eldobjuk és naplózzuk. Az inkluzív tűzfalak általában biztonságosabbak az exkluzív típusú társaiknál, mivel esetükben jelentős mértékben visszaszorul a nem kívánatos átfolyó forgalom.

Megjegyzés: Hacsak nem emeljük ki külön, a fejezet további részében minden példaként megadott szabályrendszer inkluzív tűzfalat hoz létre.

Ez a típusú védelem még tovább fokozható az “állapottartó tűzfalak” (stateful firewall) használatával. Az ilyen típusú tűzfalak szemmel tartják a rajtuk keresztül megnyitott kapcsolatokat, és vagy csak a már meglévő kapcsolathoz tartozó forgalmat engedik át vagy nyitnak egy újat. Az állapottartó tűzfalak hátránya, hogy a “Denial of Service” (DoS) típusú támadásokkal szemben sokkal sérülékenyebbek olyan helyzetekben, amikor az új kapcsolatok nagyon gyorsan jönnek létre. A legtöbb tűzfal esetében azonban tudjuk vegyíteni az állapottartó és nem állapottartó viselkedést, és ezzel egy ideális beállítást kialakítani.

30.3. Tűzfalak

A FreeBSD alaprendszerébe három különböző tűzfalat építettek be, melyek a következők: az *IPFILTER* (másik nevén IPF), az *IPFIREWALL* (más néven IPFW) és az *OpenBSD csomagszűrője* (Packet Filter, azaz PF). A forgalom szabályozására (vagyis alapvetően a sávszélesség kihasználtságának vezérlésére) a FreeBSD két beépített csomagot tartalmaz: ez az *altq(4)* és a *dummynet(4)*. Általában a *Dummynet* az IPFW, míg az *ALTQ* a PF partnere. Az *IPFILTER* esetében maga az *IPFILTER* végzi a címfordítást és a szűrést, a sávszélességet pedig az IPFW a *dummynet(4)* vagy a PF az *ALTQ* segítségével. Az IPFW és a PF szabályokkal rendelkezik a rendszerünkbe érkező vagy onnan távozó csomagokról, habár megoldásaik teljesen máshogy működnek és a szabályok megadási módja is eltér.

A FreeBSD azért tartalmaz egyszerre ennyiféle tűzfalat, mert az emberek elvárásai és igényei eltérnek. Egyikük sem tekinthető a legjobbnak.

A szerző egyébként az *IPFILTER* megoldását részesíti előnyben, mivel egy hálózati címfordítást alkalmazó környezetben sokkal könnyebb vele megfogalmazni az állapottartó szabályokat, valamint tartalmaz egy beépített FTP proxyt is, amivel így a kimenő FTP kapcsolatok beállítása még tovább egyszerűsödik.

Mivel az összes tűzfal a csomagok fejlcének bizonyos mezőinek alapján dolgozik, ezért a tűzfal szabályrendszerét megalkotó egyénnek teljesen tisztában kell lennie a TCP/IP működésével, továbbá azzal, hogy ezekben a mezőkben milyen értékek szerepelhetnek és ezeket hogyan használják egy átlagos kapcsolat alatt. Ebben a témában a <http://www.ipprimer.com/overview.cfm> címen találhatunk egy remek ismertetőt (angolul).

30.4. Az OpenBSD csomagszűrője (PF) és az ALTQ

Átnézte és aktualizálta: John Ferrell.

2003 júliusában az OpenBSD PF néven ismert csomagszűrőjét átirták FreeBSD-re és elérhetővé tették a FreeBSD Portgyűjteményének részeként. A PF programot beépítetten tartalmazó első kiadás pedig 2004 novemberében a FreeBSD 5.3 volt. A PF egy teljes, mindentudó tűzfal, amely támogatja az ún. ALTQ (Alternate Queuing, vagyis a “váltóbesorolás”) megoldást. Az ALTQ lehetővé teszi a sávszélesség korlátozását a szolgáltatás minősége (Quality of Service, QoS) alapján.

Az OpenBSD Projekt kiváló munkát végez a PF felhasználói útmutatójának (<http://www.openbsd.org/faq/pf/>) karbantartásával. A kézikönyv ezen szakasza ezért elsősorban azzal foglalkozik, hogyan kell a PF-et FreeBSD alatt használni, miközben igyekszik egy általános összefoglalást adni a témáról. A részletesebb információkkal kapcsolatban azonban feltétlenül nézzük meg a felhasználói útmutatót.

A <http://pf4freebsd.love2party.net/> címen olvashatunk többet arról (angolul), hogy a PF-et hogyan használjunk FreeBSD-n.

30.4.1. A PF rendszermagmodulok használata

A PF modul betöltéséhez a következő sort kell felvennünk az `/etc/rc.conf` állományba:

```
pf_enable="YES"
```

Ezt követően futtassuk le a hozzá tartozó rendszerindító szkriptet:

```
# /etc/rc.d/pf start
```

A PF modul abban az esetben nem fog betöltődni, ha nem találja a szabályokat tartalmazó konfigurációs állományt. Ez alapértelmezés szerint az `/etc/pf.conf` állomány. Ha a szabályok leírása rendszerünkön máshol található, akkor az `/etc/rc.conf` állományban a következő módon adhatjuk meg annak pontos helyét:

```
pf_rules="/elérési/út/pf.conf"
```

Megjegyzés: A FreeBSD 7.0 kiadással a minta `pf.conf` állomány az `/etc` könyvtárból átkerült a `/usr/share/examples/pf` könyvtárba. A FreeBSD 7.0 előtti kiadásokban alapértelmezés szerint található egy `pf.conf` állomány az `/etc` könyvtárban.

A PF modul parancssorból akár kézzel is betölthető:

```
# kldload pf.ko
```

A PF működésének naplózását a `pfllog.ko` teszi lehetővé, amelyet az alábbi sor hozzáadásával engedélyezhetünk az `/etc/rc.conf` állományban:

```
pfllog_enable="YES"
```

A modul betöltését a hozzá tartozó rendszerindító szkript segítségével kérhetjük:

```
# /etc/rc.d/pfllog start
```

Ha a PF többi funkcióját is használni szeretnénk, akkor ehhez egy új rendszermagot kell fordítanunk PF támogatással.

30.4.2. A PF rendszermagbeli beállításai

Noha egyáltalán nem szükséges beépítenünk a PF támogatását a rendszermagba, abban az esetben mégis szükségünk lehet rá, amikor a PF olyan komolyabb lehetőségeit szeretnénk kiaknázni, amelyek már nem részei a modulnak. Ilyen például a `pfsync(4)`, amely a PF által használt állapotábrázatok bizonyos változásainak megjelenítésére alkalmas pszeudoeszköz. A `carp(4)` megoldásával párosítva így akár hibatűrő tűzfalak is kialakíthatóak a PF-fel. A CARP megoldásáról a kézikönyvben bővebb ismertetést a 31.13 Szakasz ad.

A PF rendszermag konfigurációs beállításai a `/usr/src/sys/conf/NOTES` állományban találhatóak:

```
device pf
```

```
device pflog
device pfsync
```

A `device pf` beállítás engedélyezi a csomagszűrő tűzfalat (`pf(4)`).

A `device pflog` megadásával keletkezik egy `pflog(4)` pszeudo hálózati eszköz, amellyel egy `bpf(4)` eszközre érkező forgalmat tudunk naplózni. Ezután a `pflogd(8)` démon használható tőle származó naplózott adatok rögzítésére.

A `device pfsync` engedélyezi a `pfsync(4)` pszeudo hálózati eszköz létrejöttét, amely az ún. "állapotváltások" megfigyelésére alkalmas.

30.4.3. Az `rc.conf` állományban elérhető beállítások

A következő `rc.conf(5)` beállítások aktiválják a rendszerindítás során a PF és a `pflog(4)` használatát:

```
pf_enable="YES"           # a PF engedélyezése (a modul betöltése, ha kell)
pf_rules="/etc/pf.conf"   # a pf szabályait tartalmazó állomány
pf_flags=""               # a pfctl indításához szükséges további paraméterek
pflog_enable="YES"        # a pflogd(8) elindítása
pflog_logfile="/var/log/pflog" # hol tartsa a pflogd az naplóit
pflog_flags=""            # a pflogd indításához szükséges paraméterek
```

Ha a tűzfalunk mögött egy helyi hálózat is meghúzódik, akkor az ott levő gépek számára valamilyen módon tudnunk kell továbbítani a csomagokat vagy címfordítást kell végezni, így ez is mindenképpen kelleni fog:

```
gateway_enable="YES"      # az átjáró funkciók engedélyezése
```

30.4.4. A szűrési szabályok megfogalmazása

A PF a beállításait a `pf.conf(5)` állomány tárolja (amely alapértelmezés szerint az `/etc/pf.conf` helyen található), és az ebben található szabályok alapján módosítja, dobja el vagy éppen engedi át a csomagokat. A FreeBSD rendszerünkben ehhez található néhány példát a `/usr/share/examples/pf/` könyvtárban. A PF által használt szabályokról minden részletre kiterjedően a PF felhasználói útmutatójában (<http://www.openbsd.org/faq/pf/>) olvashatunk.

Figyelem A PF felhasználói útmutatójának (<http://www.openbsd.org/faq/pf/>) olvasásakor ne feledkezzünk meg róla, hogy a különböző FreeBSD verziók különböző PF verziókat tartalmaznak. A FreeBSD 7.x és későbbi változatok az OpenBSD 4.1 kiadásában szereplő PF változatot tartalmazzák.

A FreeBSD packet filter levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf>) remek hely a PF tűzfal beállításával és futtatásával kapcsolatos kérdésekre. A kérdés előtt azonban ne felejtjük el alaposan átnézni az archívumot!

30.4.5. A PF használata

A PF a `pfctl(8)` segítségével vezérelhető. Az alábbiakban ezzel kapcsolatban most összefoglalunk néhány hasznos parancsot (de ne felejtjük el megnézni a `pfctl(8)` man oldalon található többi lehetőséget sem):

Parancs	Leírás
<code>pfctl -e</code>	A PF engedélyezése
<code>pfctl -d</code>	A PF tiltása
<code>pfctl -F all -f /etc/pf.conf</code>	Az összes (címfordítási, szűrési, állapottartási stb.) szabály törlése, és az <code>/etc/pf.conf</code> állomány újratöltése
<code>pfctl -s [rules nat state]</code>	A szűrési (rules), címfordítási (nat) és állapottartási (state) információk lekérdezése
<code>pfctl -vnf /etc/pf.conf</code>	Az <code>/etc/pf.conf</code> állomány ellenőrzése a benne levő szabályok betöltése nélkül

30.4.6. Az ALTQ engedélyezése

Az ALTQ kizárólag csak úgy használható, ha a konfigurációs beállításokon keresztül beépítjük a FreeBSD rendszermagjába. Az ALTQ alkalmazását nem minden hálózati kártya meghajtója támogatja, ezért ezt a `altq(4)` man oldalon ellenőrizzük.

A következő rendszermag konfigurációs beállításokkal engedélyezhetjük az ALTQ használatát és bővíthetjük azt további lehetőségekkel:

```
options      ALTQ
options      ALTQ_CBQ      # osztályozás alapú besorolás (Class Bases Queuing, CBQ)
options      ALTQ_RED      # véletlen korai észlelés (Random Early Detection, RED)
options      ALTQ_RIO      # RED befele/kifele
options      ALTQ_HFSC     # hierarchikus csomagütemező (Hierarchical Packet Scheduler, HFSC)
options      ALTQ_PRIQ     # prioritásos besorolás (Priority Queuing, PRIQ)
options      ALTQ_NOPCC    # az SMP esetén kell
```

Az `options ALTQ` az ALTQ rendszert engedélyezi.

Az `options ALTQ_CBQ` engedélyezi a osztályozás alapú besorolást (*Class Based Queuing*, CBQ). A CBQ használatával a kapcsolatunkhoz tartozó sávszélességet különböző osztályokra vagy sorokra tudjuk bontani és a szűrési szabályoknak megfelelően osztályozni segítségükkel a forgalmat.

Az `options ALTQ_RED` a véletlen korai észlelés (*Random Early Detection*, RED) használatát engedélyezi. A RED a hálózati forgalomban keletkező torlódások elkerülésére alkalmas. A RED ezt a problémát úgy oldja meg, hogy méri a sorok hosszát és összeveti a hozzá tartozó minimális és maximális küszöbértékekkel. Ha a sor hossza meghaladja a számára előírt maximális értéket, akkor az új csomagokat eldobja. Nevéhez hűen a RED az eldobásra ítélt csomagokat véletlenszerűen választja ki.

Az `options ALTQ_RIO` engedélyezi a RED használatát mind a két irányba, tehát be- és kifelé.

Az `options ALTQ_HFSC` a pártatlan hierachikus szolgáltatási görbe alapú csomagütemezőt (*Hierarchical Fair Service Curve Packet Scheduler*, HFSC) engedélyezi. Vele kapcsolatban a <http://www-2.cs.cmu.edu/~hzhang/HFSC/main.html> címen találhatunk bővebben olvasnivalót (angolul).

Az `options ALTQ_PRIQ` a prioritásos besorolást (*Priority Queuing*, PRIQ) teszi elérhetővé. A PRIQ mindig elsőként a nagyobb értékű sorban levő forgalmat továbbítja.

Az `options ALTQ_NOPCC` az ALTQ SMP, vagyis többprocesszoros támogatását adja meg. Ilyen típusú rendszerekben ez kötelező.

30.5. Az IPFILTER (IPF) tűzfal

Az IPFILTER szerzője Darren Reed. Az IPFILTER nem kötődik egyik rendszerhez sem: ez egy olyan nyílt forráskódú alkalmazás, amelyet átírtak FreeBSD, NetBSD, OpenBSD, SunOS, HP/UX és Solaris operációs rendszerekre. Az IPFILTER karbantartása és támogatása pillanatnyilag is aktív, folyamatosan jelennek meg újabb változatai.

Az IPFILTER egy rendszermag oldalán működő tűzfalazási és egy címfordítási mechanizmusra alapszik, amelyet felhasználói programokkal tudunk felügyelni és vezérelni. A tűzfal szabályai az ipf(8) segédprogrammal állíthatók be vagy törölhetők. A hálózati címfordításra vonatkozó szabályokat az ipnat(1) segédprogrammal állíthatjuk be vagy törölhetjük. Az ipfstat(8) segédprogram képes futás közben statisztikákat készíteni az IPFILTER rendszermagban elhelyezkedő részeinek viselkedéséről. Az ipmon(8) program pedig az IPFILTER cselekvéseit képes a rendszernaplókba feljegyezni.

Az IPF eredetileg olyan szabályfeldolgozási módszer szerint készült, amelyben “az utolsó egyező szabály nyer” és csak állapot nélküli szabályokat ismert. Az idő múlásával az IPF részévé vált a “quick” opció és a “keep state” opción keresztül az állapot tartás is, melyek drámai mértékben korszerűsítették a szabályok feldolgozásának elvét. Az IPF hivatalos dokumentációja csak a régi szabályok létrehozását és azok feldolgozásának leírását tartalmazza. A korszerűsített funkciók csak kiegészítésként jelennek meg, és az általuk felkínált előnyök megértése egy sokkal magasabb szintű és biztonságosabb tűzfal megépítését teszi lehetővé.

A szakaszban szereplő utasításokban olyan szabályok szerepelnek, amelyek kihasználják a “quick” és “keep state” opciókat. Ezek az inkluzív tűzfalszabályok létrehozásának alapjai.

A régi típusú szabályokról a http://www.obfuscation.org/ipf/ipf-howto.html#TOC_1 és <http://coombs.anu.edu.au/~avalon/ip-filter.html> címen olvashatunk (angolul).

Az IPF gyakran ismételt kérdései a <http://www.phildev.net/ipf/index.html> címen érhetőek el (angolul).

A nyílt forrású IPFILTER levelezési lista kereshető archívumait a <http://marc.theaimsgroup.com/?l=ipfilter> címen találjuk (angolul).

30.5.1. Az IPF engedélyezése

Az IPF megtalálható a FreeBSD alaptelepítésében mint menet közben külön betölthető modul. Ha az `rc.conf` állományba beírjuk a `ipfilter_enable="YES"` sort, akkor ez a modul dinamikusan betöltődik. A betölthető modul alpból naplóz és a `default pass all` beállítást tartalmazza. Ha helyette a `block all` szabályt akarjuk használni, akkor emiatt még nem kell feltétlenül újrafordítanunk a FreeBSD rendszermagját, elég ha egyszerűen csak a szabályrendszerünk végére besúrjuk.

30.5.2. A rendszermag beállításai

Az IPF használatához nem kötelező a következő beállításokkal újrafordítani a FreeBSD rendszermagját, itt csupán háttérinformációként szerepel. Amikor az IPF a rendszermagba kerül, a betölthető modulra nem lesz szükség.

Az IPF a rendszermag forrásai között található `/usr/src/sys/conf/NOTES` állományban megadott beállításai a következő módon foglalhatóak össze:

```
options IPFILTER
options IPFILTER_LOG
options IPFILTER_DEFAULT_BLOCK
```

Az `options IPFILTER` engedélyezi az "IPFILTER" tűzfal támogatását.

Az `options IPFILTER_LOG` hatására az IPF az `ipl` csomagnaplózó pszeudo eszközre jegyzi fel a forgalmat — minden olyan szabály esetén, ahol megjelenik a `log` kulcsszó.

Az `options IPFILTER_DEFAULT_BLOCK` megváltoztatja az alapértelmezett viselkedést, tehát minden olyan csomag, amely nem illeszkedik a tűzfal valamelyik `pass` típusú (átengedő) szabályára, blokkolásra kerül.

Ezek a beállítások csak azt követően érvényesülnek, ha fordítottunk és telepítettünk velük egy új rendszermagot.

30.5.3. Az `rc.conf` állomány beállításai

Az `/etc/rc.conf` állományban a következő utasításokra lesz szükségünk az IPF működésbe hozására a rendszer indítása során:

```
ipfilter_enable="YES"           # az ipf tűzfal indítása
ipfilter_rules="/etc/ipf.rules" # betölti a szabályokat tartalmazó szöveges állományt
ipmon_enable="YES"              # elindítja az IP monitor naplózását
ipmon_flags="-Ds"               # D = indítás démonként
                                # s = naplózás a syslog használatával
                                # v = a tcp ablak, ack, seq csomagok naplózása
                                # n = az IP-címek és portok feloldása
```

Ha olyan helyi hálózat áll meg a tűzfal mögött, amely egy fenntartott privát IP-címtartományt használ, akkor még a következő utasításokra is szükségünk lesz a címfordítás bekapcsolásához:

```
gateway_enable="YES"           # a helyi hálózat átjárója
ipnat_enable="YES"             # az ipnat funkció elindítása
ipnat_rules="/etc/ipnat.rules" # az ipnat működéséhez szükséges definíciók
```

30.5.4. IPF

Az `ipf(8)` parancs használható a szabályokat tartalmazó állomány betöltésére. Általában egy állományba írjuk össze a tűzfal szabályait és ezzel a parancssal cseréljük le egyszerre a tűzfalban levő jelenlegi szabályokat:

```
# ipf -Fa -f /etc/ipf.rules
```

Az `-Fa` az összes belső szabály törlését jelenti.

Az `-f` jelzi, hogy egy állományból kell beolvasni a betöltendő szabályokat.

Ezzel mintegy lehetőségünk van változtatni a korábban összeállított szabályainkon, futtatni a fenti IPF parancsot és ezen keresztül úgy frissíteni a szabályok friss másolatával a már működő tűzfalat, hogy nem is kell újraindítanunk a rendszert. Ez a módszer igen kényelmes az új szabályok kipróbálásához, mivel bármikor tetszőlegesen végrehajtható.

Az `ipf(8)` man oldala tartalmazza a parancsnak megadható további beállításokat.

Az `ipf(8)` parancs a szabályokat tároló állományt egy szabványos szöveges állománynak tekinti, semmilyen szimbolikus helyettesítést alkalmazó szkriptet nem fogad el.

Lehetőségünk van azonban olyan IPF szabályokat készíteni, amelyek kiaknázzák a szkriptek szimbolikus helyettesítésének lehetőségeit. Erről bővebben lásd 30.5.9 Szakasz.

30.5.5. Az IPFSTAT

Az `ipfstat(8)` alapértelmezés szerint a arra használatos, hogy le tudjuk kérdezni és megjeleníteni a tűzfalhoz tartozó számlálók értékeit, amelyek a legutóbbi indítás vagy az `ipf -z` parancs által kiadott lenullázásuk óta a bejövő vagy kimenő forgalomból a megadott szabályoknak megfelelő csomagok alapján gyűjtenek össze statisztikákat.

A parancs működésének részleteit az `ipfstat(8)` man oldalon olvashatjuk.

Az `ipfstat(8)` meghívása alpból így néz ki:

```
input packets: blocked 99286 passed 1255609 nomatch 14686 counted 0
output packets: blocked 4200 passed 1284345 nomatch 14687 counted 0
input packets logged: blocked 99286 passed 0
output packets logged: blocked 0 passed 0
packets logged: input 0 output 0
log failures: input 3898 output 0
fragment state(in): kept 0 lost 0
fragment state(out): kept 0 lost 0
packet state(in): kept 169364 lost 0
packet state(out): kept 431395 lost 0
ICMP replies: 0 TCP RSTs sent: 0
Result cache hits(in): 1215208 (out): 1098963
IN Pullups succeeded: 2 failed: 0
OUT Pullups succeeded: 0 failed: 0
Fastroute successes: 0 failures: 0
TCP cksum fails(in): 0 (out): 0
Packet log flags set: (0)
```

Az `-i` mint bejövő (inbound), vagy az `-o` mint kimenő (outbound) forgalomra vonatkozó paraméterek megadásával a rendszerben az adott oldalon jelenleg telepített és alkalmazott szabályokat kérhetjük le és jeleníthetjük meg.

Az `ipfstat -in` parancs így a bejövő forgalomra vonatkozó belső szabályokat mutatja a szabályok számával.

Az `ipfstat -on` parancs a kimenő forgalmat érintő belső szabályokat mutatja a szabályok számával.

Az eredmény körülbelül ilyen lesz:

```
@1 pass out on xl0 from any to any
@2 block out on dc0 from any to any
@3 pass out quick on dc0 proto tcp/udp from any to any keep state
```

Az `ipfstat -ih` a bejövő forgalomhoz tartozó belső szabályokat mutatja és mindegyik elé odaírja, hogy eddig mennyi csomag illeszkedett rájuk.

Az `ipfstat -oh` ugyanígy a kimentő forgalom esetén mutatja a belső szabályokat és mindegyik előtt feltünteti, hogy az adott pillanatig mennyi csomag illeszkedett rájuk.

A kimenete nagyjából ilyen lesz:

```
2451423 pass out on xl0 from any to any
354727 block out on dc0 from any to any
430918 pass out quick on dc0 proto tcp/udp from any to any keep state
```

Az `ipfstat` parancs talán egyik legfontosabb funkciója a `-t` kapcsolóval csálható elő, melynek hatására a rendszerben aktív állapotok táblázatát mutatja meg ugyanúgy, ahogy a `top(1)` a FreeBSD rendszerben futó programokat. Amikor a tűzfalunk támadás alatt áll, ezzel a funkcióval tudjuk a problémát beazonosítani, leásni a

mélyébe és látni a támadótól érkező csomagokat. A kiegészítésképpen megadható alkapcsolók megadásával kiválaszthatjuk azt a cél vagy forrás IP-címet, portot vagy protokollt, amelyet valós időben meg akarunk figyelni. Ennek részleteit az `ipfstat(8)` man oldalán láthatjuk.

30.5.6. Az IPMON

Az `ipmon` megfelelő működéséhez be kell kapcsolnunk a rendszermag `IPFILTER_LOG` beállítását. Ez a parancs két különböző módon használható. Ha parancsot a `-D` opció nélkül gépeljük be, akkor ezek közül alapból a natív módot kapjuk meg.

A démon mód abban az esetben hasznos, ha folyamatosan naplózni akarjuk a rendszerben zajló eseményeket, majd később ezeket átnézni. Így képes egymással együttműködni a FreeBSD és az `IPFILTER`. A FreeBSD beépítve tartalmaz olyan lehetőséget, aminek révén magától cseréli a rendszernaplókat. Ezért ha átküldjük a `syslogd(8)` démonnak a naplózandó üzeneteket, akkor sokkal jobban járunk, mintha egyszerűen csak mezei állományba naplóznánk. Az `rc.conf` alapértelmezései között az `ipmon_flags` beállítás a `-Ds` kapcsolókat rögzíti:

```
ipmon_flags="-Ds" # D = indítás démonként
                  # s = naplózás a syslog használatával
                  # v = a tcp ablak, ack, seq csomagok naplózása
                  # n = az IP-címek és portok nevének feloldása
```

Ennek a viselkedésnek az előnyei minden bizonnyal egyértelműek. Segítségével képesek vagyunk az esetek megtörténte után átnézni, hogyan milyen csomagokat dobott el a rendszer, azok milyen címekről érkeztek és hova szánták. Ez egy komoly fegyver a támadók lenyomozásában.

Hiába engedélyezzük a naplózást, az `IPF` önszántából semmilyen naplózási szabályt nem fog gyártani. A tűzfal gazdájának kell eldöntenie, hogy a szabályokat közül melyiket akarja naplózni, és így neki kell megadnia a `log` kulcsszót ezekben az esetekben. Normális esetben csak a `deny` szabályokat naplózzák.

Egyáltalán nem ritka, hogy a szabályrendszer végén egy alapértelmezés szerint mindent eldobó szabály áll, amely naplóz. Ezzel lehetőségünk nyílik rögzíteni azokat a csomagokat, amelyek egyetlen szabályra sem illeszkedtek.

30.5.7. Naplózás az IPMON használatával

A `syslogd` egy saját módszert alkalmaz a naplózott adatok elkülönítésére. Egy “funkciók” (facility) és “szintek” (level) segítségével kialakított speciális csoportosítást alkalmaz. Az `IPMON -Ds` módja alapértelmezés szerint a `local0` “funkciót” használja. Ezen túl a következő szinteken különíthetjük el igényeinknek megfelelően a naplózott adatokat:

```
LOG_INFO - az átengedés vagy blokkolás helyett a "log" kulcsszóval ellátott csomagok
LOG_NOTICE - az át is engedett csomagok
LOG_WARNING - a blokkolt csomagok
LOG_ERR - a naplózott csomagok közül azok, amelyek túlságosan kicsik (hibás a fejlécük)
```

Az `IPFILTER` csak akkor tud naplózni a `/var/log/ipfilter.log` állományba, ha előtte létrehozzuk. Az alábbi parancs erre tökéletesen megfelelő:

```
# touch /var/log/ipfilter.log
```

A `syslogd(8)` működését az `/etc/syslog.conf` állományban szereplő definíciók vezérlik. A `syslog.conf` állomány számottevő mértékben képes meghatározni azt, ahogy a **syslog** az IPF és a hozzá hasonló alkalmazásoktól kapott rendszerszintű üzeneteket kezeli.

Az `/etc/syslog.conf` állományba az alábbi sor kell felvennünk:

```
local0.* /var/log/ipfilter.log
```

A `local0.*` megadásával az összes ilyen típusú üzenet egy előre rögzített helyre kerül.

Az `/etc/syslog.conf` állományban elvégzett módosításokat úgy léptethetjük érvénybe, ha újraindítjuk a számítógépet vagy az `/etc/rc.d/syslogd reload` paranccsal megkérjük a `syslogd(8)` démon, hogy olvassa újra az `/etc/syslog.conf` állományt.

Az imént létrehozott naplót ne felejtsük el megadni az `/etc/newsyslog.conf` állományban sem, és akkor ezzel a cseréjét is megoldjuk.

30.5.8. A naplózott üzenetek formátuma

Az `ipmon` által létrehozott üzenetek whitespace karakterekkel elválasztott adatmezőkből állnak. A következő mezők az összes üzenet esetében megjelennek:

1. A csomag megérkezésének dátuma
2. A csomag megérkezésének időpontja. ÓÓ:PP:MM.E alakban jelennek meg az órák, percek, másodpercek és ezredmásodpercek (ez több számjegy hosszú is lehet) szerint
3. Azon interfész a neve, ahol a csomag feldolgozásra került, például `dc0`
4. A szabályhoz tartozó csoport és sorszám, például `@0:17`

Ezek az `ipfstat -in` paranccsal nézhetőek meg.

1. Cselekvés: a `p` mint átment (passed), `b` mint blokkolt (blocked), `S` mint rövid csomag (short packet), `n` mint egyik szabályra sem illeszkedett (not match), `L` mint naplózás (log). A módosítók megjelenítésének sorrendje: `S`, `p`, `b`, `n`, `L`. A nagybetűs `P` és `B` azt jelzi, hogy a csomagot egy felsőbb szintű beállítás miatt naplózták, nem egy szabály hatására.
2. Címek: ez tulajdonképpen három mezőt takar: a forrás címet és portot (melyet egy vessző választ el), a `->` jelet és cél címet és portot. Például: `209.53.17.22,80 -> 198.73.220.17,1722`.
3. A `PR` után a protokoll neve vagy száma olvasható, például `PR tcp`.
4. A `len` csomaghoz tartozó fejléc és törzsének teljes hosszát jelöli, például `len 20 40`.

Amennyiben a csomag TCP, egy kötőjellel kezdődően további mezők is megjelenhetnek a beállított opcióknak megfelelő betűk képében. A betűket és beállításait az `ipf(5)` man oldalán olvashatjuk.

Amennyiben a csomag ICMP, a sort két mező zárja, melyek közül az első tartalma mindig "ICMP", és ezt egy perjellel elválasztva az ICMP üzenet típusa és altípusa követi. Tehát például az ICMP 3/3 a "nem elérhető port" üzenetet hordozza.

30.5.9. A szabályok felírása szimbolikus helyettesítéssel

Az IPF használatában gyakorlott felhasználók közül néhányan képesek olyan stílusú szabályrendszert készíteni, ahol szimbolikus helyettesítést használnak. Ennek az egyik legnagyobb előnye az, hogy ilyenkor elég csak a szimbolikus névhez tartozó értéket megváltoztatni és amikor a szkript lefut, akkor az összes rá hivatkozó szabályba ez kerül be. Szkript lévén a szimbolikus helyettesítéssel ki tudjuk emelni a gyakran használt értékeket és behelyettesíteni ezeket több helyre. Ezt a most következő példában láthatjuk.

Az itt alkalmazott felírás kompatibilis az sh(1), csh(1) és tcsh(1) parancsértelmezőkkel.

A szimbolikus helyettesítést egy dollárjellel fejezzük ki: \$.

A szimbolikus mezőkben nem szerepel a \$ jelölés.

A szimbolikus mező tartalmát kettős idézőjelbe (") tesszük.

Kezdjük így el a szabályok írását:

```
##### Az IPF szabályait tartalmazó szkript eleje #####

oif="dc0"           # a kimenő interfész neve
odns="192.0.2.11"   # az internet szolgáltató névszerverének IP-címe
myip="192.0.2.7"    # a szolgáltatótól kapott statikus IP-címünk
ks="keep state"
fks="flags S keep state"

# Választhatunk, hogy az /etc/ipf.rules állományt ebből a szkriptből
# hozzuk létre vagy futtathatjuk "magát" a szkriptet.
#
# Egyszerre csak az egyik sort használjuk.
#
# 1) Ezzel gyárthatjuk le az /etc/ipf.rules állományt:
#cat > /etc/ipf.rules << EOF
#
# 2) Ezzel futtathajuk "magát" a szkriptet:
/sbin/ipf -Fa -f - << EOF

# Engedélyezzük a szolgáltató névszerverének elérését.
pass out quick on $oif proto tcp from any to $odns port = 53 $fks
pass out quick on $oif proto udp from any to $odns port = 53 $ks

# Engedélyezzük kifelé a titkosítatlan www funkciót.
pass out quick on $oif proto tcp from $myip to any port = 80 $fks

# Engedélyezzük kifelé a TLS SSL felett üzemelő titkosított www funkciót.
pass out quick on $oif proto tcp from $myip to any port = 443 $fks
EOF
##### Itt az IPF szkript vége #####
```

Ennyi lenne. A példában szereplő szabályok most nem annyira lényegesek, a hangsúly most igazából a szimbolikus helyettesítésen és annak használatán van. Ha a fenti példát az /etc/ipf.rules.script állományba mentjük, akkor ezeket a szabályokat a következő paranccsal újra tudjuk tölteni:

```
# sh /etc/ipf.rules.script
```

Egyetlen aprócska gond van a beágyazott szimbólumokat tartalmazó állományokkal: az IPF maga nem képes megérteni a helyettesítéseket, azért közvetlenül nem olvassa a szkriptet.

Ez a szkript két módon hasznosítható:

- Vegyük ki megjegyzésből a `cat` paranccsal kezdődő sort, és tegyük megjegyzésbe az `/sbin/ipf` kezdetűt. A megszokottak szerint tegyük az `ipfilter_enable="YES"` sort az `/etc/rc.conf` állományba, majd minden egyes módosítása után futtassuk le a szkriptet az `/etc/ipf.rules` állomány létrehozásához vagy frissítéséhez.
- Tiltsuk le az IPFILTER aktiválását a rendszerindításkor, tehát írjuk bele az `ipfilter_enable="NO"` sort (ami mellel az alapértelmezett értéke) az `/etc/rc.conf` állományba.

Tegyük egy, az alábbi szkripthez hasonló az `/usr/local/etc/rc.d/` könyvtárba. A szkriptnek adjuk valamilyen értelmes nevet, például `ipf.loadrules.sh`. Az `.sh` kiterjesztés használata kötelező.

```
#!/bin/sh
sh /etc/ipf.rules.script
```

A szkript engedélyeit állítsuk be úgy, hogy a `root` tulajdonában legyen és képes legyen olvasni, írni valamint végrehajtani.

```
# chmod 700 /usr/local/etc/rc.d/ipf.loadrules.sh
```

Most miután a rendszer elindult, az IPF szabályai be fognak tölteni.

30.5.10. Szabályrendszerek az IPF-ben

Az IPF esetében a szabályrendszer olyan szabályokból áll, amelyek a csomagokról tartalmuk alapján eldöntik, hogy át kell engedni vagy vissza kell tartani. A gépek közt két irányban áramló csomagok egy munkamenet alapú társalgást képeznek. A tűzfalhoz tartozó szabályrendszer egyaránt feldolgozza a internetről a hálózatunk felé igyekvő csomagokat, illetve a hálózatunk ezekre adott válaszait. Az egyes TCP/IP szolgáltatásokat (mint például telnet, www, levelezés stb.) a hozzájuk tartozó protokoll és szabványos (fogadó) portszám írja le. Ezekre a forrásról általában valamilyen nem szabványos (magasabb értékű) portról érkeznek csomagok. Ekkor a kommunikáció összes paramétere (vagyis a portok és címek) bármelyike alapján definiálhatunk blokkolást vagy továbbengedést leíró szabályokat.

Az IPF eredetileg úgy íródott, hogy a szabályokat “az utolsó illeszkedő szabály nyer” stílusban dolgozza fel és csak állapot nélküli szabályokat ismert. Az idők folyamán az IPF szabályai kiegészültek a “quick” és az állapottartásra vonatkozó “keep state” opciókkal, amelynek köszönhetően óriási mértékben korszerűsödött a szabályok feldolgozása.

A szakaszban szereplő utasítások olyan szabályokat alkalmaznak, amelyekben egyaránt szerepel a “quick” és az állapottartásért felelős “keep state” beállítás. Ez az inkluzív tűzfalak létrehozásának egyik alapeszköze.

Figyelem A tűzfal szabályainak összeállítása során *nagyon óvatosnak* kell lennünk! Bizonyos beállítások hatására akár *ki is zárhatjuk magunkat* a szerverünkről. Az ebből fakadó esetleges kellemetlenségek elkerülése érdekében javasoljuk, hogy a tűzfal alapjait először helyi konzolról építsük fel, ne pedig távolról, például `ssh` segítségével.

30.5.11. A szabályok felépítése

A szabályok felépítésének bemutatását itt most leszűkítjük a modern állapottartó szabályokra és az “első illeszkedő szabály nyer” típusú feldolgozásra. A szabályok felírásának régebbi módjai az ipf(8) man oldalon találhatóak.

A # karakterrel egy megjegyzés kezdetét jelezzük, és általában a sor végén vagy egy külön sorban bukkan fel. Az üres sorokat a rendszer nem veszi figyelembe.

A szabályok kulcsszavakat tartalmaznak. Ezeknek a kulcsszavaknak balról jobbra haladva adott sorrendben kell szerepelniük. A kulcsszavakat kiemeltük. Egyes kulcsszavakhoz további beállítások is tartozhatnak, amelyek maguk is kulcsszavak lehetnek, és még további opciókkal rendelkezhetnek. Az alábbi nyelvtan mindegyik elemét kiemeltük és az alábbiakban egyenként kifejti a részleteiket.

*CSELEKVÉS BE-KI OPCIÓK SZŰRÉS ÁLLAPOTTARTÓ PROTOKOLL FORRÁS_CÍM, CÉL_CÍM OBJEKTUM
PORTSZÁM TCP_BEÁLLÍTÁS ÁLLAPOTTARTÓ*

CSELEKVÉS = block | pass

BE-KI = in | out

OPCIÓK = log | quick | on *interfész*

SZŰRÉS = proto *érték* | *forrás/cél IP* | port = *szám* | flags *beállítás*

PROTOKOLL = tcp/udp | udp | tcp | icmp

FORRÁS_CÍM, CÉL_CÍM = all | from *objektum* to *objektum*

OBJEKTUM = IP-cím | any

PORTSZÁM = *portszám*

TCP_BEÁLLÍTÁS = S

ÁLLAPOTTARTÓ = keep state

30.5.11.1. CSELEKVÉS

A cselekvés határozza meg, hogy mit kell tenni azokkal a csomagokkal, amelyek illeszkednek a szabály többi részére. Minden szabályhoz tartoznia *kell* egy cselekvésnek. A következő cselekvések közül választhatunk:

A block megadásával a szabályban szereplő szűrési feltételre illeszkedő csomagot eldobjuk.

A pass megadásával a szabályban szereplő szűrési feltételre illeszkedő csomagot átengedjük a tűzfalon.

30.5.11.2. BE-KI

Az összes szűrési szabály esetében kötelező egyértelműen nyilatkoznunk arról, hogy a bemenő vagy a kimenő forgalomra vonatkozik. Ezért a következő kulcsszó vagy az in vagy pedig az out, de közülük egyszerre csak az egyiket szabad használni, máskülönben a szabály hibásnak minősül.

Az in jelenti, hogy a szabályt az internet felől az adott interfészen beérkező csomagokra kell alkalmazni.

Az out jelenti, hogy a szabályt az internet felé az adott interfészen kiküldött csomagokra kell alkalmazni.

30.5.11.3. OPCÍÓK

Megjegyzés: Ezek az opciók csak a lentebb bemutatott sorrendben használhatók.

A `log` jelzi, hogy illeszkedés esetén a csomag fejlécét az `ipl` eszközön keresztül naplózni kell (lásd a naplózásról szóló szakaszt).

A `quick` jelzi, hogy illeszkedés esetén ez lesz a legutolsónak ellenőrzött szabály és így egy olyan “rövidzárat” tudunk képezni a feldolgozásban, amellyel elkerüljük a csomagra egyébként vonatkozó többi szabály illesztését. Ez az opció a korszerűsített szabályfeldolgozás kihasználásához elengedhetetlen.

Az `on` használatával a szűrés feltételei közé bevonhatjuk a csomaghoz tartozó hálózati interfészt. Itt az interfészek az `ifconfig(8)` által megjelenített formában adhatóak meg. Az opció megadásával csak az adott interfészen az adott irányba (befelé/kifelé) közlekedő csomagokra fog illeszkedni a szabály. Ez az opció a korszerűsített szabályfeldolgozás kihasználásához nélkülözhetetlen.

Amikor naplózunk egy csomagot, akkor a hozzá tartozó fejléc az `IPL` csomagnaplózo pszeudo eszközhöz kerül. A `log` kulcsszó után közvetlenül a következő minősítők szerepelhetnek (a következő sorrendben):

A `body` jelzi, hogy a csomag tartalmának első 128 byte-ját még jegyezzük fel a fejléc mellé.

A `first` minősítőt akkor érdemes használnunk, amikor a `log` kulcsszót a `keep state` opcióval együtt alkalmazzuk, mivel ilyenkor csak a szabályt kialakító csomag kerül naplózásra és nem minden olyan, ami illeszkedik az állapottartási feltételekre.

30.5.11.4. SZŰRÉS

Ebben a szakaszban olyan kulcsszavak jelenhetnek meg, amelyekkel a csomagok különféle tulajdonságai alapján ítélezhetünk azok illeszkedéséről. Itt adott egy kiinduló kulcsszó, amelyhez további kulcsszavak is tartoznak, és amelyek közül csak egyet választhatunk. Az alábbi általános tulajdonságok alapján tudjuk szűrni a csomagokat, ebben a sorrendben:

30.5.11.5. PROTOKOLL

A `proto` egy olyan kulcsszó, amelyhez hozzá kell rendelnünk még valamelyik opcióját is. Ez az opció segít az adott protokolloknak megfelelően válogatni a csomagok között. A korszerűsített szabályfeldolgozás lehetőségeinek kihasználásához nélkülözhetetlen.

Opcióként a `tcp/udp` | `udp` | `tcp` | `icmp`, vagy bármelyik, az `/etc/protocols` állományban megtalálható kulcsszó felhasználható. A `tcp/udp` ebből a szempontból speciálisnak tekinthető, mivel hatására egyszerre illeszthetőek a szabályra a TCP és UDP csomagok, és így a protokolltól eltekintve azonos szabályok felesleges többszörözését kerülhetjük el.

30.5.11.6. FORRÁS_CÍM/CÉL_CÍM

Az `all` kulcsszó gyakorlatilag a “from any to any” (“bárhonnan bárhova”) szinonímája és nem tartozik hozzá paraméter.

A `from` *forrás* `to` *cél* felépítése: a `from` és `to` kulcsszavak az IP-címek illesztésére használhatóak. Ilyenkor a szabályokban a *forrás* és a *cél* paramétereknek is szerepelniük kell. Az `any` egy olyan speciális kulcsszó, amely

tetszőleges IP-címre illeszkedik. Néhány példa az alkalmazására: `from any to any` vagy `from 0.0.0.0/0 to any`, `from any to 0.0.0.0/0`, `from 0.0.0.0/0 to any` vagy `from any to 0.0.0.0`.

Az IP-címek megadhatóak pontozott numerikus formában a hálózati maszk bitekben mért hosszával együtt, vagy akár egyetlen pontozott numerikus IP-címként.

Nincs lehetőség olyan IP-címtartományok illesztésére, amelyek nem adhatóak meg kényelmesen ponttal elválasztott számok és maszk hosszával. A `net-mgmt/ipcalc` port az ilyen számításokat könnyíti meg. A hálózati maszkok hosszának megállapításban segíthet az említett segédprogram (angol nyelvű) honlapja: <http://jodies.de/ipcalc>.

30.5.11.7. PORT

Amikor portra vonatkozó illeszkedést írunk elő, megadhatjuk a forrásra és célra, amit aztán vagy csak TCP vagy pedig csak UDP csomagokra alkalmazunk. A portok feltételeinek megfogalmazásánál használhatjuk a portok számát vagy az `/etc/services` állományban szereplő nevüket. Amikor a port egy `from` típusú objektum leírásában jelenik meg, akkor automatikusan a forrásportot jelenti, míg a `to` objektum leírásában pedig a célportot. A `to` objektumoknál a port megadása elengedhetetlen a korszerűsített szabályfeldolgozás előnyeinek kihasználásához. Példa: `from any to any port = 80`.

Az egyes portokat különböző műveletek segítségével, numerikusan hasonlíthatjuk össze, ahol akár porttartományt is megadhatunk.

`port "=" | "!=" | "<" | ">" | "<=" | ">=" | "eq" | "ne" | "lt" | "gt" | "le" | "ge"`.

A porttartományok megadásához használjuk a `port "<" | ">"` felírási módot.

FigyelemA forrásra és célra vonatkozó paraméterek után szereplő másik két paraméter nélkülözhetetlen a korszerűsített szabályfeldolgozás működéséhez.

30.5.11.8. TCP_BEÁLLÍTÁS

A beállítások csak a TCP forgalom szűrésénél érvényesülnek. A betűk jelölik azokat a lehetséges beállításokat, amelyek a TCP csomagok fejlécében megvizsgálhatóak.

A korszerűsített szabályfeldolgozás a `flags S` paraméter segítségével ismeri fel a TCP munkameneteket kezdeményező kéréseket.

30.5.11.9. ÁLLAPOTTARTÓ

A `keep state` jelzi, hogy a szabály paramétereinek megfelelő bármely csomag aktiválja az állapottartó szűrés használatát.

Megjegyzés: Ez a beállítás feltétlenül szükséges a korszerűsített szabályfeldolgozás megfelelő kihasználásához.

30.5.12. Állapottartó csomagszűrés

Az állapotottartó szűrés a csomagok kétirányú áramlását egy létrejött kapcsolatba sorolja be. Amikor aktiválódik, az állapotottartó szabály előre dinamikusan létrehozza a kétirányú kommunikációban megforduló csomagokhoz a megfelelő belső szabályokat. Olyan vizsgálatokat végez, amelyek segítségével ki tudja deríteni, hogy a csomag küldője és címzettje között fennálló kétirányú kapcsolat érvényes szabályok szerint zajlik-e. Minden olyan csomagot, amely nem illeszkedik megfelelően a kapcsolatra vonatkozó sémára, csalásnak tekintjük és automatikusan eldobjuk.

Az állapotottartás révén lehetőségünk van a TCP vagy UDP kapcsolatokhoz tartozó ICMP csomagokat is átengedni a tűzfalon. Tehát ha kapunk egy 3-as típusú, 4-es kódú ICMP választ valamilyen böngészésre használt állapotottartó szabályon keresztül kiküldött kérésre, akkor az automatikusan bejöhethet. Amelyik csomagot az IPF egyértelműen képes besorolni az aktív kapcsolatba, még ha az eltérő protokollt is használ, beengedi.

Ami ilyenkor történik:

Az internethez csatlakozó interfészen keresztül kifelé haladó csomagokat először egy dinamikus állapotábra alapján illesztjük, és ha a csomag illeszkedik az aktív kapcsolatban következőként várt csomagra, akkor átmegy a tűzfalon és a dinamikus állapotábrában frissül a kapcsolat állapota. Az aktív munkameneten kívül csomagok pedig egyszerűen a kimenő szabályrendszer szerint kerülnek ellenőrzésre.

Hasonlóan az előzőhöz, az internethez csatlakozó interfészen keresztül befelé haladó csomagokat először egy dinamikus állapotábra alapján illesztjük, és ha a csomag illeszkedik az aktív kapcsolatban következőként várt csomagra, akkor átmegy a tűzfalon és a dinamikus állapotábrában frissül a kapcsolat állapota. Az aktív munkamenethez nem tartozó csomagok pedig egyszerűen a bejövő szabályrendszer szerint kerülnek ellenőrzésre.

Amikor egy kapcsolat befejeződik, automatikusan törlődik a dinamikus állapotábrából.

Az állapotottartó csomagszűrés használatával az újonnan keletkező kapcsolatok elutasítására vagy engedélyezésére tudunk koncentrálni. Ha engedélyeztük egy új kapcsolat létrejöttét, akkor a rákövetkező összes többi csomag automatikusan átmegy a tűzfalon és minden más hamis csomag eldobódik. Ha tiltjuk az új kapcsolatot, akkor egyetlen rákövetkező csomag sem juthat át. Az állapotottartó szűrés által felkínált fejlett elemzési lehetőségek képesek védelmet nyújtani a behatolók részéről alkalmazott megannyi különböző támadási módszer ellen.

30.5.13. Példa inkluzív szabályrendszerre

A most következő szabályrendszer arra mutat példát, hogyan programozzunk le egy nagyon biztonságos inkluzív tűzfalat. Az inkluzív tűzfalak csak a szabályainak megfelelő szolgáltatásokat engedik keresztül, és alapértelmezés szerint minden mást blokkolnak. Egy hálózat gépeit védő tűzfalnak, amelyet gyakran “hálózati tűzfalnak” (network firewall) is neveznek, legalább két hálózati interfésszel kell rendelkeznie. Ezeket az interfészeket általában úgy állítják be, hogy tökéletesen megbíznak az egyik oldalban (a helyi hálózatban), a másikban (az internetben) pedig egyáltalán nem. A tűzfalat egyébként úgy is beállíthatjuk, hogy csak a tűzfalat működtető gépet védje — ezt “egyrendszeres tűzfalnak” (host based firewall) nevezik. Az ilyen típusú megoldásokat nem biztonságos hálózaton keresztül kommunikáló szervereknél alkalmaznak.

Mindegyik UNIX-típusú rendszert, köztük a FreeBSD-t is úgy alakították ki, hogy az operációs rendszeren belüli kommunikáció az 100 interfészen és a 127.0.0.1 IP-címen keresztül történik. A tűzfal szabályai között feltétlenül szerepelniük kell olyanoknak, amelyek lehetővé teszik ezen a speciális interfészen a csomagok zavartalan mozgását.

Az internetre csatlakozó interfészhez kell rendelni a kifelé és befelé haladó forgalom hitelesítését és a hozzáférésének vezérlését. Ez lehet a felhasználói PPP által létrehozott tun0 interfész vagy a DSL-, illetve kábelmodemhez csatlakozó hálózati kártya.

Ahol egy vagy több hálózati kártya is csatlakozik több különböző helyi hálózathoz, úgy kell beállítani a hozzájuk tartozó interfészeket, hogy egymás felé és az internet felé képesek legyenek küldeni és fogadni.

A szabályokat először három nagy csoportba kell szerveznünk: először jönnek a megbízható interfészek, ezeket követik az internet felé mutató interfészek, végül internet felől jövő, nem megbízható interfészek.

Az egyes csoportokban szereplő szabályokat úgy kell megadni, hogy közülük előre kerüljenek a leggyakrabban alkalmazottak, és a csoport utolsó szabálya blokkoljon és naplózzon minden csomagot az adott interfészen és irányban.

A kimenő forgalmat vezérlő szabályrendszer csak `pass` (tehát átengedő) szabályokat tartalmazhat, amelyek bentről az interneten elérhető szolgáltatásokat azonosítják egyértelműen. Az összes ilyen szabályban meg kell jelenni a `quick`, `on`, `proto`, `port` és `keep state` beállításoknak. A `proto tcp` szabályok esetében meg kell adni a `flag` opciót is, amivel fel tudjuk ismertetni a kapcsolatok keletkezését és ezen keresztül aktiválni az állapottartást.

A bejövő forgalmat vezérlő szabályrendszerben először az eldobni kívánt csomagokat kell megadni, aminek két eltérő oka van. Először is előfordulhat, hogy a veszélyes csomagok részleges illeszkedés miatt szabályosnak tűnnek. Az ilyen csomagokat értelemszerűen nem lenne szabad beengedni a szabályok részleges megfelelése alapján. A másodsor az eleve ismert problémás és értelmetlen csomagokat csendben el kellene vetni, mielőtt a szakaszhoz tartozó utolsó szabály fogná meg és naplózná. Ez az utolsó szabály egyébként szükség esetén felhasználható a támadók elleni bizonyítékok begyűjtésére.

A másik, amire még oda kell figyelnünk, hogy a blokkolt csomagok esetében semmilyen válasz nem keletkezzen, egyszerűen csak tűnjenek el. Így a támadó nem fogja tudni, hogy a csomagjai vajon elérték-e a rendszerünket. Minél kevesebb információt tudnak összegyűjteni a rendszerünkről a támadók, annál több időt kell szánniuk csínytevéseik kieselésére. A `log first` opciót tartalmazó szabályok csak az illeszkedésnél fogják naplózni a hozzájuk tartozó eseményt. Erre láthatunk példát az `nmap OS fingerprint` szabálynál. Az `security/nmap` segédprogramot a támadók gyakran alkalmazzák a megtámadni kívánt szerver operációs rendszerének felderítésére.

Minden `log first` opcióval megadott szabály illeszkedésénél a `ipfstat -hio` parancs meghatározódik az eddigi illeszkedések aktuális száma. Nagyobb értékek esetében következtethetünk arra, hogy a rendszerünket megtámadták (vagyis csomagokkal árasztják éppen el).

Az ismeretlen portszámok felderítésére az `/etc/services` állomány, esetleg a <http://www.securitystats.com/tools/portsearch.php> (angol nyelvű) honlap használható.

Érdemes továbbá megnézni a trójai programok által használt portokat a <http://www.simovits.com/trojans/trojans.html> címen (angolul).

A következő szabályrendszer egy olyan biztonságos “inkluzív” típusú tűzfal, amelyet éles rendszeren is használnak. Ezt a rendszerünkön nem használt szolgáltatásokra vonatkozó `pass` szabályok törlésével könnyedén a saját igényeink szerint alakíthatjuk.

Ha nem akarunk látni bizonyos üzeneteket, akkor vegyünk fel hozzájuk egy `block` típusú szabályt a befelé irányuló forgalomhoz tartozó szabályok közé.

A szabályokban írjuk át a `dc0` interfész nevét annak a hálózati kártyának az interfészére, amelyen keresztül csatlakozunk az internethez. A felhasználói PPP esetében ez a `tun0` lesz.

Tehát a következőket kell beírni az `/etc/ipf.rules` állományba:

```
#####
# A helyi hálózatunkon zajló forgalmat ne korlátozzuk.
# Csak akkor kell, ha helyi hálózathoz is csatlakozunk.
#####
```

```

#pass out quick on x10 all
#pass in quick on x10 all

#####
# A belső interfészen szintén ne korlátozzunk semmit.
#####
pass in quick on lo0 all
pass out quick on lo0 all

#####
# Az internet felé forgalmazó interfész (kimenő kapcsolatok)
# A saját hálózatunkról belülről vagy erről az átjáróról
# kezdeményezett kapcsolatokat vizsgáljuk az internet felé.
#####

# Engedélyezzük az internet szolgáltatók névszerverének elérését,
# az "xxx" helyett a névszervet IP-címét kell megadni.
# Másoljuk le ezeket a sorokat, ha a szolgáltatóknak több
# névszerverét is beakarjuk állítani. A címeiket az /etc/resolv.conf
# állományban találjuk.
pass out quick on dc0 proto tcp from any to xxx port = 53 flags S keep state
pass out quick on dc0 proto udp from any to xxx port = 53 keep state

# DSL vagy kábeles hálózatoknál engedélyezzük a
# szolgáltatónk DHCP szerverének elérését.
# Ez a szabály nem kell, ha "felhasználói PPP"-vel
# kapcsolódunk az internethez, ilyenkor tehát az egész
# csoport törölhető.
# Használjuk az alábbi szabályt és keressük meg a naplóban az
# IP-címet. Ha megtaláltuk, akkor tegyük bele a megjegyzésben
# szereplő szabályba és töröljük az első szabályt.
pass out log quick on dc0 proto udp from any to any port = 67 keep state
#pass out quick on dc0 proto udp from any to z.z.z.z port = 67 keep state

# Kifelé engedélyezzük a szabványos nem biztonságos WWW funkciókat.
pass out quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Kifelé engedélyezzük a biztonságos WWW funkciókat TLS SSL
# protokollal.
pass out quick on dc0 proto tcp from any to any port = 443 flags S keep state

# Kifelé engedélyezzük az e-mailek küldését és fogadását.
pass out quick on dc0 proto tcp from any to any port = 110 flags S keep state
pass out quick on dc0 proto tcp from any to any port = 25 flags S keep state

# Kifelé engedélyezzük az idő szolgáltatást.
pass out quick on dc0 proto tcp from any to any port = 37 flags S keep state

# Kifelé engedélyezzük az nntp híreket.
pass out quick on dc0 proto tcp from any to any port = 119 flags S keep state

# Kifelé engedélyezzük az átjáróról és a helyi hálózatról a nem
# biztonságos FTP használatát (passzív és aktív módokban is). Ez a

```

```

# funkció a működéséhez a nat szabályokat tartalmazó állományban
# hivatkozott FTP proxyt használja. Amennyiben a pkg_add paranccsal
# csomagokat akarunk telepíteni az átjáróra, erre a szabályra
# mindenképpen szükségünk lesz.
pass out quick on dc0 proto tcp from any to any port = 21 flags S keep state

# Kifelé engedélyezzük az ssh/sftp/scp # (biztonságos telnet/rlogin/FTP)
# szolgáltatások # elérését az SSH (secure shell) használatával.
pass out quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Kifelé engedélyezzük a nem biztonságos telnet elérését.
pass out quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Kifelé engedélyezzük FreeBSD CVSUp funkcióját.
pass out quick on dc0 proto tcp from any to any port = 5999 flags S keep state

# Kifelé engedélyezzük a pinget.
pass out quick on dc0 proto icmp from any to any icmp-type 8 keep state

# Kifelé engedélyezzük a helyi hálózatról érkező whois kéréseket.
pass out quick on dc0 proto tcp from any to any port = 43 flags S keep state

# Minden mást eldobunk és naplózzuk az első előfordulásukat.
# Ez a szabály blokkol alapértelmezés szerint mindent.
block out log first quick on dc0 all

#####
# Az internet felőli interfész (bejövő kapcsolatok)
# A saját hálózatunk felé vagy erre az átjáróra
# nyitott kapcsolatokat vizsgáljuk az internet felől.
#####

# Eldobjuk az összes olyan bejövő forgalmat, amit hivatalosan nem
# lehetne továbbítani vagy fenntartott címterülethez tartozik.
block in quick on dc0 from 192.168.0.0/16 to any      #RFC 1918: privát IP
block in quick on dc0 from 172.16.0.0/12 to any      #RFC 1918: privát IP
block in quick on dc0 from 10.0.0.0/8 to any         #RFC 1918: privát IP
block in quick on dc0 from 127.0.0.0/8 to any        #helyi
block in quick on dc0 from 0.0.0.0/8 to any          #helyi
block in quick on dc0 from 169.254.0.0/16 to any     #DHCP
block in quick on dc0 from 192.0.2.0/24 to any       #dokumentációs célokra fenntartva
block in quick on dc0 from 204.152.64.0/23 to any   #Sun klaszterek összekötésére használt
block in quick on dc0 from 224.0.0.0/3 to any       #D és E osztályú multicast

##### Itt eldobunk egy rakás csúf dolgot #####
# Ezeket nem akarjuk a naplóban látni:

# Eldobjuk a töredékcsoomagokat.
block in quick on dc0 all with frags

# Eldobjuk a túlságosan rövid TCP csomagokat.
block in quick on dc0 proto tcp all with short

```

```

# Eldobjuk a forrás által közvetített (source routed) csomagokat.
block in quick on dc0 all with opt lsrr
block in quick on dc0 all with opt ssrr

# Elutasítjuk az "OS fingerprint" kéréseket.
# Naplózzuk az első előfordulást, így nálunk lesz a kíváncsiskodó
# egyén IP-címe.
block in log first quick on dc0 proto tcp from any to any flags FUP

# Eldobunk mindent, aminek speciális beállításai vannak.
block in quick on dc0 all with ipopts

# Elutasítjuk a publikus pinget.
block in quick on dc0 proto icmp all icmp-type 8

# Elutasítjuk az ident kéréseket.
block in quick on dc0 proto tcp from any to any port = 113

# Blokkoljuk az összes Netbios szolgáltatást: 137=név, 138=datagram,
# 139=session. A Netbios az MS Windows megosztását implementálja.
# Blokkoljuk az MS Windows hosts2 névszerver kéréseit is a 81-es
# porton.
block in log first quick on dc0 proto tcp/udp from any to any port = 137
block in log first quick on dc0 proto tcp/udp from any to any port = 138
block in log first quick on dc0 proto tcp/udp from any to any port = 139
block in log first quick on dc0 proto tcp/udp from any to any port = 81

# Engedélyezzük a szolgáltatónk DHCP szerverétől érkező forgalmat.
# Ebben a szabályban meg kell adnunk a szolgáltató DHCP szerverének
# IP-címét, mivel itt csak a hiteles forrásból fogadunk el csomagokat.
# Erre csak DSL- és kábelmodemes kapcsolat esetében van szükség, a
# "felhasználói PPP" alkalmazása során szükségtelen. Ez az IP-cím
# megegyezik a kimenő kapcsolatoknál megadott címmel.
pass in quick on dc0 proto udp from z.z.z.z to any port = 68 keep state

# Befelé engedélyezzük a szabványos WWW funkciót, mivel webszerverünk
# van.
pass in quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Befelé engedélyezzük az internetről érkező nem biztonságos telnet
# kapcsolatokat. Azért nem biztonságos, mert az azonosítókat és
# jelszavakat titkosítatlan formában közli az interneten keresztül.
# Töröljük ezt a szabályt, ha nem használunk telnet szerveret.
#pass in quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Befelé engedélyezzük az internetről # érkező ssh/sftp/scp (biztonságos
# telnet/rlogin/FTP) # kapcsolatokat az SSH (secure shell) használatával.
pass in quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Minden mást dobjuk el és naplózzuk az első előfordulásukat.
# Az első alkalom naplózásával elejét tudjuk venni a "Denial of
# Service" típusú támadásoknak, amivel egyébként lehetséges lenne a
# napló elárasztása.

```

```
# Ez a szabály blokkol alapértelmezés szerint mindent.
block in log first quick on dc0 all
##### Itt van a szabályok vége #####
```

30.5.14. NAT

A NAT jelentése *Network Address Translation*, vagyis hálózati címfordítás. A Linux esetében ezt “IP masqueradingnak”, vagyis IP maszkolásnak hívják. A hálózati címfordítás és az IP maszkolás lényegben ugyanazt takarja. Az IPF címfordításért felelős funkciójának köszönhetően képesek vagyunk a tűzfal mögött elhelyezkedő helyi hálózat számára megosztani az internet-szolgáltatótól kapott publikus IP-címet.

Sokakban felmerülhet a kérdés, hogy erre vajon mi szükségünk lehet. Az internet-szolgáltatók a magánszemélyeknek általában dinamikus IP-címeket osztanak ki. A dinamikus itt arra utal, hogy a címünk minden alkalommal változik, amikor betárcsázunk a szolgáltatóhoz vagy amikor ki- és bekapcsoljuk a modemünket. Ez a dinamikus IP-cím fog azonosítani minket az interneten.

Most tegyük fel, hogy öt gépünk van otthon, viszont csak egyetlen előfizetéssel rendelkezünk. Ebben az esetben öt telefonvonalat kellene használnunk és mindegyik géphez előfizetni az internetre.

A hálózati címfordítás alkalmazásával azonban mindössze egyetlen előfizetés kell. A gépek közül négyet hozzákötünk egy switch-hez és a switch-et pedig a fennmaradó géphez, amelyen FreeBSD fut. Ez utóbbi lesz az így kialakított helyi hálózatunk átjárója. A tűzfalban működő címfordítás segítségével a helyi hálózaton található gépek IP-címeit észrevétlenül át tudjuk fordítani a hálózatunk publikus IP-címére, ahogy a csomagok elhagyják az átjárót. A beérkező csomagok esetében mindez visszafelé történik meg.

Az IP-címek közül adott egy tartomány, amit a címfordítást használó helyi hálózatok részére tartanak fenn. Az RFC 1918 szerint az alábbi IP-címtartományok használhatók a helyi hálózatban, mivel ezeken keresztül közvetlenül sosem lehet kijutni az internetre:

Kezdő IP: 10.0.0.0	-	Záró IP: 10.255.255.255
Kezdő IP: 172.16.0.0	-	Záró IP: 172.31.255.255
Kezdő IP: 192.168.0.0	-	Záró IP: 192.168.255.255

30.5.15. IPNAT

A címfordításra vonatkozó szabályokat az `ipnat` paranccsal tudjuk betölteni. Az ilyen típusú szabályokat általában az `/etc/ipnat.rules` állományban találjuk. A részleteket lásd az `ipnat(1)` man oldalán.

Amikor a címfordítás üzembe helyezése után meg akarjuk változtatni a címfordítás szabályait, először a címfordítás szabályait tartalmazó állományt módosítuk, majd a belső címfordítási szabályok és a címfordítási táblázatban szereplő aktív bejegyzések törléséhez futassuk le az `ipnat` parancsot a `-CF` beállítással.

A címfordítási szabályok újratöltését egy ehhez hasonló paranccsal tudjuk elvégezni:

```
# ipnat -CF -f /etc/ipnat.szabályok
```

A címfordításhoz tartozó statisztikákat ezzel a paranccsal tudjuk lekérdezni:

```
# ipnat -s
```

A címfordítási táblázatban pillanatnyilag szereplő összerendeléseket a következő paranccsal tudjuk listázni:

```
# ipnat -l
```

A szabályok feldolgozásával és az aktív szabályokkal/bejegyzésekkel kapcsolatos információk részletezését így engedélyezhetjük:

```
# ipnat -v
```

30.5.16. A címfordítási szabályok

A címfordítási szabályok nagyon rugalmasak és rengeteg olyan funkciót meg tudunk velük valósítani, ami az üzleti és otthoni felhasználók számára egyaránt hasznos.

Itt most a szabályok felépítését csak egyszerűsítve mutatjuk be, leginkább a nem üzleti környezetek tekintetében. A szabályok komplett formai leírását az `ipnat(5)` man oldalán találjuk.

Egy címfordítási szabály tehát valahogy így néz ki:

```
map INTERFÉSZ HELYI_IP_TARTOMÁNY -> PUBLIKUS_CÍM
```

A szabályt a `map` kulcsszó kezdi.

A `INTERFÉSZ` helyére az internet felé mutató külső interfész nevét írjuk be.

A `HELYI_IP_TARTOMÁNY` lesz az, amelyben a kliensek címeznek. Ez például a `192.168.1.0/24`.

A `PUBLIKUS_CÍM` lehet egy külső IP-cím vagy a `0/32` speciális kulcsszó, amellyel a `FELÜLET`-hez rendelt IP-címre hivatkozunk.

30.5.17. Hogyan működik a hálózati címfordítás

A publikus cél felé haladó csomag megérkezik a helyi hálózatról. Miután a kimenő kapcsolatokra vonatkozó szabályok átengedik, a címfordítás kapja meg a szerepet és fentről lefelé haladva neki lát alkalmazni a saját szabályait, ahol az első egyező szerint cselekszik. A címfordítás a szabályokat a csomaghoz tartozó interfészre és a forrás IP-címére illeszti. Amikor a csomag interfészének neve illeszkedik egy címfordítási szabályra, akkor ezután a csomag forrás (vagyis a helyi hálózaton belüli) IP-címéről igyekszik eldönteni, hogy a szabály nyílának bal oldalán szereplő tartományba esik-e. Ha erre is illeszkedik, akkor a forrás IP-címét átírjuk a `0/32` kulcsszó alapján felderített publikus IP-címre. A címfordító rutin ezt feljegyzi a saját belső táblázatába, így amikor a csomag visszatér az internetről, akkor képes lesz visszafordítani az eredeti belső IP-címére és feldolgozásra átadni a tűzfal szabályainak.

30.5.18. A címfordítás engedélyezése

A címfordítás életre keltéséhez a következőket kell beállítanunk az `/etc/rc.conf` állományban.

Először engedélyezzük a gépünknek, hogy közvetítsen forgalmat az interfészek között:

```
gateway_enable="YES"
```

Minden alkalommal indítsuk el a címfordításért felelős IPNAT programot:

```
ipnat_enable="YES"
```

Adjuk meg az IPNAT számára a betöltendő szabályokat:

```
ipnat_rules="/etc/ipnat.rules"
```

30.5.19. Hálózati címfordítás nagyon nagy helyi hálózatok esetében

Az olyan helyi hálózatokban, ahol rengeteg PC található vagy több alhálózatot is tartalmaz, az összes privát IP-cím egyetlen publikus IP-címbe tömörítése igen komoly problémává tud dagadni és az azonos portok gyakori használata a helyi hálózatra kötött számítógépek között ütközéseket okoz. Két módon tudunk megoldást nyújtani erre a problémára.

30.5.19.1. A használható portok kiosztása

Egy normális címfordítási szabály valahogy így nézne ki:

```
map dc0 192.168.1.0/24 -> 0/32
```

A fenti szabályban a csomag forrásportját az IPNAT változatlanul a feldolgozás után hagyja. Ha ehhez még hozzátesszük a `portmap` kulcsszót, akkor ezzel utasítani tudjuk az IPNAT-ot, hogy csak az adott tartományban képezze le a forrásportokat. Például a következő szabály hatására az IPNAT a forrásportokat egy adott tartományon belül fogja módosítani:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:60000
```

Ha viszont még inkább meg akarjuk könnyíteni a dolgunkat, akkor itt egyszerűen csak adjuk meg az `auto` kulcsszót, amellyel az IPNAT önmagától megállapítja, hogy milyen portokat tud használni:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp auto
```

30.5.19.2. Több publikus cím használata

Minden nagyobb helyi hálózat esetében elérkeünk ahhoz a ponthoz, ahol már egyetlen publikus cím nem elég. Ha több publikus IP-címmel is rendelkezünk, akkor ezekből a címekből egy “közös készletet” hozhatunk létre, amiből majd az IPNAT válogathat miközben a csomagok címeit átírja kifelé menetben.

Például ahelyett, hogy a csomagokat egyetlen publikus IP-címre képeznénk le, ahogy itt tesszük:

```
map dc0 192.168.1.0/24 -> 204.134.75.1
```

A hálózati maszk segítségével meg tudjuk adni IP-címek egy tartományát is:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/255.255.255.0
```

CIDR-jelöléssel:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/24
```

30.5.20. A portok átirányítása

Gyakran előfordul, hogy van webszerverünk, levelező szerverünk, adatbázis szerverünk és névszerverünk, melyek a helyi hálózat különböző gépein futnak. Ebben az esetben a szerverekhez tartozó forgalmat is fordítanunk kell, illetve valamilyen módon a bejövő forgalmat is át kell irányítanunk a helyi hálózat megfelelő gépeihez. Az IPNAT ezt a gondot a hálózati címfordítás átirányítást támogató funkcióival szünteti meg. Tegyük fel, hogy a 10.0.10.25 belső címen van egy webszerverünk, amelyhez a 20.20.20.5 publikus IP tartozik. Ilyenkor a következő szabályt adjuk meg:

```
rdr dc0 20.20.20.5/32 port 80 -> 10.0.10.25 port 80
```

vagy:

```
rdr dc0 0.0.0.0/0 port 80 -> 10.0.10.25 port 80
```

Így tudjuk beállítani a 10.0.10.33 címmel rendelkező névszervert a kintől érkező névfeloldási kérések fogadására:

```
rdr dc0 20.20.20.5/32 port 53 -> 10.0.10.33 port 53 udp
```

30.5.21. Az FTP és a címfordítás

Az FTP egy olyan őskövület, amely még az internet egy régi korszakából maradt fenn, amikor az egyetemek között még bérelt vonal létezett és az FTP szolgált a kutatók közt az állományok megosztására. Ez még abban az időben történt, amikor a biztonság egyáltalán nem volt lényeges szempont. Az évek előrehaladtával az FTP protokoll beleivódott a feltörekvő internet gerincébe és a titkosítatlanul küldött azonosítóival és jelszavaival továbbra is ugyanolyan védtelen maradt. Az FTP két változatban, aktív és passzív módban képes működni. Az eltérés kettejük között az adatcsatorna megállapításában van. A passzív mód sokkal biztonságosabb, mivel ilyenkor az adatcsatornát az FTP kapcsolatot kezdeményező állítja be. Az FTP különböző módjainak magyarázatát és a köztük levő különbséget a <http://www.slacksite.com/other/ftp.html> címen ismerhetjük meg részleteiben (angolul).

30.5.21.1. Az IPNAT szabályai

Az IPNAT egy speciális beépített FTP proxyval rendelkezik, amelyre a hálózati címfordítás leképezései között hivatkozhatunk. Képes figyelni az összes aktív vagy passzív FTP kapcsolathoz tartozó kimenő kérést és ezekhez dinamikusan létrehozni olyan ideiglenes szűrési szabályokat, amelyek valóban csak az adatcsatornához felhasznált portokat tartalmazzák. Ezzel ki tudjuk küszöbölni az FTP azon káros hatását a tűzfalra nézve, hogy egyszerre túlságosan sok magasabb tartománybeli port legyen nyitva.

Ez a szabály a belső hálózat összes FTP forgalmát lekezeli:

```
map dc0 10.0.10.0/29 -> 0/32 proxy port 21 ftp/tcp
```

Ez a szabály pedig az átjáróról érkező FTP forgalommal bírkózik meg:

```
map dc0 0.0.0.0/0 -> 0/32 proxy port 21 ftp/tcp
```

Ez a szabály kezeli a belső hálózatról érkező összes nem FTP típusú forgalmat:

```
map dc0 10.0.10.0/29 -> 0/32
```

Az FTP leképzésére vonatkozó szabály a szokásos leképzési szabály elé kerül. Az összes csomag fentről haladva az első illeszkedő szabály alapján kerül feldolgozásra. Először az interfész nevét vizsgáljuk, majd a belső hálózati forrás IP-t, végül azt, hogy a csomag egy FTP kapcsolat része. Ha minden paraméterében megfelel, akkor az FTP proxy készít egy ideiglenes szűrési szabályt hozzá, amellyel az FTP kapcsolathoz tartozó csomagok mind a két irányba képesek lesznek vándorolni, természetesen a címfordítással együtt. Az összes többi bentről érkező csomag átlép ezen a szabályon és megáll a harmadiknál, ahol az interfésznek és forrás IP-nek megfelelően átfordítjuk a címét.

30.5.21.2. Az IPNAT szűrési szabályai FTP-re

Az FTP esetében csak egyetlen szűrési szabályra van szükségünk a hálózati címfordításba épített FTP proxy használatához.

FTP proxy nélkül az alábbi három szabály kellene:

```
# Kifelé engedélyezzük a belső gépek FTP elérését az internet irányába,
# aktív és passzív módokban.
pass out quick on rl0 proto tcp from any to any port = 21 flags S keep state

# Kifelé engedélyezzük a passzív módhoz tartozó magasabb tartománybeli
# adatcsatornákat.
pass out quick on rl0 proto tcp from any to any port > 1024 flags S keep state

# Aktív módban beengedjük az FTP szerverről érkező adatcsatornát.
pass in quick on rl0 proto tcp from any to any port = 20 flags S keep state
```

30.6. IPFW

Az IPFW (IPFW) a FreeBSD által támogatott tűzfalazó alkalmazás, melyet a FreeBSD Projektben résztvevő önkéntesek fejlesztettek ki és tartanak karban. Régi típusú, állapotmentes szabályokat használ, és az itt használatos szabályírási technikát “egyszerű állapotmentes megoldásnak” nevezzük.

Az IPFW szabvány FreeBSD-ben levő, mintaként szolgáló szabályrendszere (ez az `/etc/rc.firewall` és `/etc/rc.firewall6` állományokban található meg) annyira egyszerű, hogy komolyabb módosítások nélkül nem ajánlatos használni. Ez a példa nem tartalmaz állapotmentes szűrést, ami viszont a legtöbb esetben kívánatos lenne, ezért ezt a szakaszt nem erre alapozzuk.

Az IPFW állapotmentes szabályainak felépítésében olyan technikailag kifinomult leválogatási képességek bújnak meg, amelyek jócskán meghaladják az átlagos tűzfalépítők tudását. Az IPFW elsősorban olyan szakemberek vagy szakmailag előrehaladott felhasználók számára készült, akiknek speciális csomagszűrési igényeik vannak. A különböző protokollok használatának és a hozzájuk tartozó fejlcinformációk mindenre kiterjedő ismerete szinte nélkülözhetetlen az IPFW valódi erejének kihasználásához. Ez a szint azonban túlmutat a kézikönyv ezen szakaszának keretein.

Az IPFW hét komponensből épül fel, melyek közül az elsődleges a rendszermag tűzfalazásért felelős szabályfeldolgozó és a hozzá tartozó csomagnyilvántartás, majd ezt követi a naplózás, a hálózati címfordítást aktiváló `divert` szabály, valamint a komolyabb célok megvalósítására alkalmas lehetőségek: a forgalom korlátozásáért felelős `dummynet`, a továbbküldésre alkalmas `fwd rule` szabály, a hálózati hidak támogatása, illetve az `ipstealth`. Az IPFW egyaránt használható IPv4 és IPv6 esetén.

30.6.1. Az IPFW engedélyezése

Az IPFW az alap FreeBSD telepítésben külön, futás időben betölthető modulként érhető el. Ha az `rc.conf` állományban megadjuk a `firewall_enable="YES"` beállítást, akkor a rendszer indulásakor ezt a modult dinamikusan betölti. Az IPFW-t csak akkor kell a FreeBSD rendszermagjába beépítenünk, ha szükségünk van a címfordítási funkciójára is.

Ha tehát az `rc.conf` állományban megadtuk a `firewall_enable="YES"` sort és újraindítottuk a számítógépünket, akkor a következő fehérrel kiemelt üzenet fog megjelenni a rendszerindítás során:

```
ipfw2 initialized, divert disabled, rule-based forwarding disabled, default to deny, logging disabled
```

A “logging disabled” üzenetből kiderül, hogy a modul nem végez naplózást. A naplózást és a hozzá tartozó részletesség szintjét úgy tudjuk beállítani, ha az `/etc/sysctl.conf` állományba felvesszük a következő sorokat, amivel a következő indításkor már működni fog:

```
net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose_limit=5
```

30.6.2. A rendszermag beállításai

Ha nem akarjuk kihasználni az IPFW által felkínált címfordítási lehetőségeket, akkor egyáltalán nem szükséges a FreeBSD rendszermagjába belefördítani a támogatását. Ezért az alábbiakat csak kiegészítő információként tüntettük fel.

```
options      IPFIREWALL
```

Ez a beállítás engedélyezi az IPFW használatát a rendszermag részeként.

```
options      IPFIREWALL_VERBOSE
```

Ezzel és a `log` kulcsszóval tudjuk az IPFW szabályain keresztülhaladó csomagokat naplózni.

```
options      IPFIREWALL_VERBOSE_LIMIT=5
```

Ez az érték korlátozza a `syslogd(8)` segítségével naplózott azonos bejegyzések maximális számát. Ezt a beállítást olyan veszélyes környezetekben érdemes használnunk, ahol naplózni akarunk. Segítségével meg tudjuk akadályozni, hogy a rendszernapló elárasztásával megakasszák a rendszerünket.

```
options      IPFIREWALL_DEFAULT_TO_ACCEPT
```

Ezen beállítás hatására a tűzfal alapértelmezés szerint mindent átenged, ami általában akkor jöhet jól, amikor először beállítjuk a tűzfalat.

```
options      IPDIVERT
```

Ezzel a beállítással engedélyezzük a címfordítás használatát.

Megjegyzés: Ha nem adjuk meg az `IPFIREWALL_DEFAULT_TO_ACCEPT` beállítást, vagy ha nem engedélyezzük a bejövő csomagokat, akkor a gépünkre semmilyen csomag nem lesz képes bejutni, illetve onnan kijutni.

30.6.3. Az `/etc/rc.conf` beállításai

Így tudjuk engedélyezni a tűzfalat:

```
firewall_enable="YES"
```

A FreeBSD-hez mellékelt alapértelmezett tűzfaltípusok közül az `/etc/rc.firewall` állomány átolvasásával tudunk választani, és megadni az alábbi helyett:

```
firewall_type="open"
```

A következő értékek állnak rendelkezésünkre:

- `open` — átengedi az összes forgalmat
- `client` — csak ezt a gépet védi
- `simple` — az egész hálózatot védi
- `closed` — a helyi interfész kivételével minden IP alapú forgalmat tilt
- `UNKNOWN` — tiltja a tűzfal szabályainak betöltését
- *állománynév* — a tűzfal szabályait tartalmazó állomány abszolút elérési útvonala

Két különböző módon lehet betölteni a saját **ipfw** szabályainkat. Az egyik közülük, ha a `firewall_type` változóban megadjuk a *tűzfal szabályait* tartalmazó állomány abszolút elérési útvonalát, az `ipfw(8)` parancssori beállításai nélkül. Az alábbi példában egy olyan egyszerű szabályrendszert láthatunk, amely blokkolja az összes bejövő és kimenő forgalmat:

```
add deny in
add deny out
```

Másrésről az `firewall_script` változóban is megadhatjuk azt a szkriptet, amelyben a rendszerindítás során meghívjuk `ipfw` parancsot. Az iménti szabályrendszert az alábbi szkripttel tudjuk kiváltani:

```
#!/bin/sh

ipfw -q flush

ipfw add deny in
ipfw add deny out
```

Megjegyzés: Ha a `firewall_type` változó `client` vagy `simple` értékét használjuk, akkor az `/etc/rc.firewall` állományban található alapértelmezett szabályokat érdemes átvizsgálnunk, hogy kellően illeszkednek-e az adott géphez. Hozzátennénk, hogy a fejezetben szereplő példák azt feltételezik, hogy a `firewall_script` értéke az `/etc/ipfw.rules` állomány.

A naplózás így engedélyezhető:

```
firewall_logging="YES"
```

FigyelemA `firewall_logging` változó egyedül csak annyit tesz, hogy beállítja a `net.inet.ip.fw.verbose` `sysctl` változónak az `1` értéket (lásd 30.6.1 Szakasz). A napló korlátozására nincs külön változó az `rc.conf` állományon belül, de az `/etc/sysctl.conf` állomány segítségével és manuálisan be tudjuk állítani a hozzá tartozó változót:

```
net.inet.ip.fw.verbose_limit=5
```

Amennyiben a gépünk átjáróként viselkedik, tehát a `natd(8)` segítségével címfordítást végez, a 31.9 Szakaszban olvashatunk utána, hogy ehhez az `/etc/rc.conf` állományban milyen beállításokat kell megadnunk.

30.6.4. Az IPFW parancs

Normál esetben az `ipfw` parancs használatos arra, hogy a tűzfal működése közben az aktív belső szabályai közé vegyünk fel vagy töröljünk közülük manuálisan bejegyzéseket. Ennek a módszernek az egyedüli hátránya, hogy az így végrehajtott módosítások el fognak veszni a rendszer leállításával. Itt inkább azt a megoldást javasoljuk, hogy az összes szabályt tegyük bele egy állományba és a rendszerindítás során ezt töltjük be, majd ha változtatni akarunk a tűzfalon, akkor ezt az állományt módosítsuk és a régiek törlésével töltjük be újra az egész szabályrendszert.

Az `ipfw` parancs mellesleg remekül használható a jelenleg futó tűzfalszabályok megjelenítésére a konzolon. Az IPFW nyilvántartásában az egyes szabályokhoz dinamikusan jönnek létre számlálók, amelyek a rá illeszkedő csomagokat számolják. A tűzfal tesztelése folyamán a szabályok és hozzá tartozó számlálók lekérdezése a megfelelő működés ellenőrzésének egyik lehetséges módja.

A szabályokat így tudjuk egymás után felsoroltatni:

```
# ipfw list
```

A szabályokat így tudjuk az utolsó illeszkedésük idejével együtt megjeleníteni:

```
# ipfw -t list
```

A következő példában a nyilvántartási információkat kérdezzük le, ekkor a szabályok mellett az illeszkedő csomagok száma is láthatóvá válik. Az első sorban a szabály száma szerepel, majd ezt követi rendre az illeszkedő kimenő és bejövő csomagok mennyisége, valamint végül maga a szabály.

```
# ipfw -a list
```

A statikus szabályok mellett a dinamikusakat így lehet kilistázni:

```
# ipfw -d list
```

A lejárt dinamikus szabályokat is meg tudjuk nézni:

```
# ipfw -d -e list
```

A számlálók nullázása:

```
# ipfw zero
```

Csak a SZÁM sorszámu szabályhoz tartozó számlálók nullázása:

```
# ipfw zero SZÁM
```

30.6.5. Szabályrendszerek az IPFW-ben

Az IPFW esetében a szabályrendszer olyan szabályokból áll, amelyek a csomagokról tartalmuk alapján eldöntik, hogy át kell engedni vagy vissza kell tartani. A gépek közt két irányban áramló csomagok egy munkamenet alapú társalgást képeznek. A tűzfalhoz tartozó szabályrendszer egyaránt feldolgozza a internetről a hálózatunk felé igyekvő csomagokat, illetve a hálózatunk ezekre adott válaszait. Az egyes TCP/IP szolgáltatásokat (mint például telnet, www, levelezés stb.) a hozzájuk tartozó protokoll és szabványos (fogadó) portszám írja le. Ezekre a forrásról általában valamilyen nem szabványos (magasabb értékű) portról érkeznek csomagok. Ekkor a kommunikáció összes paramétere (vagyis a portok és címek) bármelyike alapján definiálhatunk blokkolást vagy továbbengedést leíró szabályokat.

Amikor egy csomag eléri a tűzfalat, a szabályrendszer első szabályával kerül összehasonlításra és amíg nem illeszkedik valamelyikre, addig lefut rá a többi szabály is fentről lefelé egyesével, a sorszámnak megfelelő növekvő sorrendben. Ha a csomag megfelel valamelyik szabály leválogatási paramétereinek, akkor a benne megnevezett cselekvés zajlik le, és számára a feldolgozás befejeződik. Ezt a viselkedést neveztük “az első illeszkedés nyer” típusú keresésnek. Amennyiben a csomag egyetlen szabályra sem illeszkedik, akkor az IPFW 65535-ös sorszámu állandó szabálya fogja elcsípni, amely feladata szerint eldobja az összes hozzá beérkező csomagot anélkül, hogy bármit is válaszolna a csomag feladójának.

Megjegyzés: A keresés a count, skipto és tee szabályok után még folytatódik.

Az itt szereplő utasítások különböző állapottartásra vonatkozó opciókat, például a keep state, limit, in, out és via kulcsszavakat tartalmazó szabályokon alapulnak. Lényegében ezt tekinthetjük az inkluzív típusú tűzfalak kiindulási alapjaként.

Figyelem A tűzfal szabályainak beállítása során nem árt óvatosnak lennünk, mert figyelmetlenségünk révén könnyen kizárhatjuk magunkat a gépünkről.

30.6.5.1. A szabályok felépítése

Az itt bemutatásra kerülő szabályok felépítését csak olyan mértékig részletezzük, ami elengedő a szabványos inkluzív típusú tűzfalak kialakításához. A szabályok felépítésének pontos leírását az ipfw(8) man oldalán találhatjuk meg.

A szabályok kulcsszavakat tartalmaznak. Ezeket a kulcsszavakat soronként egy előre rögzített sorrendben kell szerepeltetni. A kulcsszavakat a szövegben kiemeltük. Bizonyos kulcsszavakhoz további opciókhoz is tartozhatnak, amelyek gyakran maguk is kulcsszavak és szintén további opciókat tartalmazhatnak.

A # egy megjegyzés kezdetét jelzi, mely egyaránt megjelenhet egy külön sorban, vagy egy szabályt tartalmazó sor végén. Az üres sorok nem vesznek részt a feldolgozásban.

PARANCS SZABÁLY_SZÁM CSELEKVÉS NAPLÓZÁS SZŰRÉS ÁLLAPOTTARTÁS

30.6.5.1.1. PARANCS

Minden új szabály előtt az *add* (mint hozzáadás) parancsnak kell szerepelni, amellyel a belső táblázatba tudjuk felvenni.

30.6.5.1.2. SZABÁLY_SZÁM

A szabályokhoz mindig tartozik egy sorszám is.

30.6.5.1.3. CSELEKVÉS

A szabályhoz az alábbi cselekvések valamelyike kapcsolható, amely akkor hajtódik végre, amikor a csomag megfelel a hozzá tartozó szűrési feltételeknek.

allow | *accept* | *pass* | *permit*

A fentiek közül mindegyik ugyanazt jelenti, vagyis hatásukra az illeszkedő csomag kilép a tűzfalból. Ez a szabály megállítja a keresést.

check-state

A csomagot a dinamikus szabályokat tároló táblázattal veti össze. Ha itt egyezést talál, akkor végrehajtja az egyező dinamikus szabályhoz tartozó cselekvést, minden más esetben továbblép a következő szabályra. Ennek a szabálynak nincs illeszthető paramétere. Ha a szabályrendszerben nem szerepel ilyen, akkor a dinamikus szabályok vizsgálatát az első *keep-state* vagy *limit* használatánál vonja be a rendszer.

deny | *drop*

Mind a két szó ugyanarra utal, vagyis a szabályra illeszkedő csomagokat el kell dobni. Ebben az esetben a keresés befejeződik.

30.6.5.1.4. NAPLÓZÁS

log vagy *logamount*

Amikor egy csomag egy *log* kulcsszót tartalmazó szabályra illeszkedik, akkor a rendszernaplóban egy üzenet keletkezik a *security* (biztonság) funkción keresztül. A naplóba ténylegesen csak akkor kerül bele az üzenet, ha az adott szabály még nem haladta meg a hozzá tartozó *logamount* paraméter értékét. Ha ezt nem adtuk meg, akkor az itt érvényes korlát a *net.inet.ip.fw.verbose_limit* sysctl változóból fog származni. A nulla érték mind a két esetben megszünteti ezt a korlátozást. Ha elértük a korlátot, akkor a naplózást úgy tudjuk újra engedélyezni, ha töröljük a naplózáshoz tartozó számláló értékét, lásd az *ipfw reset log* parancsot.

Megjegyzés: A naplózás mindig az összes paraméter illeszkedésének ellenőrzése után történik, de még a cselekvés (*accept*, *deny*) elvégzése előtt. Teljesen rajtunk múlik, hogyan milyen szabályokat naplózunk.

30.6.5.1.5. SZŰRÉS

Ebben a szakaszban azok a kulcsszavak találhatók, amelyek segítségével a csomagok különböző tulajdonságait tudjuk megvizsgálni és eldönteni, hogy illeszkedik-e a szabályra vagy sem. A következő általános tulajdonságokat tudjuk megvizsgálni, ebben a kötött sorrendben:

udp | tcp | icmp

Bármilyen más olyan protokoll is megadható, amely megtalálható az `/etc/protocols` állományban. Ezzel adjuk a csomaghoz tartozó protokollt. Használata kötelező.

from forrás to cél

Mind a `from` és `to` kulcsszavak IP-címek illesztésére alkalmasak. A szabályoknak tartalmazniuk kell a *forrás* ÉS a *cél* paramétereket is. Az `any` egy olyan kulcsszó, amely tetszőleges IP-címre illeszkedik. A `me` pedig egy olyan speciális kulcsszó, amely a tűzfalat működtető FreeBSD-s gép (tehát ez a gép) adott interfészhez tartozó IP-címét jelöli, mint ahogy a `from me to any`, `from any to me`, `from 0.0.0.0/0 to any`, `from any to 0.0.0.0/0`, `from 0.0.0.0 to any`, `from any to 0.0.0.0` vagy `from me to 0.0.0.0` paraméterekben. Az IP-címek numerikus pontozott formában a hálózati maszk hosszával együtt (CIDR-jelöléssel), vagy egyszerűen csak pontozott formában adhatóak meg. A hálózati maszkok megállapításában a `net-mgmt/ipcalc` port lehet segítségünkre. Erről bővebb információkat a segédprogram honlapján, a <http://jodies.de/ipcalc> címen találhatunk (angolul).

port szám

A portszámokat is ismerő protokollok esetében (mint például a TCP vagy UDP) adhatjuk meg. Fontos, hogy itt annak a szolgáltatásnak a portszámát adjuk meg, amelyre a szabály vonatkozik. A szolgáltatás (az `/etc/services` állományból származó) nevét is megadhatjuk a port száma helyett.

in | out

A beérkező valamint a kimenő csomagokat adhatjuk meg ezen a módon. Itt az `in` és `out` kulcsszavak, melyeket kötelező megadni a szabály részeként.

via interfész

Név szerint az adott interfészen keresztül haladó csomagokat tudjuk szűrni. A `via` kulcsszó hatására a használt interfész is számítani fog a csomag feldolgozása során.

setup

Ez a kulcsszó a TCP csomagok esetében a kapcsolatok felépítésére vonatkozó kéréseket segít beazonosítani.

keep-state

Ez egy kötelező kulcsszó. Feldolgozásakor a tűzfal létrehoz dinamikus szabályt, amely alapértelmezés szerint az egyazon protokollt használó forrás és cél IP/port párosok közti kétirányú forgalomra fog automatikusan illeszkedni.

limit {forráscím | forrásport | célcím | célport}

A tűzfal csak *N* darab, a szabálynak megfelelő azonos paraméterű kapcsolatot fog átengedi. Itt egy vagy több forrás- és célcím valamint forrás- és célport adható meg. A `limit` és a `keep-state` egy szabályon belül nem használható. A `limit` ugyanazokat az állapottartó funkciókat képviseli, mint a `keep-state`, csak a saját kiegészítéseivel megtoldva.

30.6.5.2. ÁLLAPOTTARTÁS

Az állapottartó szűrés a kétirányú csomagváltásokat egy létrejött kapcsolatba sorolja. Olyan vizsgálatokat végez, amivel képes megállapítani, hogy a csomag küldője és címzettje között kialakult kommunikáció követ-e valamilyen kétirányú csomagküldésre érvényes folyamatot. Az így felállított sablontól eltérő összes csomag hamisnak minősül és automatikusan eldobásra kerül.

A `check-state` segítségével ellenőrizhetjük, hogy az adott csomag a IPFW szerint megfelel-e valamelyik dinamikusan leképzett szabálynak. Ha egyezik valamelyikőjükkel, akkor a csomag a tűzfalból kilépve folytatja útját és a kommunikációban soron következő csomag számára létrejön egy másik dinamikus szabály. Ha nincs egyezés, akkor csomag feldolgozása a szabályrendszer következő szabályánál folytatódik.

A dinamikus szabályokat kezelő rutin sebezhető, mivel ha egyszerre nagy mennyiségű SYN csomagot küldünk, akkor olyan sok dinamikus bejegyzés keletkezik, hogy egyszerűen kifogyunk a rendelkezésre álló erőforrásokból. A FreeBSD fejlesztői azonban az ilyen természetű támadások kivédésére is felkészítették, és kialakították belőle a `limit` opciót. Alkalmazásával le tudjuk korlátozni az egyszerre folyó párhuzamos kapcsolatok számát a forrás vagy a cél a `limit` paraméternél megadott mezőinek és a csomag IP-címe alapján. Így az adott szabályhoz és IP-címhez csak előre rögzített mennyiségű nyitott állapotú dinamikus szabály létezhet egy időben. Ha ezt a korlátot átlépjük, a csomag eldobódik.

30.6.5.3. A tűzfal üzeneteinek naplózása

A naplózás előnyei nyilvánvalóak. Ha engedélyezzük, aktiválása után képesek leszünk olyan információknak utánanézni, mint például milyen csomagokat dobtunk el, honnan érkeztek, hova tartottak. Ez egy komoly fegyverünk lehet a potenciális támadókkal szemben.

Azonban hiába engedélyezzünk önmagában a naplózást, attól az IPFW még saját magától nem fog naplózást előíró szabályokat gyártani. A tűzfal karbantartóinak maguknak kell eldöntenie, hogy a szabályrendszerben mely szabályokhoz tartozzon naplózás, nekik kell felvenni ezekhez a `log` kulcsszót. Általában csak az eldobással járó deny típusú szabályokat vagy a bejövő ICMP pingeket szokták naplózni. Gyakran úgy oldják meg ezt, hogy a szabályrendszer utolsó szabályaként lemásolják az `ipfw` alapértelmezett "mindent eldobunk" szabályát és a naplózást adják meg benne. Ezen a módon fény derül azokra a csomagokra, amelyek a szabályrendszerben semmire sem illeszkedtek.

A naplózás azonban egy kétélű fegyver, mivel ha nem vagyunk elég körültekintőek, akkor a sok naplóinformáció között könnyen el tudunk veszni és a lemezünk is gyorsan betelhet a mindent elfoglaló naplólóktól. Mellesleg a naplók megdagasztását célzó DoS típusú támadás a rendszerek lebénítására alkalmazott egyik legősibb technika. Ezek az üzenetek nem csak a rendszernaplóba kerülnek bele, hanem az elsődleges konzol képernyőjére is kiíródnak, ami egy idő után idegesítő tud lenni.

A rendszermag `IPFWALL_VERBOSE_LIMIT=5` beállításával azonban képesek vagyunk korlátozni azokat a rendszernapló felé küldött egymás után következő üzeneteket, amelyek ugyanarra a szabályra vonatkoznak. Amikor ezt a beállítást megadjuk a rendszermag fordításánál, akkor az egyes szabályokhoz az általa meghatározott értéken felül nem jön létre több hasonló üzenet. Hiszen semmi sem derül ki 200 teljesen azonos naplóüzenetből. Például, ha az egyes szabályokhoz legfeljebb öt egymást követő üzenetet engedélyezünk, akkor a többi fennmaradó azonos üzenetet összeszámolja a rendszer és a következő módon közvetíti a rendszernapló szolgáltató felé:

```
last message repeated 45 times
```

Ami magyarul így hangzik:

```
az utolsó üzenet 45 alkalommal ismétlődött meg
```

Az összes csomagokkal kapcsolatos naplózás alapértelmezés szerint a `/var/log/security` állományba kerül, amelyet az `/etc/syslog.conf` állomány definiál.

30.6.5.4. Szabályokat tartalmazó szkript készítése

A rutinosabb IPFW felhasználók a szabályokat egy állományban programozzák le olyan stílusban, hogy szkriptként is futtatható legyen. Ennek az egyik legnagyobb előnye, hogy a tűzfal szabályai így egyszerre cserélhetők a rendszer újraindítása nélkül. Ez a módszer nagyon kényelmes az új szabályok kipróbálásánál, mivel tetszőleges alkalommal végrehajthatjuk. Mivel ez egy szkript, ki tudjuk használni az itt megszokott szimbolikus helyettesítés által felkínált lehetőségeket, és ezzel a gyakran használt értékeket is egyszerre több szabályban tudjuk helyettesíteni. Erre a következőkben fogunk egy konkrét példát látni.

A szkript felépítése kompatibilis a sh(1), csh(1) és tcsh(1) parancsértelmezőkkel. A szimbolikus mezők helyettesítését a \$ vagyis dollárjel vezeti be. Maguk a szimbolikus mezők nem tartalmazzák a \$ előtagot. A szimbolikus mezők értékeit "kettős idézőjelek" között kell megadni.

A szabályok összeírását kezdjük el így:

```
##### itt kezdődik az ipfw szabályait tartalmazó szkript #####
#
ipfw -q -f flush          # töröljük az összes aktuális szabályt
# Set defaults
oif="tun0"                # a kimenő interfész
odns="192.0.2.11"         # az internet szolgáltató névszerverének IP-címe
cmd="ipfw -q add "        # a szabályok hozzáadásához szükséges elemek
ks="keep-state"           # csupán a lustaság miatt
$cmd 00500 check-state
$cmd 00502 deny all from any to any frag
$cmd 00501 deny tcp from any to any established
$cmd 00600 allow tcp from any to any 80 out via $oif setup $ks
$cmd 00610 allow tcp from any to $odns 53 out via $oif setup $ks
$cmd 00611 allow udp from any to $odns 53 out via $oif $ks
#### itt fejeződik be az ipfw szabályait tartalmazó szkript #####
```

Ezzel készen is vagyunk. Most ne törődjünk a példában szereplő szabályokkal, itt most a szimbolikus helyettesítés használatát igyekeztük bemutatni.

Ha az iménti példát az /etc/ipfw.rules állományba mentettük el, akkor az alábbi parancs kiadásával tudjuk újratölteni a benne szereplő szabályokat:

```
# sh /etc/ipfw.rules
```

Az /etc/ipfw.rules állományt egyébként tetszőleges néven hívhatjuk és bárhová rakhatjuk.

Ugyanez természetesen elérhető a következő parancsok egymás utáni begépelésével is:

```
# ipfw -q -f flush
# ipfw -q add check-state
# ipfw -q add deny all from any to any frag
# ipfw -q add deny tcp from any to any established
# ipfw -q add allow tcp from any to any 80 out via tun0 setup keep-state
# ipfw -q add allow tcp from any to 192.0.2.11 53 out via tun0 setup keep-state
# ipfw -q add 00611 allow udp from any to 192.0.2.11 53 out via tun0 keep-state
```

30.6.5.5. Állapottartó szabályrendszerek

A most következő címfordítás nélküli szabályrendszer arra mutat példát, hogyan valósítsunk meg egy biztonságos "inkluzív" tűzfalat. Az inkluzív tűzfalak csak a szabályainak megfelelő szolgáltatásokat engedik át, minden más alapértelmezés szerint tiltanak. A komplett hálózati szegmensek védelmére összeállított tűzfalaknak legalább két interfészük van, amelyek mindegyikéhez tartoznia kell szabályoknak a megfelelő működéshez.

Az UNIX mintájú operációs rendszer, köztül a FreeBSD is olyan, hogy a rendszerben belüli kommunikációt a `lo0` nevű interfészen és a `127.0.0.1` IP-címen bonyolítja le. A tűzfalban mindenképpen szerepelniük kell olyan szabályoknak, amelyek gondoskodnak ezen speciális belső csomagok zavartalan közlekedéséről.

Az internet felé csatlakozó interfész lesz az, amelyen keresztül a kifelé menő kéréseket hitelesítjük és vezéreljük az internet elérését, valamint ahol szűrjük az internet felől érkező kéréseket. Ez lehet a PPP esetében a `tun0` eszköz, vagy a DSL-, illetve kábelmodemhez csatlakozó hálózati kártya.

Abban az esetben, amikor egy vagy több hálózati kártyával csatlakozunk a tűzfal mögött található belső helyi hálózatra, szintén gondoskodnunk kell a helyi hálózaton belül mozgó csomagok akadálymentes továbbításáról.

A szabályokat először három nagyobb osztályba kell sorolnunk: az összes szabadon forgalmazó interfész, a publikus kimenő és a publikus bejövő interfész csoportjába.

A publikus interfészekhez tartozó csoportokban úgy kell rendeznünk a szabályokat, hogy előre kerüljenek a gyakrabban használtak és hátra a kevésbé használtak, valamint a csoportok utolsó szabálya blokkoljon és naplózzon minden csomagot az adott interfészen és irányban.

A következő szabályrendszerben szereplő, a kimenő kapcsolatokat tartalmazó csoport csak olyan `allow` típusú szabályokat tartalmaz, amelyek szűrési feltételei egyértelműen azonosítják az interneten elérhető szolgáltatásokat. Az összes szabályban megjelennek a `proto`, `port`, `in/out`, `via` és `keep state` opciók. A `proto tcp` szabályokban emellett szerepel még egy `setup` opció is, amellyel a kapcsolatokat kezdeményező csomagokat tudjuk azonosítani és felvenni az állapottartásért felelős dinamikus szabályok közé.

A bejövő forgalmat vezérlő szabályrendszerben először az eldobni kívánt csomagokat kell megadni, aminek két eltérő oka van. Először is előfordulhat, hogy a veszélyes csomagok részleges illeszkedés miatt szabályosnak tűnnek. Az ilyen csomagokat értelemszerűen nem lenne szabad beengedni a szabályok részleges megfelelése alapján. A másodszor az eleve ismert problémás és értelmetlen csomagokat csendben el kellene vetni, mielőtt a szakaszhoz tartozó utolsó szabály fogná meg és naplózná. Ez az utolsó szabály egyébként szükség esetén felhasználható a támadók elleni bizonyítékok begyűjtésére.

A másik, amire még oda kell figyelnünk, hogy a blokkolt csomagok esetében semmilyen válasz nem keletkezzen, egyszerűen csak tűnjenek el. Így a támadó nem fogja tudni, hogy a csomagjai vajon elérték-e a rendszerünket. Minél kevesebb információt tudnak összegyűjteni a rendszerünkről a támadók, annál biztonságosabbnak tekinthető. Amikor ismeretlen portokra érkező csomagokat naplózunk, érdemes az `/etc/services/` állományban vagy <http://www.securitystats.com/tools/portsearch.php> címen (angolul) utánanézni a porthoz tartozó szolgáltatásnak. A különböző trójai programok által portok számai ezen a linken érhetőek el (angolul): <http://www.simovits.com/trojans/trojans.html>.

30.6.5.6. Példa egy inkluzív szabályrendszerre

A most következő, címfordítást nem tartalmazó szabályrendszer teljesen inkluzív típusú. Éles rendszereken is nyugodtan alkalmazhatjuk. Egyszerűen csak annyit kell tennünk, hogy megjegyzésbe tesszük az olyan szolgáltatásokra vonatkozó szabályokat, amelyeket nem akarunk engedélyezni. Amikor pedig olyan üzenetek jelennek meg a naplóban, amelyeket nem akarunk tovább látni, a bejövő kapcsolatokhoz vegyünk fel egy `deny`

típusú szabályt hozzájuk. Minden szabályban cseréljük ki a `dc0` interfészt arra a hálózati kártyára, amely közvetlenül csatlakoztatja rendszerünket az internethez. A felhasználói PPP esetében ez a `tun0`.

A szabályok használatában felfedezhetünk egyfajta rendszerszerűséget:

- Mindegyik sorban, ahol az internet felé nyitunk meg egy kapcsolatot, a `keep-state` opciót használjuk.
- Az internetről az összes hitelesített szolgáltatás elérése tartalmazza a `limit` opciót az elárasztások kivédése miatt.
- Az összes szabályban az `in` vagy az `out` paraméterrel megadjuk szűrni kívánt forgalom irányát.
- Az összes szabályban szerepel a `via` paraméterrel a csomagokat továbbító interfész neve.

Az alábbi szabályokat tegyük az `/etc/ipfw.rules` állományba.

```
##### Itt kezdődnek az IPFW szabályai #####
# Kezdés előtt töröljük az összes aktív szabályt.
ipfw -q -f flush

# Állítsuk be a parancsok további szükséges opciót.
cmd="ipfw -q add"
pif="dc0"      # az internethez csatlakozó
               # interfész neve

#####
# A belső hálózat számára ne korlátozzunk semmit se.
# Ha nincs helyi hálózatunk, akkor erre nincs szükségünk.
# Az 'xl0' nevét írjuk át a helyi hálózatra csatlakozó
# interfész nevére.
#####
$cmd 00005 allow all from any to any via xl0

#####
# A rendszer belső interfészét se szűrjük.
#####
$cmd 00010 allow all from any to any via lo0

#####
# A csomagot engedjük át a tűzfalon, ha korábban már felvettünk
# hozzá egy dinamikus szabályt a keep-state opcióval.
#####
$cmd 00015 check-state

#####
# Az internet felé forgalmazó interfész (kimenő kapcsolatok)
# A saját hálózatunkról belülről vagy erről az átjáróról
# kezdeményezett kapcsolatokat vizsgáljuk az internet felé.
#####

# Kifelé engedélyezzük az internet-szolgáltatónk névszerverének
# elérését. Az x.x.x.x a szolgáltatónk névszerverének IP-címe
# legyen. Ha a szolgáltatónak több névszervere is van, akkor
# másoljuk le ezeket a sorokat és az /etc/resolv.conf
# állományban található IP-címeket helyettesítsük be.
$cmd 00110 allow tcp from any to x.x.x.x 53 out via $pif setup keep-state
```

```

$cmd 00111 allow udp from any to x.x.x.x 53 out via $pif keep-state

# Kábel/DSL konfigurációk esetében kifelé engedélyezzük a
# szolgáltatónk DHCP szerverének elérését. Ha a "felhasználói
# PPP"-t használjuk, akkor erre nem lesz szükségünk, az egész
# csoportot törölhetjük. Az alábbi szabállyal csíphetjük el a
# beírandó IP-címet. Ha a naplóban megtaláltuk, akkor vegyük
# ki az első szabályt, a másodikba írjuk bele a címet és
# engedélyezzük.
$cmd 00120 allow log udp from any to any 67 out via $pif keep-state
#$cmd 00120 allow udp from any to x.x.x.x 67 out via $pif keep-state

# Kifelé engedélyezzük a szabvány nem biztonságos WWW
# funkció elérését.
$cmd 00200 allow tcp from any to any 80 out via $pif setup keep-state

# Kifelé engedélyezzük a biztonságos HTTPS funkció
# elérését TLS SSL használatával.
$cmd 00220 allow tcp from any to any 443 out via $pif setup keep-state

# Kifelé engedélyezzük a e-mailek küldését és fogadását.
$cmd 00230 allow tcp from any to any 25 out via $pif setup keep-state
$cmd 00231 allow tcp from any to any 110 out via $pif setup keep-state

# Kifelé engedélyezzük a FreeBSD (a make install és a CVSUP)
# funkcióit. Ezzel lényegében a rendszeradminisztrátornak
# „ISTENI” jogokat adunk.
$cmd 00240 allow tcp from me to any out via $pif setup keep-state uid root

# Kifelé engedélyezzük a pinget.
$cmd 00250 allow icmp from any to any out via $pif keep-state

# Kifelé engedélyezzük az idő szolgáltatást.
$cmd 00260 allow tcp from any to any 37 out via $pif setup keep-state

# Kifelé engedélyezzük az nntp news szolgáltatást
# (vagyis a hírcsoportokat)
$cmd 00270 allow tcp from any to any 119 out via $pif setup keep-state

# Kifelé engedélyezzük a biztonságos FTP, telnet és SCP
# elérését az SSH (secure shell) használatával.
$cmd 00280 allow tcp from any to any 22 out via $pif setup keep-state

# Kifelé engedélyezzük a whois szolgáltatást.
$cmd 00290 allow tcp from any to any 43 out via $pif setup keep-state

# Dobjuk el és naplózzunk mindent, ami megpróbál kijutni.
# Ez a szabály gondoskodik róla, hogy alapértelmezés szerint
# mindent blokkoljunk.
$cmd 00299 deny log all from any to any out via $pif

#####
# Az internet felőli interfész (bejövő kapcsolatok)

```

```
# A saját hálózatunk felé vagy erre az átjáróra
# nyitott kapcsolatokat vizsgáljuk az internet felől.
#####

# Blokkoljunk minden olyan bejövő forgalmat, amely a fenntartott
# címtartományok felé tart.
$cmd 00300 deny all from 192.168.0.0/16 to any in via $pif #RFC 1918: privát IP
$cmd 00301 deny all from 172.16.0.0/12 to any in via $pif #RFC 1918: privát IP
$cmd 00302 deny all from 10.0.0.0/8 to any in via $pif #RFC 1918: privát IP
$cmd 00303 deny all from 127.0.0.0/8 to any in via $pif #helyi
$cmd 00304 deny all from 0.0.0.0/8 to any in via $pif #helyi
$cmd 00305 deny all from 169.254.0.0/16 to any in via $pif #DHCP
$cmd 00306 deny all from 192.0.2.0/24 to any in via $pif #dokumentációs célokra fenntartott
$cmd 00307 deny all from 204.152.64.0/23 to any in via $pif #Sun klaszterek összekötésére használ
$cmd 00308 deny all from 224.0.0.0/3 to any in via $pif #D és E osztályú multicast

# A nyilvános pingek tiltása.
$cmd 00310 deny icmp from any to any in via $pif

# Az ident szolgáltatás tiltása.
$cmd 00315 deny tcp from any to any 113 in via $pif

# Blokkoljuk az összes Netbios szolgáltatást: 137=név, 138=datagram,
# 139=session. A Netbios az MS Windows megosztását implementálja.
# Blokkoljuk az MS Windows hosts2 névszerver kéréseit is a 81-es
# porton.
$cmd 00320 deny tcp from any to any 137 in via $pif
$cmd 00321 deny tcp from any to any 138 in via $pif
$cmd 00322 deny tcp from any to any 139 in via $pif
$cmd 00323 deny tcp from any to any 81 in via $pif

# Eldobjuk az összes későn érkező csomagot.
$cmd 00330 deny all from any to any frag in via $pif

# Eldobjuk azokat az ACK csomagokat, amelyek egyik dinamikus
# szabálynak sem felelnek meg.
$cmd 00332 deny tcp from any to any established in via $pif

# Befelé engedélyezzük a szolgáltató DHCP szerverének válaszát. Ebben
# a szabályban csak a DHCP szerver IP-címe szerepelhet, mivel ez az
# egyetlen olyan hitelesített forrás, ami ilyen csomagokat küldhet.
# Ez csak a kábeles és DSL típusú kapcsolatok esetében szükséges.
# Amikor a "felhasználói PPP"-vel csatlakozunk az internethez, nem
# kell ez a szabály. Ugyanazt az IP-címet kell megadnunk, amelyet a
# kimenő kapcsolatoknál is.
#$cmd 00360 allow udp from any to x.x.x.x 67 in via $pif keep-state

# Befelé engedélyezzük a szabvány WWW funkciót, mivel webszerverünk
# is van.
$cmd 00400 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Befelé engedélyezzük a biztonságos FTP, telnet és SCP
# típusú kapcsolatokat az internetről.
```

```
$cmd 00410 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Befelé engedélyezzük az internetről érkező nem biztonságos telnet
# kapcsolatokat. Azért tekintjük nem biztonságosnak, mert az
# azonosítók és a jelszavak az interneten titkosítatlanul vándorolnak.
# Töröljük ezt a csoportot, ha nincs telnet szolgáltatásunk.
$cmd 00420 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Dobjuk el és naplózzuk az összes többi kintről érkező csomagot.
$cmd 00499 deny log all from any to any in via $pif

# Alapértelmezés szerint dobjuk el mindent. Az ide érkező
# csomagokat is naplózzuk, amiből többet is ki tudunk majd
# deríteni.
$cmd 00999 deny log all from any to any
##### Itt fejeződnek be az IPFW szabályai #####
```

30.6.5.7. Példa hálózati címfordításra és állapottartásra

Az IPFW címfordító funkciójának kihasználásához további konfigurációs beállítások alkalmazására is szükségünk lesz. A rendszermagban opció között meg kell adnunk az `option IPDIVER`t sort a többi `IPFIREWALL` sor mellett, és fordítanunk egy saját verziót.

Emellett még az `/etc/rc.conf` állományban is engedélyezni kell az IPFW alapvető funkcióit.

```
natd_enable="YES"                # engedélyezzük a címfordításért felelős demont
natd_interface="rl0"             # az internet felé mutató hálózati kártya neve
natd_flags="-dynamic -m"        # -m = a portszámok megtartása, ha lehetséges
```

Az állapottartó szabályok használata a `divert natd` címfordítási opcióval együtt nagyban növeli a szabályrendszer leprogramozásának bonyolultságát. A `check-state` és `divert natd` szabályok helye kritikus a megfelelő működés tekintetében. Az eddig megszokott egyszerű viselkedés itt már nem érvényesül. Bevezetünk egy új cselekvést is, amelynek a neve `skipto`. A `skipto` parancs használatához elengedhetetlen a szabályok sorszámozása, mivel pontosan tudnunk kell, hogy a `skipto` hatására hova kell ugrania a vezérlésnek.

A következő példában nem fogunk sok megjegyzést látni, mivel benne az egyik lehetséges programozási stílust próbáljuk érzékeltetni és a csomagok szabályrendszerek közti áramlását magyarázzuk.

A feldolgozás a szabályokat tartalmazó állomány tetején található első szabállyal kezdődik, és innen egyesével pereg végig lefelé a feldolgozás egészen addig, amíg a csomag a szűrési feltételek valamelyikének eleget nem tesz és távozik a tűzfalból. Leginkább a 100-as, 101-es, 450-es, 500-as és 510-es sorszámú szabályokat emelnénk ki. Ezek vezérlik kimenő és bejövő csomagok fordítását, ezért a hozzájuk tartozó dinamikus állapottartó bejegyzések mindig a helyi hálózat IP-címeire hivatkoznak. Amit még érdemes megfigyelnünk, hogy az összes áteresztő és eldobó szabályban szerepel a csomag haladási iránya (tehát kimenő vagy éppen bejövő) és az érintett interfészt megnevezése. Emellett azt is vegyük észre, hogy az összes kifelé irányuló kapcsolatlétrehozási kérés az 500-as sorszámú szabályhoz fog ugrani a címfordítás elvégzéséhez.

Tegyük fel, hogy a helyi hálózatunkon levő felhasználók szeretnek honlapokat nézgetni az interneten. A honlapok a 80-as porton keresztül kommunikálnak. Tehát amikor egy ilyen csomag eléri a tűzfalat, nem fog illeszkedni a 100-as szabályra, mert a fejléce szerint kifelé halad és nem befelé. A 101-es szabályon is átlép, mivel ez az első csomag, így a dinamikus állapottartó táblázatban sem szerepel még. A csomag végül a 125-ös szabályra fog illeszkedni: kifelé halad az internetre csatlakozó hálózati kártyán. A csomagban azonban még mindig az eredeti forrás IP-címe

található, amely a helyi hálózat egyik gépére hivatkozik. A szabály illeszkedésekor két cselekvés is végbemeget. A `keep-state` opció hatására ez a szabály felveszi ezt a kapcsolatot az állapottartó dinamikus szabályok közé és végrehajtja a másik megadott feladatot. Ez a feladat része a dinamikus táblázatba rögzített bejegyzésnek, ami ebben az esetben a `skipto 500` ("ugorjunk az 500-as szabályra") lesz. Az 500-as szabály a továbbküldés előtt lefordítja a csomag forrás IP-címét. Ezt ne felejtjük el, nagyon fontos! A csomag ezután eljut a céljához, és visszatérve ismét belép a szabályrendszer tetején. Ezúttal illeszkedni fog a 100-as szabályra és a cél IP-címét visszafordítjuk a helyi hálózatunk megfelelő gépének címére. Ezután a `check-state` szabályhoz kerül, amely megtalálja a dinamikus szabályok között és továbbengedi a belső hálózatra. Ezzel visszakerül a küldő géphez, amely egy újabb csomagot küld egy újabb adatszeletet kérve a távoli szervertől. Ekkor már a `check-state` szabály megtalálja a hozzá tartozó bejegyzést a dinamikus szabályok között és végrehajtódik a korábban letárolt `skipto 500` művelet. A csomag erre az 500-as szabályra ugrik, ahol lefordítjuk a címét és továbbküldjük.

Az bejövő oldalon minden, ami egy korábban kialakult kapcsolat részeként érkezik, automatikusan a `check-state` és a megfelelő helyre rakott `divert natd` szabályok által dolgozódik fel. Itt mindössze a rossz csomagok eldobásával és a hitelesített szolgáltatások elérésének biztosításával kell foglalkoznunk. Például a tűzfalon egy webszerver fut, és azt szeretnénk, hogy az internetről képesek legyenek elérni a rajta levő oldalakat. Az újonnan beérkező kapcsolatépítési kérelem a 100-as szabályra fog illeszkedni, amelynek a cél IP-címét a tűzfal helyi hálózaton található címére fogjuk leképezni. A csomagot ezután még megvizsgáljuk, nem tartalmaz-e valamilyen huncutságot, majd végül a 425-ös szabálynál fog kikötni. Az egyezéskor két dolog történhet: a csomaghoz felveszünk egy dinamikus szabályt, de ezúttal az adott forrás IP-címről érkező kapcsolatkéresek számát 2-re lekorlátozzuk. Ezzel az adott szolgáltatás portján meg tudjuk óvni a tűzfalat üzemeltető gépet a DoS típusú támadásoktól. A csomagot ezután hozzá tartozó cselekvés szerint továbbengedjük a belső hálózat felé. Visszatéréskor a tűzfal felismeri, hogy a csomag egy már meglevő kapcsolathoz tartozik, ezért közvetlenül az 500-as szabályhoz kerül címfordításra, majd a kimenő interfészen keresztül továbbküldjük.

Íme az első példa egy ilyen szabályrendszerre:

```
#!/bin/sh
cmd="ipfw -q add"
skip="skipto 500"
pif=r10
ks="keep-state"
good_tcpo="22,25,37,43,53,80,443,110,119"

ipfw -q -f flush

$cmd 002 allow all from any to any via x10 # nem szűrjük a belső hálózatot
$cmd 003 allow all from any to any via lo0 # nem szűrjük a helyi interfészt

$cmd 100 divert natd ip from any to any in via $pif
$cmd 101 check-state

# A kimenő csomagok hitelesítése:
$cmd 120 $skip udp from any to xx.168.240.2 53 out via $pif $ks
$cmd 121 $skip udp from any to xx.168.240.5 53 out via $pif $ks
$cmd 125 $skip tcp from any to any $good_tcpo out via $pif setup $ks
$cmd 130 $skip icmp from any to any out via $pif $ks
$cmd 135 $skip udp from any to any 123 out via $pif $ks

# Az összes olyan csomagot eldobjuk, amely a fenntartott
# címtartományokba tart:
```

```
$cmd 300 deny all from 192.168.0.0/16 to any in via $pif #RFC 1918: privát IP
$cmd 301 deny all from 172.16.0.0/12 to any in via $pif #RFC 1918: privát IP
$cmd 302 deny all from 10.0.0.0/8 to any in via $pif #RFC 1918: privát IP
$cmd 303 deny all from 127.0.0.0/8 to any in via $pif #helyi
$cmd 304 deny all from 0.0.0.0/8 to any in via $pif #helyi
$cmd 305 deny all from 169.254.0.0/16 to any in via $pif #DHCP
$cmd 306 deny all from 192.0.2.0/24 to any in via $pif #dokumentációs célokra fenntartott
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif #Sun klaszter
$cmd 308 deny all from 224.0.0.0/3 to any in via $pif #D és E osztályú multicast
```

Az érkező csomagok hitelesítése:

```
$cmd 400 allow udp from xx.70.207.54 to any 68 in $ks
$cmd 420 allow tcp from any to me 80 in via $pif setup limit src-addr 1
```

```
$cmd 450 deny log ip from any to any
```

Ide ugrunk a kimenő állapottartó szabályoknál:

```
$cmd 500 divert natd ip from any to any out via $pif
$cmd 510 allow ip from any to any
```

a szabályok vége

A következő példa teljesen megegyezik az előzővel, azonban itt már dokumentációs szándékkal szerepelnek megjegyzések is, melyek a tapasztalatlan IPFW szabályíróknak segítik jobban megérteni a szabályok pontos működését.

A második példa:

```
#!/bin/sh
##### Az IPFW szabályai itt kezdődnek #####
# Kezds előtt töröljük az összes jelenleg aktív szabályt:
ipfw -q -f flush

# Beállítjuk a parancsok megfelelő előtagjait:
cmd="ipfw -q add"
skip="skipto 800"
pif="rl0" # az internethez csatlakozó
          # hálózati interfész neve

#####
# A belső hálózat számára ne korlátozzunk semmit se.
# Ha nincs helyi hálózatunk, akkor erre nincs szükségünk.
# Az 'xl0' nevét írjuk át a helyi hálózatra csatlakozó
# interfész nevére.
#####
$cmd 005 allow all from any to any via xl0

#####
# A rendszer belső interfészét se szűrjük.
#####
$cmd 010 allow all from any to any via lo0
```

```
#####
# Ellenőrizzük, hogy ez egy beérkező csomag és ha igen, akkor
# fordítsuk a címét.
#####
$cmd 014 divert natd ip from any to any in via $pif

#####
# Ha ehhez a csomaghoz korábban már vettük fel dinamikus
# szabályt a keep-state opció révén, akkor engedjük tovább.
#####
$cmd 015 check-state

#####
# Az internet felé forgalmazó interfész (kimenő kapcsolatok)
# A saját hálózatunkról belülről vagy erről az átjáróról
# kezdeményezett kapcsolatokat vizsgáljuk az internet felé.
#####

# Kifelé engedélyezzük az internet-szolgáltatónk névszerverének
# elérését. Az x.x.x.x a szolgáltató névszerverének IP-címe
# lesz. Ha a szolgáltatóknak több névszervere is van, akkor
# az /etc/resolv.conf állományból nézzük ki a címeiket és
# másoljuk le az alábbi sor mindegyikükhöz.
$cmd 020 $skip tcp from any to x.x.x.x 53 out via $pif setup keep-state

# A kábeles és DSL kapcsolatok esetén engedélyezzük a szolgáltató
# DHCP szerverének elérését.
$cmd 030 $skip udp from any to x.x.x.x 67 out via $pif keep-state

# Kifelé engedélyezzük a szabvány nem biztonságos WWW funkciót
$cmd 040 $skip tcp from any to any 80 out via $pif setup keep-state

# Kifelé engedélyezzük a biztonságos HTTPS funkciót a TLS SSL
# használatával.
$cmd 050 $skip tcp from any to any 443 out via $pif setup keep-state

# Kifelé engedélyezzük az e-mailek küldését és fogadását.
$cmd 060 $skip tcp from any to any 25 out via $pif setup keep-state
$cmd 061 $skip tcp from any to any 110 out via $pif setup keep-state

# Kifelé engedélyezzük a FreeBSD (make install és CVSUP) funkcióit.
# Ezzel a rendszeradminisztrátornak „ISTENI” jogokat adunk.
$cmd 070 $skip tcp from me to any out via $pif setup keep-state uid root

# Kifelé engedélyezzük a pinget.
$cmd 080 $skip icmp from any to any out via $pif keep-state

# Kifelé engedélyezzük az idő szolgáltatást.
$cmd 090 $skip tcp from any to any 37 out via $pif setup keep-state

# Kifelé engedélyezzük az nntp news szolgáltatást (tehát a
# hírcsoportokat).
```

```

$cmd 100 $skip tcp from any to any 119 out via $pif setup keep-state

# Kifelé engedélyezzük a biztonságos FTP, telnet és SCP
# funkciókat az SSH (secure shell) használatával.
$cmd 110 $skip tcp from any to any 22 out via $pif setup keep-state

# Kifelé engedélyezzük ki a whois kéréseket.
$cmd 120 $skip tcp from any to any 43 out via $pif setup keep-state

# Kifelé engedélyezzük az NTP időszerver elérését.
$cmd 130 $skip udp from any to any 123 out via $pif keep-state

#####
# Az internet felőli interfész (bejövő kapcsolatok)
# A saját hálózatunk felé vagy erre az átjáróra
# nyitott kapcsolatokat vizsgáljuk az internet felől.
#####

# Tiltuk a fenntartott címtartományok felé haladó összes beérkező
# forgalmat.
$cmd 300 deny all from 192.168.0.0/16 to any in via $pif #RFC 1918: privát IP
$cmd 301 deny all from 172.16.0.0/12 to any in via $pif #RFC 1918: privát IP
$cmd 302 deny all from 10.0.0.0/8 to any in via $pif #RFC 1918: privát IP
$cmd 303 deny all from 127.0.0.0/8 to any in via $pif #helyi
$cmd 304 deny all from 0.0.0.0/8 to any in via $pif #helyi
$cmd 305 deny all from 169.254.0.0/16 to any in via $pif #DHCP
$cmd 306 deny all from 192.0.2.0/24 to any in via $pif #dokumentációs célokra fenntartott
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif #Sun klaszter
$cmd 308 deny all from 224.0.0.0/3 to any in via $pif #D és E osztályú multicast

# Az ident tiltása.
$cmd 315 deny tcp from any to any 113 in via $pif

# Blokkoljuk az összes Netbios szolgáltatást: 137=név, 138=datagram,
# 139=session. A Netbios az MS Windows megosztását implementálja.
# Blokkoljuk az MS Windows hosts2 névszerver kéréseit is a 81-es
# porton.
$cmd 320 deny tcp from any to any 137 in via $pif
$cmd 321 deny tcp from any to any 138 in via $pif
$cmd 322 deny tcp from any to any 139 in via $pif
$cmd 323 deny tcp from any to any 81 in via $pif

# Dobjuk el a későn érkező csomagokat.
$cmd 330 deny all from any to any frag in via $pif

# Dobjuk el azokat az ACK csomagokat, amelyekre nincs
# dinamikus szabály.
$cmd 332 deny tcp from any to any established in via $pif

# Engedélyezzük a szolgáltató DHCP szerverétől érkező forgalmat. Ennek
# a szabálynak tartalmaznia kell a DHCP szerver címét, mert csak tőle
# fogadunk el ilyen típusú csomagokat. Egyedül csak kábeles vagy DSL
# konfigurációk esetén használatos, a "felhasználói PPP" esetében

```

```
# törölhetjük. Ez ugyanaz az IP-cím, amelyet a kimenő kapcsolatoknál
# megadtunk.
$cmd 360 allow udp from x.x.x.x to any 68 in via $pif keep-state

# Befelé engedélyezzük a szabvány WWW funkciót, mivel van
# webszerverünk.
$cmd 370 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Befelé engedélyezzük a biztonságos FTP, telnet és SCP
# használatát az internetről.
$cmd 380 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Befelé engedélyezzük a nem biztonságos telnet elérését az
# internetről. Azért nem tekintjük biztonságosnak, mert az
# azonosítókat és a jelszavakat az interneten titkosítatlanul
# közvetíti. Ha nincs telnet szolgáltatásunk, akkor törölhetjük is ezt
# a csoportot.
$cmd 390 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Dobjuk el és naplózzuk az összes internetről érkező hitelesítetlen kapcsolatot.
$cmd 400 deny log all from any to any in via $pif

# Dobjuk el és naplózzuk az összes internetre menő hitelesítetlen kapcsolatot.
$cmd 450 deny log all from any to any out via $pif

# Ez lesz a kimenő szabályokhoz tartozó "skipto" célja.
$cmd 800 divert natd ip from any to any out via $pif
$cmd 801 allow ip from any to any

# Minden mást alapértelmezés szerint tiltunk és naplózunk.
$cmd 999 deny log all from any to any
##### Az IPFW szabályai itt fejeződnek be #####
```

Fejezet 31. Egyéb haladó hálózati témák

31.1. Áttekintés

Ebben a fejezetben számos komolyabb hálózati témát fogunk tárgyalni.

A fejezet elolvasása során megismerjük:

- az átjárók és az útválasztás alapjait;
- hogyan állítsunk be IEEE® 802.11 és Bluetooth eszközöket;
- a FreeBSD segítségével hogyan tudunk két hálózatot összekötni hálózati hidakon keresztül;
- hogyan indítsuk hálózatról egy lemez nélküli gépet;
- hogyan állítsunk be hálózati címfordítást;
- hogyan kapcsoljunk össze két számítógépet PLIP használatával;
- hogyan állítsuk be az IPv6 használatát egy FreeBSD-s gépen
- hogyan állítsuk be az ATM használatát;
- hogyan engedélyezzük és használjuk a Közös címredundancia protokollt FreeBSD-ben.

A fejezet elolvasásához ajánlott:

- az `/etc/rc` könyvtárban található szkriptek működésének ismerete;
- az alapvető hálózati fogalmak ismerete;
- egy új FreeBSD rendszermag beállításának és telepítésének ismerete (8 fejezet);
- a külső szoftverek telepítésének ismerete (4 fejezet).

31.2. Átjárók és az útválasztás

Készítette: Coranth Gryphon.

Egy gép egy másikat úgy tud megtalálni a hálózaton, ha erre létezik egy olyan mechanizmus, amely leírja, hogyan tudunk eljutni az egyiktől a másikig. Ezt hívjuk *útválasztásnak* (routing). Az “útvonal” (route) címek egy párjaként adható meg, egy “céllal” (destination) és egy “átjáróval” (gateway). Ez a páros mondja meg, hogy ha el akarjuk érni ezt a *célt*, akkor ezen az *átjárón* keresztül kell továbbhaladnunk. A céloknak három típusa lehet: egyéni gépek, alhálózatok és az “alapértelmezett”. Az “alapértelmezett útvonalat” (default route) abban az esetben alkalmazzuk, ha semelyik más útvonal nem megfelelő. Az alapértelmezett útvonalokról a későbbiekben még beszélni fogunk. Három típusa van az átjáróknak: egyéni gépek, felületek (avagy “linkek”) és a hardveres Ethernet címek (MAC-címek).

31.2.1. Példa

Az útválasztás különböző területeit a következő `netstat` parancs alapján fogjuk bemutatni:

```
% netstat -r
Routing tables
```

Destination	Gateway	Flags	Refs	Use	Netif	Expire
default	outside-gw	UGSc	37	418	ppp0	
localhost	localhost	UH	0	181	lo0	
test0	0:e0:b5:36:cf:4f	UHLW	5	63288	ed0	77
10.20.30.255	link#1	UHLW	1	2421		
example.com	link#1	UC	0	0		
host1	0:e0:a8:37:8:1e	UHLW	3	4601	lo0	
host2	0:e0:a8:37:8:1e	UHLW	0	5	lo0 =>	
host2.example.com	link#1	UC	0	0		
224	link#1	UC	0	0		

Az első két sorban az alapértelmezett útvonalat (melyről részleteiben majd a következő szakaszban fogunk szólni) és a localhost útvonalát láthatjuk.

A localhost címhez az útválasztási táblázatban a lo0 eszköz tartozik (a Netif oszlopban), amelyet loopback eszköznek is neveznek. Ez arra utasítja a rendszert, hogy az ide küldött csomagokat ne a helyi hálózaton küldje keresztül, hanem csak ezen a belső felületen, mivel úgyis oda jutnának vissza, ahonnan indultak.

A táblázatban a következő sor egy 0:e0 kezdetű címet tartalmaz. Ez egy hardveres Ethernet cím, más néven MAC-cím. A FreeBSD magától képes beazonosítani tetszőleges gépet (ebben a példában a test0 gépet) a helyi Ethernetes hálózaton és felvenni hozzá egy útvonalat, közvetlenül az ed0 Ethernetes csatolófelületen keresztül. Ehhez a típusú útvonalhoz tartozik még egy lejáratí idő is (a Expire oszlop), amely akkor kap szerepet, ha ennyi idő elteltével nem kapunk semmilyen hírt a gépről. Amikor ilyen történik, az géphez eddig nyilvántartott útvonal automatikusan törlődik. Ezek a gépek a RIP (útvonal-információs protokoll, Routing Information Protocol) nevű mechanizmuson keresztül azonosítódnak, mely a legrövidebb út kiszámítása alapján határozza meg a helyi gépekhez vezető útvonalat.

A FreeBSD a helyi alhálózat (10.20.30.255 és example.com, az alhálózathoz tartozó név) esetében is felvesz útvonalakat. A link#1 megnevezés a gépben található első Ethernet-kártyát jelöli. Megfigyelhetjük, hogy rajta kívül nincs is több felülete.

Mindegyik csoport (a helyi hálózati gépek és a helyi alhálózatok) útvonalait a **routed** nevű démon tartja automatikusan karban. Ha ez nem fut, akkor csak a statikusan definiált (vagyis az előre megadott) útvonalak fognak létezni.

A host1 sor a saját gépünkre vonatkozik, amelyet az Ethernet címe szerint ismerünk. Mivel mi vagyunk küldő gép, a FreeBSD tudni fogja, hogy ilyenkor az Ethernetes felület helyett a loopback eszközt (lo0) kell használnia.

A két host2 sor arra mutat példát, amikor az ifconfig(8) paranccsal álneveket hozunk létre (ennek konkrét okait lásd az Ethernetről szóló részben). A lo0 felület neve után szereplő => szimbólum azt jelzi, hogy ez nem csak egy loopback felület (mivel a címe szintén a helyi gépre mutat), hanem a felület egy másik neve. Ilyen útvonalak csak az álneveket ismerő gépeknél jelennek meg. A helyi hálózaton minden más gépnél egyszerűen csak a link#1 jelenik meg az ilyen útvonalak esetében.

Az utolsó sor (a 224 céllal rendelkező alhálózat) a multicastre (többesküldésre) szolgál, amellyel majd egy másik szakaszban foglalkozunk.

Végezetül az útvonalakhoz tartozó különféle tulajdonságok a Flags oszlopban láthatóak. Az alábbi rövid táblázatban összefoglaltunk közülük néhányat:

U	Up: az útvonal aktív
H	Host: az útvonal egyetlen gépre mutat

G	Gateway: az adott cél felé ezen a gépen keresztül küldjük, amely majd kitalálja, hogy merre küldje tovább
S	Static: ez az útvonal statikus, nem a rendszer hozta létre automatikusan
C	Clone: ebből az útvonalból származtatunk új útvonalat azokhoz a gépekhez, amelyekhez csatlakozunk. Ilyen útvonalakat általában a helyi hálózatokban találhatunk
W	WasCloned: azt jelzi, hogy ezt az útvonalat egy helyi hálózatra mutató (klón, avagy Clone típusú) útvonal alapján hoztuk létre automatikusan
L	Link: az útvonal Ethernetes hardverhez kapcsolódik

31.2.2. Alapértelmezett útvonalak

Amikor a helyi rendszernek fel kell vennie a kapcsolatot egy távoli géppel, ellenőrzi az útválasztási táblázatban, hogy létezik-e már hozzá valamilyen útvonal. Ha a távoli gép egy olyan alhálózatba esik, amelyet már el tudunk érni (klónozott útvonalak), akkor a rendszer megnézi, hogy a hozzá tartozó felületen képes-e kapcsolatot létesíteni.

Ha minden ismert útvonal csődöt mond, akkor a rendszerünknek marad még egy utolsó esélye: az “alapértelmezett” útvonal használata. Ez az útvonal egy speciális átjáró útvonal (ebből általában csak egyetlen egy létezik a rendszerben) és tulajdonságai között mindig szerepel a c. A helyi hálózat gépei közül ez az átjáró az legyen, amelyik közvetlenül kapcsolódik a külső világhoz (PPP összeköttetéssel, DSL, kábelmodem, T1 vagy bármilyen más hálózati felületen keresztül).

Amikor pedig magát a külső világ felé átjáróként szolgáló gépet állítjuk be, az alapértelmezett útvonal az internet-szolgáltatónk által megadott gép címe lesz.

Vegyünk egy példát az alapértelmezett útvonalakra. Egy tipikus konfiguráció:



A Helyi1 és Helyi2 gépek a hálózatunk tagjai. A Helyi1 az internet-szolgáltatót éri el egy betárcsázós PPP kapcsolaton keresztül. A PPP szerver a külső felületén keresztül a helyi hálózaton pedig egy másik átjáróhoz csatlakozik.

Az egyes gépek alapértelmezett útvonalai így alakulnak:

Gép	Alapértelmezett átjáró	Felület
Helyi2	Helyi1	Ethernet
Helyi1	T1-ÁJ	PPP

Gyakran felmerül a kérdés, hogy “Miért (és hogy-hogy) a T1-ÁJ a Helyi1 gép számára az alapértelmezett átjáró és nem a szolgáltató azon szervere, amelyhez csatlakozott?”

Ne felejtjük el, hogy a PPP felület a szolgáltató helyi hálózatában a mi részünkre kap címet, és a itt az összes többi géphez tartozó útvonal automatikusan létrejön. Emiatt már eleve el tudjuk érni a T1-ÁJ gépet, ezért amikor a szolgáltatón keresztül küldünk, nincs szükségünk egy további lépcsőre.

Általában a x.x.x.1 címet szokták a helyi hálózat átjárójának kiosztani. Ezért (az előbbi példát újrahasznosítva) ha a helyi hálózatunkon a C osztályú 10.20.30 címtartományt használjuk, és a szolgáltatónkhoz a 10.9.9 címtartomány tartozik, akkor az alapértelmezett útvonalak a következők lesznek:

Gép	Alapértelmezett útvonal
Helyi2 (10.20.30.2)	Helyi1 (10.20.30.1)
Helyi1 (10.20.30.1, 10.9.9.30)	T1-ÁJ (10.9.9.1)

Az `/etc/rc.conf` állományon keresztül könnyen meg tudjuk adni az alapértelmezett útvonalat. A példánkban a Helyi2 gép `/etc/rc.conf` állományába kell felvennünk a következő sort:

```
defaultrouter="10.20.30.1"
```

A `route(8)` parancs használatával viszont akár közvetlenül is megtehetjük mindezt:

```
# route add default 10.20.30.1
```

A `route(8)` man oldalon olvashatunk arról bővebben, hogy a hálózati útválasztási táblázatokat kézzel hogyan tudjuk módosítani.

31.2.3. Kettős hálózati gépek

Egy másik típusú konfigurációról is szót kell ejtenünk, ahol a gép egyszerre két hálózatnak is tagja. Gyakorlatilag az átjáróként üzemelő számítógépek (mint például az, amelyik a fenti példában PPP kapcsolattal csatlakozott) ilyen kettős hálózati gépek tekinthetők. Ez a kifejezés azonban igazából csak azokra az esetekre illik, ahol a gép egyszerre két helyi hálózatban is megjelenik.

Az egyik esetben a gépben két Ethernet kártya található, melyek mindegyike birtokol egy-egy hálózati címet az egyes alhálózatokon. De előfordulhat az is, hogy a gépünkben csupán egyetlen Ethernet kártya van és az `ifconfig(8)` segítségével álneveket hoztunk létre hozzá. Az előbbi általában két fizikailag elkülönülő Ethernet alapú hálózat esetében történik, míg az utóbbinál csak egyetlen fizikai hálózati szegmensről van szó, amely viszont logikailag két külön alhálózatot tartalmaz.

Akarmelyiket is vesszük, az útválasztási táblázatok úgy jönnek létre, hogy bennük a gép a másik alhálózat felé átjáróként (bejövő útvonalként) lesz nyilvántartva. Ebben a konfigurációban a gép a két alhálózat között útválasztóként fog tevékenykedni, és gyakran valamelyik vagy éppen mind a két irányba be kell állítanunk valamilyen csomagszűrést vagy tűzfalazást.

Ha azt szeretnénk, hogy ez a gép a két felület között továbbítsa csomagokat, akkor a FreeBSD-ben külön engedélyezni kell ezt a lehetőséget. A következő szakaszban ennek részleteit tárjuk fel.

31.2.4. Az útválasztók beállítása

A hálózati útválasztó nem csinál mást, csak továbbküldi az egyik felületén beérkező csomagokat egy másik felületére. Az internetes szabványok és a sokéves mérnöki tapasztalat azonban nem engedik, hogy a FreeBSD Projekt alapértelmezés szerint is elérhetővé tegye ezt a FreeBSD rendszerekben. Ezt a lehetőséget az alábbi változó `YES` értékűre állításával lehet engedélyezni az `rc.conf(5)` állományban:

```
gateway_enable="YES"           # Ez legyen YES, ha átjáróként akarunk üzemelni
```

Ezzel lényegében a `net.inet.ip.forwarding sysctl(8)` változó értékét állítjuk 1-re. Ha valamiért egy időre szüneteltetni akarjuk a csomagok továbbküldését, akkor állítsuk a változó értékét 0-ra.

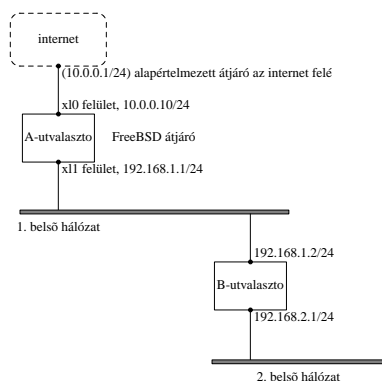
Az új útválasztónak nem árt arról sem tudnia, hogy merre továbbítsa a forgalmat. Ha elég egyszerű a hálózatunk, akkor akár statikus útvonalakat is használhatunk. A FreeBSD alpból tartalmazza a BSD-k esetén szabványos routed(8) útválasztó démont, amely a RIP (v1 és v2) valamint az IRDP megoldásokat ismeri. A BGP v4, OSPF v2 és a többi fejlettebb útválasztási protokoll a net/zebra csomagban érhető el. Az ettől bonyolultabb hálózati útválasztási feladatokhoz olyan kereskedelmi termékek is elérhetőek, mint például a **GateD®**.

31.2.5. Statikus útvonalak beállítása

Írta: Al Hoang.

31.2.5.1. Manuális konfiguráció

Tegyük fel, hogy hálózatunk a következő:



Ebben a forgatókönyvben az A-utvalaszo a mi FreeBSD-s gépünk, amely az internet felé vezető útválasztó szerepét játssza. Számára az alapértelmezett útvonal a 10.0.0.1, amelyen keresztül a külső világot tudja elérni. Feltételezzük, hogy a B-utvalaszo nevű gépet már eleve jól állítottuk be, ezért tudja merre kell mennie. (A kép alapján egyszerű: csak vegyünk fel egy alapértelmezett útvonalat a B-utvalaszo gépéhez, ahol így a 192.168.1.1 lesz az átjáró.)

Ha megnézzük most az A-utvalaszo útválasztási táblázatát, akkor nagyjából a következőket fogjuk látni:

```
% netstat -nr
Routing tables
```

```
Internet:
Destination      Gateway          Flags    Refs      Use  Netif  Expire
default          10.0.0.1        UGS      0         49378   x10
127.0.0.1        127.0.0.1       UH       0          6     lo0
10.0.0/24        link#1          UC       0          0     x10
192.168.1/24     link#2          UC       0          0     x11
```

Az A-utvalaszo útválasztási táblázata alapján jelen helyzetben nem lehet elérni a 2. belső hálózatot. Nincs ugyanis olyan útvonal, amely a 192.168.2.0/24 alhálózat felé vezetne. Ezt például úgy tudjuk megoldani, ha manuálisan felvesszük ezt az útvonalat. Az alábbi paranccsal hozzáadjuk a 2. belső hálózat elérését az A-utvalaszo útválasztási táblázatához, ahol a 192.168.1.2 lesz a következő ugrási pont (next hop):

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Most már az A-utvalasztó bármelyik gépet képes elérni a 192.168.2.0/24 hálózaton.

31.2.5.2. Rögzített konfiguráció

A fenti példa tökéletesen szemlélteti a statikus útvonalak felvételét egy működő rendszeren. Azonban ezzel az a gond, hogy az így megadott útválasztási információ nem marad meg a gép újraindítása után. Ezért az előbbihez hasonló statikus útvonalakat inkább az `/etc/rc.conf` állományban rögzítsük:

```
# A 2. belső hálózat elérését felvesszük statikus útvonalként
static_routes="belsohalo2"
route_belsohalo2="-net 192.168.2.0/24 192.168.1.2"
```

A `static_routes` konfigurációs változó karakterláncok szóközzel tagolt felsorolását tartalmazza. Mindegyik karakterlánc egy útvonal neve. Az iménti példában csak egyetlen ilyen név szerepelt a `static_routes` értékében, amely a `belsohalo2` volt. Utána beírtunk még egy konfigurációs változót is, amelynek a neve `route_belsohalo2`. Ide helyeztük a `route(8)` parancsnak átadandó beállítás összes paraméterét. Ez pontosan olyan, mintha a következő parancsot adtuk volna ki:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Ezért kellett a `"-net 192.168.2.0/24 192.168.1.2"`.

Ahogy már korábban is említettük, a `static_routes` értékében több karakterláncot is megadhatunk, aminek segítségével egyszerre több statikus útvonalat is létrehozhatunk. A következő sorok arra mutatnak példát, hogy a 192.168.0.0/24 és 192.168.1.0/24 hálózatok számára miként állítsunk be statikus útvonalakat a képzeletbeli útválasztónkon:

```
static_routes="net1 net2"
route_net1="-net 192.168.0.0/24 192.168.0.1"
route_net2="-net 192.168.1.0/24 192.168.1.1"
```

31.2.6. Az útvonalak terjedése

Azt már tudjuk, hogyan adjuk meg a külvilág felé vezető útvonalakat, azonban arról még nem beszéltünk, hogy kívülről miként találják meg bennünket.

Annyit már megismertünk, hogy az útválasztási táblázatokban megadhatjuk a hálózaton azt a gépet, amelyen keresztül az adott címtartomány (a példában egy C osztályú alhálózat) felé küldhetünk, amely pedig továbbküldi a hozzá érkező csomagokat.

Amikor a csatlakozunk az internet-szolgáltatónkhoz, a nála levő útválasztási táblázatok úgy állítódnak be, hogy az alhálózatunk felé igyekvő adatok a korábban létrejött PPP összeköttetésen keresztül jutnak el hozzánk. A világ többi részén levő rendszerek viszont honnan fogják tudni, hogy a mi internet-szolgáltatónknak küldjenek?

Van egy rendszer (ez leginkább a névszerverek elosztott információs adatbázisához hasonlít), ami nyilvántartja a pillanatnyilag kiosztott címtartományokat és megadja a csatlakozási pontjukat az internet gerinchálózatán. Ez a "gerinc" tulajdonképpen olyan fővonalakból áll, amelyen keresztül a világban az országok között mozog az internet forgalma. A gerinchálózat mindegyik gépe tárolja a központi útválasztási táblázatok egy másolatát, ami a forgalmat egy adott hálózatról a megadott gerincbeli hordozóra irányítja át, végig az internet-szolgáltatók láncán egészen addig, amíg az el nem éri a hálózatunkat.

A szolgáltatónk feladata, hogy a gépünk felé leágazásként (és így a felénk vezető útként) beregisztálja magát a gerinchálózat gépein. Ezt nevezik az útvonal terjedésének.

31.2.7. Hibaelhárítás

Néha gondok lehetnek az útvonal terjedésével, és egyes gépek nem képesek elérni minket. A `traceroute(8)` parancs mind közül talán az egyik leghasznosabb ilyen helyzetekben, mivel ezzel fel tudjuk deríteni, hogy az útválasztás hol akad meg. Ugyanilyen jól hasznosítható azokban az esetekben, amikor látszólag nem tudunk elérni egy távoli gépet (tehát a `ping(8)` csődöt mond).

A `traceroute(8)` parancsnak annak a távoli gépnek a nevét kell megadnunk, amelyhez csatlakozni akarunk. Futása közben megjeleníti azokat az átjárókat, amelyeken keresztül csatlakozni próbál, akár sikerült elérni a célgépet, akár a kapcsolat hiánya miatt kudarcot vall.

A parancs használatáról és működéséről részletesebb információkat a `traceroute(8)` man oldalán találunk.

31.2.8. Útválasztás multicast esetén

A FreeBSD alapból támogatja mind a multicastet használó alkalmazásokat, mind pedig a multicasthez tartozó útválasztást. Multicast esetében semmilyen speciális beállítás nem szükséges, az ilyen alkalmazások egyből el tudják érni ezt a lehetőséget. A multicast kérések útválasztásához azonban be kell építenünk némi támogatást a rendszermagba:

```
options MROUTING
```

Emellett még el kell indítanunk az `mROUTED(8)` démon is, amelyhez az `/etc/mROUTED.conf` állományban még be kell állítanunk tunneleket és a DVMRP használatát. A multicasthez tartozó további beállításokat az `mROUTED(8)` man oldalán találhatjuk.

Megjegyzés: A FreeBSD 7.0 megjelenésével a `mROUTED(8)` démon kivették az alaprendszerből. Azt a DVMRP többesküldési protokollt valósítja meg, amelyet a legtöbb alkalmazásban mostanság már a `pim(4)` segítségével oldanak meg. Ennek megfelelően a hozzá tartozó multicast protokollt valósítja meg, amelyet a legtöbb alkalmazásban mostanság már a `pim(4)` segítségével oldanak meg. Ennek megfelelően a hozzá tartozó `map-mbone(8)` és `mrinfo(8)` segédprogramok is eltávolításra kerültek. Ezek a programok attól a kiadástól kezdődően a Portgyűjtemény részeként érhetőek el a `net/mROUTED` portban.

31.3. Vezeték nélküli hálózatok

Loader, Marc Fonvieille, és Murray Stokely.

31.3.1. A vezeték nélküli hálózatok alapjai

A legtöbb vezeték nélküli hálózat az IEEE 802.11 szabványon nyugszik. Az alapvető vezeték nélküli hálózatokban több olyan állomást találhatunk, amelyek egymással rádiójelek szórásával kommunikálnak a 2,4 GHz vagy 5 GHz

frekvenciatartományban (noha ez a helyi viszonyoknak megfelelően változhat, és a 2,3 GHz, illetve a 4,9 GHz tartományokban is lehetséges a kommunikáció).

A 802.11 szabványú hálózatok kétféleképpen szerveződnek. Először is *infrastrukturálisan*, (infrastructural mode) ahol az egyik állomást kinevezzük a központnak és a többi pedig ehhez fog tartozni. Az ilyen hálózatokat BSS-nek nevezzük és az imént említett központ neve hozzáférési pont (Access Point, AP) lesz. A BSS-ben az összes kommunikáció a hozzáférési pontokon keresztül halad még abban az esetben is, amikor az egyik állomás egy másik vezeték nélküli állomással akarja felvenni a kapcsolatot. Az ilyen jellegű hálózatok másik típusú szerveződési módjában nincsenek kijelölt központok és a kommunikáció az állomások között közvetlenül zajlik. A hálózat ezen formáját IBBS-nek nevezzük, vagy ismeretebb nevén *ad-hoc hálózatnak* (ad-hoc network).

A 802.11 alapú hálózatok elsőként a 2,4 GHz-es sávot hódították meg, és az IEEE 802.11 valamint 802.11b szabványokban rögzített protokollokat használták. Ezekben a specifikációkban megtalálhatjuk a működési frekvenciát, a közeghozzáférési réteg jellemzőinek leírását, beleértve a keretezést és az átviteli sebességeket (a kommunikáció ugyanis eltérő sebességekkel is történhet). A később kiadott 802.11a szabvány azt specifikálja, hogy az 5 GHz-es tartományban miként működjenek, ahol többek közt megtalálhatjuk a különféle jelkezelési mechanizmusokat és a nagyobb átviteli sebességek használatát. Ezt még a 802.11g szabvány követte, ami a 802.11b hálózatokkal kompatibilis módon lehetővé tette a 802.11a jelkezelésének és átviteli módszereinek használatát a 2,4 GHz-es sávban.

A 802.11 alapú hálózatok mindenféle átviteli technikáitól eltekintve többféle biztonsági megoldással találkozhatunk. Az korai 802.11 dokumentumok egy nagyon egyszerű biztonsági protokollt, a WEP-et említene. Ez a protokoll a hálózaton mozgó adatokat egy rögzített és ismert osztott kulccsal kódolja le az RC4 titkosítással. A kommunikációhoz az összes állomásnak előre meg kell egyeznie ebben a kulcsban. Erről a sémáról időközben kiderült, hogy könnyen feltörhető és manapság már csak nagyon ritkán alkalmazzák, kivéve talán csak a kóbor felhasználók elijesztésére. A jelenleg érvényes biztonsági előírásokat az IEEE 802.11i specifikáció adja meg, amely új kriptográfiai titkosításokat definiál valamint egy további protokollt az állomások azonosítására és a kulcsok cseréjére. Emellett a titkosításhoz használt kulcsok időszakosan frissülnek és külön eszközök állnak rendelkezésre a betörési kísérletek észlelésére (és azok elhárítására). A vezeték nélküli hálózatok esetében másik elterjedt titkosítási protokoll a WPA. Ez igazából 802.11i elődjének tekinthető, amelyet egy ipari csoport definiált, amíg a 802.11i minősítés alatt állt. A WPA ennek megfelelően teljesíti a 802.11i szabvány elvárásainak egy részét és kifejezetten a régi hardverek számára készült. A WPA működéséhez egyedül a TKIP titkosításra van szükségünk, amely az eredeti WEP titkosításból származik. A 802.11i engedi a TKIP használatát, de az adatok kódolására egy erősebb titkosítás, az AES-CCM ismeretét is igényli. (Az AES a WPA esetében nem kell, mivel a régi eszközök esetében túlságosan költségesnek ítélték meg a használatát.)

A fenti szabványokon kívül a 802.11e a másik fontos szabvány, amire tekintettel kell lennünk. Ez írja le a 802.11 hálózatokon a multimédiás alkalmazások közvetítéséhez, mint például a videók valós idejű lejátszásához vagy a VoIP (voice over IP) megvalósításához tartozó protokollokat. A 802.11i szabványhoz hasonlóan a 802.11e is magában foglal egy előzetes specifikációt, amelyet WME (később pedig már WMM)-nek neveznek. Ezt szintén egy ipari csoport definiálta a 802.11e részeként, amivel a 802.11e végső elfogadásáig tudják a multimédiás igényeket kiszolgálni. Amit a 802.11e és WME/WMM megoldásaival kapcsolatban érdemes tudnunk: a QoS (Quality of Service) protokoll és más egyéb fejlett közeghozzáférési protokollok segítségével a vezeték nélküli hálózatokban lehetővé teszik a forgalom prioritás szerinti ütemezését. Ezen protokollok megfelelő implementációjának segítségével tehát a fontosabb adatok nagy sebességgű küldését és áramoltatását vagyunk képesek elérni.

A FreeBSD a 6.0 verzió óta ismeri a 802.11a, 802.11b és 802.11g szabványokon alapján működő hálózatokat. A WPA és 802.11i biztonsági protokollok (a 11a, 11b és 11g szabványok bármelyike esetén) hasonlóképpen támogatottak, valamint a WME/WMM protokollok működéséhez szükséges QoS csak bizonyos vezeték nélküli eszközök esetében.

31.3.2. Kezdeti beállítások

31.3.2.1. A rendszermag beállítása

A vezeték nélküli hálózatok használatához egy vezeték nélküli hálózati kártyára lesz szükségünk, valamint a rendszermagban is be kell állítani ehhez a megfelelő támogatást. Ez utóbbit több különböző modulra szedték szét, és ezek közül csak azokat kell beállítani, amelyeket tényleg használni is fogunk.

Először is tehát kell egy vezeték nélküli eszköz. Az elterjedtebb típusaik általában az Atheos által gyártott alkatrészeket tartalmazzák. Az ilyen fajtájú eszközöket az ath(4) meghajtó kezeli, melyet úgy tudunk a rendszer indításakor betölteni, ha a `/boot/loader.conf` állományba felvesszük a következő sort:

```
if_ath_load="YES"
```

Az Atheos meghajtója három különálló részre oszlik: maga a meghajtó (ath(4)), a hardveres réteg, ami a chipfüggő funkciókat kezeli (ath_hal(4)) és a keretek küldésével kapcsolatban az átviteli sebesség megválasztását lehetővé tevő algoritmus (ez itt most az ath_rate_sample). Amikor ezt a támogatást modulként töltjük be, ezek a függőségek automatikusan feloldódnak. Ha az Atheos eszközök helyett valamelyik másikkhoz tartozó modult szeretnénk használni, akkor például az Intersil Prism esetében a wi(4) meghajtót kell megadnunk:

```
if_wi_load="YES"
```

Megjegyzés: A leírás további részeiben az ath(4) eszközt fogjuk használni, minden más esetben ennek a nevét kell csak lecserélünk a példákban. A rendszerben elérhető vezeték nélküli meghajtók és az általuk támogatott kártyák listája a FreeBSD Hardverjegyzetekben található. Ezek a jegyzetek a különböző architektúrákra és kiadásokhoz a FreeBSD honlapjáról, a Kiadási jegyzetek (<http://www.FreeBSD.org/releases/index.html>) oldalról érhetőek el. Ha a vezeték nélküli eszközünkhöz nem létezik natív FreeBSD-s meghajtó, akkor az NDIS meghajtó segítségével akár közvetlenül a Windows-os meghajtóját is használhatjuk.

FreeBSD 7.x esetén az eszközmeghajtó beállításával együtt a 802.11 hálózatok támogatását is be kell töltenünk a rendszermagba. Ez az ath(4) meghajtó esetében a legalább a wlan(4), wlan_scan_ap és wlan_scan_sta modulok betöltését jelenti. A wlan(4) modul a vezeték nélküli eszköz meghajtóprogramjával együtt töltődik be, míg a többi modult a `/boot/loader.conf` állomány használatával kell a rendszerindítás során betöltenünk:

```
wlan_scan_ap_load="YES"
wlan_scan_sta_load="YES"
```

A FreeBSD 8.0 kiadástól kezdődően ezek a modulok részei a wlan(4) meghajtónak, amely a hálózati kártya meghajtójával együtt mindig automatikusan betöltődik.

Emellett még azokra a modulokra is szükségünk van, amelyek a használni kívánt biztonsági protokollokhoz nyújtanak kriptográfiai támogatást. Ezek hivatalosan a wlan(4) modul kérésére automatikusan betöltődnek, azonban itt most manuálisan állítjuk be. Erre a célra a következő modulokat találjuk: wlan_wep(4), wlan_ccmp(4) és wlan_tkip(4). A wlan_ccmp(4) és wlan_tkip(4) meghajtók csak akkor fognak kelleni, ha a WPA és/vagy a 802.11i biztonsági protokollokat használjuk. Amennyiben a hálózatunkon nincs titkosítás, akkor még a wlan_wep(4) támogatás sem kell. Ezeket a modulok úgy lehet betölteni a rendszerindításnál, ha felvesszük a következő sorokat a `/boot/loader.conf` állományba:

```
wlan_wep_load="YES"
wlan_ccmp_load="YES"
```

```
wlan_tkip_load="YES"
```

Miután ezt megcsináltuk, egyszerűen csak indítsuk újra a gépünket. Ha még nem akarjuk újraindítani a gépet, akkor a `kldload(8)` parancs segítségével akár kézzel is betölthetjük az előbb felsorolt modulokat.

Megjegyzés: Ha nem akarunk modulokat használni, a működéshez szükséges meghajtókat a rendszermagba is be tudjuk építeni a következő sorok megadásával a rendszermag beállításait tartalmazó állományban:

```
device wlan          # a 802.11 támogatása
device wlan_wep      # 802.11 WEP támogatás
device wlan_ccmp     # 802.11 CCMP támogatás
device wlan_tkip     # 802.11 TKIP támogatás
device wlan_amrr     # AMRR forgalomvezérlési algoritmus
device ath           # Atheros IEEE 802.11 vezeték nélküli hálózati meghajtó
device ath_hal       # az Atheros meghajtó hardveres rétege
options AH_SUPPORT_AR5416 # az AR5416 tx/rx leírók engedélyezése
device ath_rate_sample # SampleRate forgalomvezérlési algoritmus
```

Hozzáteesszük, hogy az alábbi sorok hozzáadása a FreeBSD 7.x változatában kötelező, más verzióknál viszont nem:

```
device wlan_scan_ap      # a 802.11 AP módú keresés
device wlan_scan_sta     # a 802.11 STA módú keresés
```

Az előbbieket megadásával fordítsuk újra és telepítsük a rendszermagot, majd indítsuk újra a számítógépünket.

Miután a rendszerünk újra elindult, a rendszer indítás során generált üzenetei között találunk kell valamennyi információt a felismert vezeték nélküli eszközökről. Például:

```
ath0: <Atheros 5212> mem 0x88000000-0x8800ffff irq 11 at device 0.0 on cardbus1
ath0: [ITHREAD]
ath0: AR2413 mac 7.9 RF2413 phy 4.5
```

31.3.3. Az infrastruktúrális működési mód

Általában az infrastruktúrális avagy a BSS mód használata a gyakori. Ebben a működési módban adott számú vezeték nélküli hozzáférési pont csatlakozik a hagyományos hálózatra. Mindegyik vezeték nélküli hálózatnak saját neve van, amit a hálózat SSID-jének hívunk. A vezeték nélküli kliensek ezekhez a vezeték nélküli hozzáférési pontokhoz kapcsolódnak.

31.3.3.1. A FreeBSD-s kliensek használata

31.3.3.1.1. Hogyan keressünk hozzáférési pontokat

A hálózatok kereséséhez az `ifconfig` parancsal tudunk nekifogni. Egy ilyen kérés kiszolgálása eltarthat néhány pillanatig, mivel ekkor a rendszernek végig kell bóklásznia az összes elérhető frekvenciát és azokon hozzáférési pontok után kutatni. Egyedül a rendszeradminisztrátor kezdeményezheti ezeket a kereséseket:

```
# ifconfig wlan0 create wlandev ath0
```

```
# ifconfig wlan0 up scan
SSID/MESH ID      BSSID              CHAN  RATE  S:N      INT  CAPS
dlinkap           00:13:46:49:41:76  11    54M   -90:96   100  EPS   WPA WME
freebsdap         00:11:95:c3:0d:ac   1     54M   -83:96   100  EPS   WPA
```

Megjegyzés: Csak `up` jelzésű felületen tudunk hálózatokat keresni. További keresésekre már nincs szükség a felület `up` állapotban tartásához.

Megjegyzés: FreeBSD 7.x esetén a `wlan0` eszköz helyett közvetlenül az adott eszköz nevét kell megadnunk, például `ath0`. Az iménti sorokat ennek megfelelően tehát ebben az esetben így kell értelmezni:

```
# ifconfig ath0 up scan
```

A leírás további részében a FreeBSD 7.x felhasználóknak ezen séma alapján kell használniuk a parancsokat és a konfigurációs beállításokat.

A keresés során keletkező listában láthatjuk megtalált BBS vagy IBBS fajtájú hálózatokat. A hálózatok neve és SSID-ja mellett még megjelenik egy BSSID oszlop is, ahol a hozzáférési pontok MAC-címe szerepel. A CAPS oszlop az egyes állomások tulajdonságait adja meg:

E

Extended Service Set (ESS): az állomás egy infrastrukturális vagyis BBS hálózat része.

I

IBSS/ad-hoc hálózat: az állomás egy ad-hoc hálózat része.

P

Privacy: a BBS-en belül minden keretet titkosítani kell. Tehát a BSS arra kötelezi az állomást, hogy WEP, TKIP vagy AES-CCMP titkosítás használatával kódolja a hálózat tagjai között közlekedő kereteket.

S

Short Preamble: a hálózatban rövid bevezetőjeleket használnak (a 802.11b High Rate/DSSS PHY előírásai szerint), ahol a szokványos 128 bites szinkronizációs mező hossza csak 56 bit.

S

Short Slot Time: a 802.11g hálózat rövid slotidőt használ, mivel nem találhatók benne régi (802.11b szabványú) állomások.

A jelenleg ismert hálózatok listáját így tudjuk lekérdezni:

```
# ifconfig wlan0 list scan
```

Ezt az információt maga az adapter automatikusan, vagy a felhasználó tudja frissíteni a `scan` kérés kiadásával. Az elavult adatok maguktól törölődnek a gyorsítótárból, így idővel a lista zsugorodni fog, hacsak nem keresünk folyamatosan hálózatokat.

31.3.3.1.2. Alapvető beállítások

Ebben a szakaszban arra mutatunk példákat, hogy miként tudunk FreeBSD alatt titkosítás nélkül használni egy vezeték nélküli hálózati kártyát. Miután elsajátítottuk az itt szereplő ismereteket, határozottan javasoljuk, hogy a vezeték nélküli hálózatunkat WPA használatával állítsuk be.

A vezeték nélküli hálózatok beállítása három elemi lépésből épül fel: a hozzáférési pont kiválasztása, az állomásunk hitelesítése és az IP-cím beállítása. A következőkben ezeket a lépéseket vitatjuk meg.

31.3.3.1.2.1. A hozzáférési pont kiválasztása

A legtöbb esetben hagyjuk, hogy a rendszer válassza ki magának a különböző heurisztikák alapján a leginkább megfelelő hozzáférési pontot. Ez az alapértelmezett tevékenység, amikor aktiváljuk a felületet vagy valamilyen más módon, például az `/etc/rc.conf` állományból hivatkozunk rá:

```
wlans_ath0="wlan0"
ifconfig_wlan0="DHCP"
```

Megjegyzés: A korábban említettek szerint a FreeBSD 7.x felhasználóknak csak a kártyát kell beállítani:

```
ifconfig_ath0="DHCP"
```

Ha viszont több hozzáférési pont közül mi magunk akarunk kiválasztani egyet, akkor ezt az SSID megadásával tehetjük meg:

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid saját_ssid DHCP"
```

Amikor olyan környezetben vagyunk, ahol több hozzáférési pontnak is megegyezik az SSID-ja (gyakran így próbálják egyszerűsíteni azt, hogy automatikusan váltani lehessen köztük), akkor szükségünk lehet ezt egy adott eszközhöz hozzárendelni. Ebben az esetben a hozzáférési pont BSSID-ját is definiálni kell (és az SSID-t akár el is hagyhatjuk):

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid saját_ssid bssid xx:xx:xx:xx:xx:xx DHCP"
```

Más módokon is képesek vagyunk szabályozni a hozzáférési pontok megválasztását, például a rendszerünk által vizsgált frekvenciasávok megadásával. Ez olyankor tud hasznos lenni, ha többsávós vezeték nélküli kártyánk van, és az összes tartomány végigpásztázása túlságosan sok időt venne el. Ezt a művelet a `mode` paraméter megadásával lehet egy konkrét sávra leszűkíteni, például a

```
wlans_ath0="wlan0"
ifconfig_wlan0="mode 11g ssid saját_ssid DHCP"
```

beállítás hatására a kártya 802.11g módban fog üzemelni, ami kizárólag csak 2,4 GHz-es frekvenciákon használható, így az 5 GHz-es csatornákat egyszerűen figyelmen kívül hagyjuk. Ugyanezt a `channel` paraméterrel is meg tudjuk oldani, mivel így a működést egy adott frekvenciára korlátozzuk, valamint a `chanlist` paraméterrel, ahol a pásztázandó csatornákat sorolhatjuk fel. Ezekről a paraméterekről részletesebb leírást az `ifconfig(8)` man oldalon találhatunk.

31.3.3.1.2.2. Hitelesítés

Miután sikeresen kiválasztottuk a számunkra megfelelő hozzáférési pontot, az adatok küldéséhez az állomásunknak valamilyen módon hitelesítenie kell magát. A hitelesítés több módon történhet. Erre a leggyakrabban alkalmazott sémát nyílt hitelesítésnek (open authentication) nevezik, ahol a hálózathoz tetszőleges állomás csatlakozhat és kommunikálhat vele. Ezt a típusú hitelesítést akkor érdemes használni, amikor a vezeték nélküli hálózatunkat teszteljük. Más sémákban az adatfolyam megindításához egy titkosítási kézfogás szükséges, vagy előre megosztott kulcsok esetleg jelszavak segítségével, vagy bonyolultabb sémák esetében itt még olyan különböző háttérszolgáltatások is megjelennek, mint például a RADIUS. A legtöbb felhasználó a nyílt hitelesítést használja, ami egyben az alapértelmezés is. A másik legelterjedtebb beállítás a WPA-PSK, avagy WPA Personal, amelyről lentebb még szólni fogunk.

Megjegyzés: Ha Apple AirPort® Extreme Base Station típusú hozzáférési pontunk van, akkor az osztott kulcsú hitelesítés mellett egy WEP kulcsot is be állítanunk. Ezt az `/etc/rc.conf` állományban vagy a `wpa_supplicant(8)` programban tehetjük meg. Ha egyetlen AirPort bázisállomásunk van, akkor az elérést valahogy így tudjuk beállítani:

```
wlans_ath0="wlan0"
ifconfig_wlan0="authmode shared wepmode on weptxkey 1 wepkey 01234567 DHCP"
```

Általánosságban véve elmondhatjuk, hogy az osztott kulcsú hitelesítést inkább kerüljük el, mivel WEP kulcsok használatára alapszik és ráadásul olyan módon, hogy nagyon könnyű feltörni. Ha már mindenképpen a WEP mellett kell döntenünk (például a régebbi eszközökkel így tudunk csak kompatibilisek maradni), akkor jobban járunk, ha a nyílt hitelesítéshez alkalmazzuk. A WEP használatát érintő további információkat a 31.3.3.1.4 Szakaszban találjuk.

31.3.3.1.2.3. IP-cím szerzése DHCP használatával

Miután kiválasztottunk egy hozzáférési pontot és beállítottuk a hitelesítés paramétereit, egy IP-cím is kelleni fog a kommunikációhoz. Az esetek túlnyomó részében DHCP-n keresztül kapunk IP-címet a vezeték nélküli kapcsolatunkhoz. Ezt úgy érhetjük el, ha egyszerűen megnyitjuk az `/etc/rc.conf` állományt és az alábbihoz hasonló módon felvesszük a DHCP paramétert az eszközünk beállításaihoz:

```
wlans_ath0="DHCP"
ifconfig_wlan0="DHCP"
```

Így már készen is állunk a vezeték nélküli felület használatára:

```
# /etc/rc.d/netif start
```

Ahogy a felület működőképpé válik, az `ifconfig` parancs segítségével ellenőrizni is tudjuk az `ath0` felület állapotát:

```
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.1.100 netmask 0xffffffff broadcast 192.168.1.255
    media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
    status: associated
    ssid dlinkap channel 11 (2462 Mhz 11g) bssid 00:13:46:49:41:76
```

```
country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
roam:rate 5 protmode CTS wme burst
```

A `status: associated` azt jelenti, hogy sikeresen csatlakoztunk egy vezeték nélküli hálózathoz (jelen esetben ez a `dlinkap`). A `bssid 00:13:46:49:41:76` rész a hozzáférési pont MAC-címét tartalmazza. Az `authmode OPEN` pedig arról számol be, hogy a kommunikáció nem titkosított.

31.3.3.1.2.4. Statikus IP-cím

Ha valami okból nem tudjuk az IP-címünket DHCP szerveren keresztül lekérni, beállíthatunk rögzített IP-címet is. Ehhez nem kell mást tennünk, mint a korábban bemutatott DHCP kulcsszót kicserélni egy konkrét címmel. A hozzáférési ponthoz megadott többi paramétert azonban feltétlenül hagyjuk meg:

```
wlans_ath0="wlan0"
ifconfig_wlan0="inet 192.168.1.100 netmask 255.255.255.0 ssid saját_ssid"
```

31.3.3.1.3. WPA

A WPA (Wi-Fi Protected Access, vagyis védett wi-fi hozzáférés) a 802.11 szabványokban használatos biztonsági protokoll, amelyet a WEP gyengeségeinek és megfelelő hitelesítésének ellensúlyozására dolgoztak ki. A WPA a 802.1X hitelesítési protokolljait erősíti és az adat sértetlenségének megőrzésére a WEP helyett több titkosítási algoritmust is felhasznál. A WPA által igényelt egyetlen titkosítás a TKIP (Temporary Key Integrity Protocol, vagyis az ideiglenes kulcs integritási protokoll), amely a WEP által az integritás ellenőrzésére és a bejutások észlelésére és azok reagálására szánt alap RC4 titkosítást bővíti ki. A TKIP a régebbi hardvereken csupán szoftveres módosítással működőképesse tehető. Ez a kompromisszum a védelmet ugyan növeli, de még mindig kevés a támadások megfelelő elhárításához. A WPA a TKIP mellett tartalmazza még az AES-CCMP titkosítást is, és ennek a használata javasolt. Ezt a specifikációt gyakran WPA2 (vagy RSN) néven emlegetik.

A WPA definiál hitelesítési és titkosítási protokollokat. A hitelesítés általában a következő két technika egyike alapján történik: vagy 802.1X és egy háttérszolgáltatás, például a RADIUS segítségével, vagy egy előre megosztott kulcsot alkalmazó minimális kézfogással az állomás és a hozzáférési pont között. Az előbbit gyakran WPA Enterprise-nak, míg az utóbbit WPA Personalnak hívják. Mivel a legtöbben nem állítanak be egy komplett RADIUS alapú szervert a vezeték nélküli hálózatukhoz, ezért a WPA-PSK a WPA leginkább elterjedten használt változata.

A vezeték nélküli kapcsolat és a hitelesítés (kulcs alapján vagy szerverrel) vezérlését a `wpa_supplicant(8)` segédprogram végzi. Ennek a programnak működéséhez egy konfigurációs állományra van szüksége, amely az `/etc/wpa_supplicant.conf` néven érhető el. Erről az állományról bővebb információt a `wpa_supplicant.conf(5)` man oldalán lelhetünk.

31.3.3.1.3.1. WPA-PSK

A WPA-PSK, más néven WPA-Personal, egy adott jelszó alapján generált előre megosztott kulcssal (pre-shared key, PSK) működik, amit a vezeték nélküli hálózatokban mesterkulcsként használnak. Ez azt jelenti, hogy minden egyes vezeték nélküli felhasználó ugyanazon a kulcson osztozik. A WPA-PSK olyan kis méretű hálózatok esetében megfelelő, ahol a hitelesítést elvégző szerver használata nem lehetséges vagy nem oldható meg.

Figyelem Mindig igyekezzünk erős jelszavakat használni, melyek kellően hosszúak és sokféle karaktert tartalmaznak, és így nehezebben fejthetők meg vagy törhetőek fel.

Először az `/etc/wpa_supplicant.conf` állományban állítsuk be az SSID-t és a hálózatunkhoz tartozó előre megosztott kulcsot:

```
network={
    ssid="freebsdap"
    psk="freebsdmail"
}
```

Ezután az `/etc/rc.conf` állományban jelezzük, hogy a vezeték nélküli eszközt a WPA segítségével állítjuk be és az IP-címet a DHCP szervertől kérjük el:

```
wlans_ath0="wlan0"
ifconfig_ath0="WPA DHCP"
```

Innentől már fel is tudjuk éleszteni a felületet:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 5
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 6
DHCPOFFER from 192.168.0.1
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.254 netmask 0xfffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet autoselect OFDM/36Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

Kézzel is megpróbálhatjuk elindítani az előbb elkészített `/etc/wpa_supplicant.conf` állomány használatával:

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:11:95:c3:0d:ac (SSID='freebsdap' freq=2412 MHz)
Associated with 00:11:95:c3:0d:ac
WPA: Key negotiation completed with 00:11:95:c3:0d:ac [PTK=CCMP GTK=CCMP]
CTRL-EVENT-CONNECTED - Connection to 00:11:95:c3:0d:ac completed (auth) [id=0 id_str=]
```

A következő parancs a `dhclient` indítása legyen, amivel megszerezünk a DHCP szervertől az IP-címünket:

```
# dhclient wlan0
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
```

```
# ifconfig wlan0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.254 netmask 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

Megjegyzés: Ha az `/etc/rc.conf` állományban szerepel a `ifconfig_wlan0="DHCP"` sor, akkor egyáltalán nem szükséges a `dhclient` parancs manuális kiadása, mivel a `dhclient` magától el fog indulni, miután a `wpa_supplicant` egyeztetette a kulcsokat.

Amikor a DHCP nem használható, megadhatunk a statikus IP-címet is, miután a `wpa_supplicant` sikeresen lebonyolította a hitelesítést:

```
# ifconfig wlan0 inet 192.168.0.100 netmask 255.255.255.0
# ifconfig wlan0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.100 netmask 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

Ha egyáltalán nem használunk DHCP szervert, akkor nekünk kell beállítani az alapértelmezett átjárót és a névszervert is:

```
# route add default alapértelmezett_átjáró
# echo "nameserver névszerver" >> /etc/resolv.conf
```

31.3.3.1.3.2. WPA és EAP-TLS

A másik mód, ahogy a WPA használható, az a 802.1X hitelesítési szerveren keresztül történik, és ebben az esetben a WPA neve WPA-Enterprise. Ez sokkal biztonságosabb a WPA-Personal előre kiosztott kulcsaival szemben. A WPA-Enterprise az EAP (Extensible Authentication Protocol, azaz Bővíthető hitelesítési protokoll) használatán alapszik.

Az EAP önmaga nem végez titkosítást, mivel úgy alakították ki, hogy magát az EAP protokollt kell egy titkosított járaton keresztül bújtatni. Az EAP hitelesítési módszereinek több típusát is kidolgozták, melyek közül a legismertebbek az EAP-TLS, EAP-TTLS valamint a EAP-PEAP.

Az EAP-TLS (EAP szállítási rétegbeli védelemmel) a vezeték nélküli világban egy nagyon jól támogatott hitelesítési protokoll, mivel ez volt az első EAP módszer, amit a Wi-fi szövetség (<http://www.wi-fi.org/>) jóváhagyott. Az EAP-TLS működéséhez három tanúsítvány kell: egy hitelesítő hatóságtól (Certificate Authority, CA), egy a hitelesítést végző szervertől és egy a kliensről. Ezzel az EAP módszerrel mind a hitelesítő szerver, mind a vezeték nélküli kliens külön képviselik a saját tanúsítványukat, és ezeket a szervezetünket hitelesítő hatóság aláírása alapján ellenőrzik.

A korábbiaknak megfelelően a beállításokat szintén az `/etc/wpa_supplicant.conf` állományon keresztül végezzük el:

```
network={
    ssid="freebsdap" ❶
    proto=RSN ❷
    key_mgmt=WPA-EAP ❸
    eap=TLS ❹
    identity="loader" ❺
    ca_cert="/etc/certs/cacert.pem" ❻
    client_cert="/etc/certs/clientcert.pem" ❼
    private_key="/etc/certs/clientkey.pem" ❽
    private_key_passwd="freebsdmallclient" ❾
}
```

- ❶ Ez a mező adja meg a hálózat nevét (SSID).
- ❷ Itt az RSN (IEEE 802.11i), vagyis a WPA2 protokollt használjuk.
- ❸ A `key_mgmt` sor a kulcskezelési protokollt adja meg. A mi esetünkben ez a WPA lesz, EAP hitelesítéssel: WPA-EAP.
- ❹ Ebben a mezőben az EAP módszert nevezzük meg a kapcsolathoz.
- ❺ Az `identity` mező az EAP esetén használt azonosítót tartalmazza.
- ❻ A `ca_cert` mező a hitelesítő hatóság tanúsítványát tároló állomány elérési útvonalát adja meg. Ezt a szerver tanúsítványának hitelesítéséhez használjuk.
- ❼ A `client_cert` sor a kliens tanúsítványát tartalmazó állomány elérési útvonalát adja meg. Ennek a vezeték nélküli hálózat minden egyes kliense esetében egyedinek kell lennie.
- ❽ A `private_key` mező a kliens tanúsítványnak privát kulcsát tároló állomány elérési útját adja meg.
- ❾ A `private_key_passwd` mező a privát kulcshoz tartozó jelmondatot rögzíti.

Az `/etc/rc.conf` állományba vegyük fel a következő sorokat:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

A következő lépés a felület felébresztése lesz az `rc.d` eszköz segítségével:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
```

```
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.254 netmask 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

Természetesen, ahogy azt már az előbbiekben is megmutattuk, mindezt manuálisan is el tudjuk végezni a `wpa_supplicant` és az `ifconfig` parancsok segítségével.

31.3.3.1.3.3. WPA és EAP-TTLS

Az EAP-TLS használatakor mind a hitelesítést végző szervernek és kliensnek is kell tanúsítvány, azonban az EAP-TTLS (szállítási rétegbeli védelem EAP tunnelen keresztül) esetében a kliensnél ez elhagyható. Ez a módszer nagyjából olyan, mint amit a webes oldalak csinálnak, ahol a webszerverek egy védett SSL tunnelt képeznek még akkor is, amikor a látogatók nem rendelkeznek kliens oldali tanúsítvánnyal. Az EAP-TTLS egy titkosított TLS tunnelen keresztül védi le a hitelesítési adatok forgalmát.

Ezt ismét az `/etc/wpa_supplicant.conf` állományon keresztül tudjuk beállítani:

```
network={
    ssid="freebsdap"
    proto=RSN
    key_mgmt=WPA-EAP
    eap=TTLS ❶
    identity="test" ❷
    password="test" ❸
    ca_cert="/etc/certs/cacert.pem" ❹
    phase2="auth=MD5" ❺
}
```

- ❶ Ebben a mezőben az EAP módszert állítjuk be a kapcsolathoz.
- ❷ Az `identity` mező a titkosított TLS tunnelen keresztül az EAP hitelesítésnél felhasznált azonosítót adja meg.
- ❸ A `password` tartalmazza az EAP hitelesítésnél használt jelmondatot.
- ❹ A `ca_cert` mező hivatkozik a hitelesítő hatóság tanúsítványát tartalmazó állományra. Ez az állomány kell a szerver tanúsítványának ellenőrzéséhez.
- ❺ Ebben a mezőben a titkosított TLS tunnelben használt hitelesítési módszer nevezzük meg. Jelen esetünkben ez az EAP MD5-Challenge használatával. A “belső hitelesítés” fázisát gyakran csak “phase2”-nak (2. fázisnak) hívják.

Mindezek mellett még a következő sorokat is vegyük fel az `/etc/rc.conf` állományba:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Ezután hozzuk működésbe a felületet:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.254 netmask 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

31.3.3.1.3.4. WPA és EAP-PEAP

A PEAP (Védett EAP) az EAP-TTLS egyik alternatívájaként jött létre. A PEAP módszernek két változata van, melyek közül a leggyakoribb a PEAPv0/EAP-MSCHAPv2. A leírás további részében a PEAP elnevezéssel erre az EAP módszerre fogunk hivatkozni. A PEAP az EAP-TLS után a leginkább alkalmazott szabvány, más szóval, ha a hálózatunkban többféle operációs rendszer is megtalálható, akkor az EAP-TLS után valószínűleg a PEAP lesz a másik, amit mindegyik ismerni fog.

A PEAP hasonló az EAP-TTLS-hez: szerver oldali tanúsítványokkal hitelesíti a klienseket és titkosított TLS tunnelt hoz létre a kliens és a hitelesítést végző szerver között, amivel segíti megővni a hitelesítési információkat. Biztonság szempontjából az EAP-TTLS és a PEAP között az a különbség, hogy a PEAP hitelesítés a felhasználói nevet titkosítatlanul küldi és csak a jelszó megy át a titkosított TLS tunnelen. Az EAP-TTLS egyaránt a TLS tunnelt használja mind a felhasználói név, mind a jelszó esetében.

Az EAP-PEAP beállításait az `/etc/wpa_supplicant.conf` állományba kell felvenni:

```
network={
    ssid="freebsdap"
    proto=RSN
    key_mgmt=WPA-EAP
    eap=PEAP ❶
    identity="test" ❷
    password="test" ❸
    ca_cert="/etc/certs/cacert.pem" ❹
    phase1="peaplabel=0" ❺
    phase2="auth=MSCHAPV2" ❻
}
```

- ❶ Ebben a mezőben megadjuk, az EAP módszert használjuk a kapcsolathoz.
- ❷ Az identity mező az EAP hitelesítés során a titkosított TLS tunnelben átküldött azonosítót tartalmazza.
- ❸ A password mező az EAP hitelesítés során használt jelmondatot definiálja.

- ④ A `ca_cert` mező a hitelesítő hatóság tanúsítványát tartalmazó állomány elérési útját adja meg. Ez az állomány kell a szerver tanúsítványának ellenőrzéséhez.
- ⑤ Ez a mező a hitelesítés első fázisának (vagyis a TLS tunnel) paramétereit tartalmazza. A hitelesítést végző szervertől függően a hitelesítéshez meg kell adnunk bizonyos címkéket. A legtöbb esetben a címke a “kliens oldali EAP titkosítás” lesz, amit a `peaplabel=0` használatával állítunk be. A részleteket a `wpa_supplicant.conf(5)` man oldalon olvashatjuk.
- ⑥ Ebben a mezőben a titkosított TLS tunnelben alkalmazott hitelesítést protokollt nevezzük meg. A PEAP esetében ez az `auth=MSCHAPV2` lesz.

A következőket kell még hozzátennünk az `/etc/rc.conf` állományhoz:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Ezután már működésbe is hozhatjuk a felületet:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.254 netmask 0xfffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

31.3.3.1.4. WEP

A WEP (Wired Equivalent Privacy, azaz kábellel egyenértékű titkosság) az eredeti 802.11 szabvány része. Nincs külön hitelesítési mechanizmusa, csupán a hozzáférés-vezérlés egy gyenge formájával találkozhatunk benne, amit azonban könnyen fel lehet törni.

A WEP `ifconfig` parancs használatán keresztül állítható be:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 inet 192.168.1.100 netmask 255.255.255.0 \
    ssid saját_hálózat wepmode on weptxkey 3 wepkey 3:0x3456789012
```

- A `weptxkey` utal arra, hogy a küldés során WEP kulcsot használunk. Itt most egy harmadik kulcsot használtunk, amelynek egyeznie kell a hozzáférési pont beállításával. Ha nem tudjuk pontosan, hogy milyen kulcsot használ a hozzáférési pont, akkor próbálkozzunk az 1 érték (vagyis az első kulcs) megadásával.

- A wepkey után következik a kiválasztott WEP kulcs. *index:kulcs* alakban kell megadni, és ha itt nem adunk meg indexet, akkor azzal az 1 indexű kulcsot állítjuk be. Úgyis fogalmazhatnánk, hogy az indexet csak olyankor kell megadni, amikor nem az első kulcsot akarjuk használni.

Megjegyzés: A 0x3456789012 értéket a hozzáférési pontnál beállított kulcsra kell beállítani.

Ha érdekelnek minket a további részletek, akkor bátran lapozzuk fel az `ifconfig(8)` parancs man oldalát.

A `wpa_supplicant` segédprogramot is bevonhatjuk a vezeték nélküli felületek WEP alapú használatába. A fenti példát a következő módon tudjuk leírni az `/etc/wpa_supplicant.conf` állományban:

```
network={
    ssid="sajat_halozat"
    key_mgmt=NONE
    wep_key3=3456789012
    wep_tx_keyidx=3
}
```

Majd:

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:13:46:49:41:76 (SSID='dlinkap' freq=2437 MHz)
Associated with 00:13:46:49:41:76
```

31.3.4. Az ad-hoc működési mód

Az IBSS vagy más néven ad-hoc módot pont-pont típusú kapcsolatok kialakítására tervezték. Például, ha az A és a B gépek között egy ad-hoc típusú hálózatot akarunk létesíteni, akkor egyszerűen csak ki kell választanunk két IP-címet és egy SSID-t.

Így állítjuk be az A gépet:

```
# ifconfig wlan0 create wlandev ath0 wlanmode adhoc
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    ether 00:11:95:c3:0d:ac
    inet 192.168.0.1 netmask 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
    status: running
    ssid freebsdap channel 2 (2417 MHz) bssid 02:11:95:c3:0d:ac
    country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
    protmode CTS wme burst
```

Az `adhoc` paraméterrel utalunk arra, hogy a felület most IBSS módban működik.

A B gépen ezután már képesek vagyunk észlelni az A gépet:

```
# ifconfig wlan0 create wlandev ath0 wlanmode adhoc
      SSID/MESH ID      BSSID              CHAN RATE  S:N      INT CAPS
      freebsdap         02:11:95:c3:0d:ac    2   54M  -64:-96  100 IS    WME
```

A kimenetben szereplő I is megerősíti, hogy az A gépet ad-hoc módban érjük el. Így már csak a B gépet kell beállítanunk egy másik IP-címmel:

```
# ifconfig wlan0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.2 netmask 0xfffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
      status: running
      ssid freebsdap channel 2 (2417 Mhz 11g) bssid 02:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
      protmode CTS wme burst
```

Most már mind az A és mind a B készen áll az adatok cseréjére.

31.3.5. FreeBSD alapú hozzáférési pontok

A FreeBSD képes hozzáférési pontként (Access Point, AP) is üzemelni, így nem kell külön hardveres hozzáférési pontot vásárolnunk vagy ad-hoc hálózatot használnunk. Ez különösen akkor hasznos, amikor a FreeBSD gépet egy másik hálózat (például az internet) felé állítottuk be átjárónak.

31.3.5.1. Alapvető beállítások

Mielőtt nekiállnánk a FreeBSD-s gépünket hozzáférési pontnak beállítani, egy olyan rendszermagra lesz szükségünk, amely tartalmazza a megfelelő vezeték nélküli támogatást a kártyánkhoz. Emellett az alkalmazni kívánt biztonsági protokollok támogatását is bele kell építenünk. Ennek részleteit lásd a 31.3.2 Szakaszban.

Megjegyzés: Jelenleg az NDIS meghajtón keresztül használt Windows-os meghajtók nem teszik lehetővé hozzáférési pontok kialakítását. Egyedül a vezeték nélküli eszközök natív FreeBSD-s meghajtói ismerik a hozzáférési pont módot.

Ahogy betöltöttük a vezeték nélküli hálózatok támogatását, egyből ellenőrizni is tudjuk, hogy a vezeték nélküli eszközünk használható-e hozzáférési pontként (avagy “hostap” módban):

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 list caps
drivercaps=6f85edc1<STA,FF,TURBOP,IBSS,HOSTAP,AHDEMO,TXPMGT,SHSLOT,SHPREAMBLE,MONITOR,MBSS,WPA1,W
cryptocaps=1f<WEP,TKIP,AES,AES_CCM,TKIPMIC>
```

A fenti kimenetben láthatjuk a kártyánk tulajdonságait. A HOSTAP szó arról tanúskodik, hogy a vezeték nélküli kártyánk képes hozzáférési pontként viselkedni. Mellette még a különféle támogatott titkosítási módszerek is láthatóak: WEP, TKIP, AES stb. Ezekből az információkból tudjuk kideríteni, hogy a hozzáférési pontunkon milyen titkosítási protokollokat tudunk használni.

A vezeték nélküli eszközüket inentől már csak hozzáférési pontnak állíthatjuk át a virtuális hálózati eszköz létrehozásakor, ezért a korábban létrehozott eszközt ehhez először meg kell semmisítenünk:

```
# ifconfig wlan0 destroy
```

Ezzel létrejön a megfelelő beállításokkal, majd ezekhez állítjuk még be a többbit:

```
# ifconfig wlan0 create wlandev ath0 wlanmode hostap
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mode 11g channel 1
```

Az `ifconfig` parancs ismételt használatával le is tudjuk kérdezni az `wlan0` felület állapotát:

```
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    inet 192.168.0.1 netmask 0xffffffff broadcast 192.168.0.255
    inet6 fe80::211:95ff:fec3:dac%ath0 prefixlen 64 scopeid 0x4
    ether 00:11:95:c3:0d:ac
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
    status: running
    ssid freebsdap channel 1 (2412 Mhz) bssid 00:11:95:c3:0d:ac
    country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
    protmode CTS wme burst dtimperiod 1 -dfs
```

A `hostap` paraméterből kiderül, hogy a felület hozzáférési pont módban van.

Ha az `/etc/rc.conf` állományban megadjuk a következő sorokat, akkor a felület beállítása a rendszer indításakor magától megtörténik:

```
wlans_ath0="wlan0"
create_args_wlan0="wlanmode hostap"
ifconfig_wlan0="inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mode 11g channel 1"
```

31.3.5.2. Hitelesítés vagy titkosítás nélküli hozzáférési pontok

Habár a hozzáférési pontok működtetése nem javasolt hitelesítés vagy titkosítás nélkül, ebben a módban könnyen meg tudunk győződni a hozzáférési pontunk használhatóságáról. Ez a típusú konfiguráció ezenkívül még fontos szerepet játszik a klienseken felbukkanó hibák kiszűrésében is.

Miután sikerült az előbbieken bemutatottak alapján beállítani a hozzáférési pontunkat, egy másik vezeték nélküli gépről rögtön meg is kezdhethetjük a keresését:

```
# ifconfig ath0 up scan
SSID/MESH ID      BSSID              CHAN  RATE   S:N      INT CAPS
freebsdap         00:11:95:c3:0d:ac   1     54M    -66:-96  100 ES   WME
```

Láthatjuk, hogy a kliens megtalálta a hozzáférési pontot és tudunk is rá kapcsolódni:

```
# ifconfig wlan0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    inet 192.168.0.2 netmask 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
    status: associated
```

```
ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
roam:rate 5 protmode CTS wme burst
```

31.3.5.3. WPA titkosítást használó hozzáférési pontok

Ebben a szakaszban a FreeBSD-s hozzáférési pontunkat WPA titkosítással állítjuk be. A WPA és a WPA alapú kliensek beállításának részleteit a 31.3.3.1.3 Szakaszban találjuk.

A WPA titkosítást használó hozzáférési pontokon a **hostapd** démon foglalkozik a kliensek hitelesítésével és a kulcsok kezelésével.

A továbbiakban az összes beállítást egy olyan FreeBSD-s gépen végezzük el, amely hozzáférési pontként működik. Ahogy sikerült beállítanunk a hozzáférési pont módot, az `/etc/rc.conf` állományban a következő sor segítségével könnyen meg tudjuk oldani, hogy az **hostapd** démon a rendszerrel együtt magától elinduljon:

```
hostapd_enable="YES"
```

Mielőtt megpróbálnánk beállítani a **hostapd** démont, ne felejtsük el elvégezni a 31.3.5.1 Szakaszban említett alapvető beállításokat sem.

31.3.5.3.1. WPA-PSK

A WPA-PSK használatát olyan kis méretű hálózatok számára szánják, ahol egy külön hitelesítő szerver alkalmazása nem lehetséges vagy nem kívánatos.

A konfiguráció az `/etc/hostapd.conf` állományon keresztül történik:

```
interface=wlan0 ❶
debug=1 ❷
ctrl_interface=/var/run/hostapd ❸
ctrl_interface_group=wheel ❹
ssid=freebsdap ❺
wpa=1 ❻
wpa_passphrase=freebsdmall ❼
wpa_key_mgmt=WPA-PSK ❽
wpa_pairwise=CCMP TKIP ❾
```

- ❶ Ebben a mezőben jelöljük ki a hozzáférési pontként használt vezeték nélküli felületet.
- ❷ Ebben a mezőben adjuk meg a **hostapd** futtatása során keletkező üzenetek részletességét. A példában szereplő 1 érték ennek a legkisebb szintjét jelöli.
- ❸ A `ctrl_interface` mező megadja a **hostapd** által használt könyvtár elérési útvonalát, amiben azokat a tartományokhoz tartozó socketeket tároljuk, amelyeken keresztül olyan programokkal tudunk kommunikálni, mint például a `hostapd_cli(8)`. Itt az alapértelmezett értéket írtuk be.
- ❹ A `ctrl_interface_group` sor beállítja azt a csoportot (ez jelen esetben a `wheel`), amin keresztül a vezérlőfelület (control interface) állományaihoz hozzá tudunk férni.
- ❺ Ebben a mezőben a hálózat nevét állítjuk be.

- ⑥ A `wpa` mezővel engedélyezzük a WPA használatát és megadjuk, hogy melyik WPA hitelesítési protokollt alkalmazzuk. Az itt szereplő 1 érték a WPA-PSK hitelesítés állítja be a hozzáférési pont számára.
- ⑦ A `wpa_passphrase` mező a WPA hitelesítéshez szükséges ASCII jelmondatot tartalmazza.

Figyelem Lehetőleg mindig erős jelszavakat használjunk, amelyek kellően hosszúak és sokféle karaktert tartalmaznak, így nehezebben fejthetőek meg vagy törhetőek fel.

- ⑧ A `wpa_key_mgmt` sor a kulcsok kezelésére használt protokollt definiálja. Ez a mi esetünk most a WPA-PSK.
- ⑨ A `wpa_pairwise` mező a hozzáférési pont által elfogadott titkosítási algoritmusokat határozza meg. A példában a TKIP (WPA) és CCMP (WPA2) titkosítást is támogatjuk. A CCMP titkosítás a TKIP egyik alternatívája, és lehetőség szerint használjuk ezt. A TKIP csak olyan állomások esetében javasolt, amelyek nem támogatják a CCMP használatát.

A következő lépés a **hostapd** elindítása:

```
# /etc/rc.d/hostapd forcestart

# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2290
    inet 192.168.0.1 netmask 0xffffffff broadcast 192.168.0.255
    inet6 fe80::211:95ff:fec3:dac%ath0 prefixlen 64 scopeid 0x4
    ether 00:11:95:c3:0d:ac
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
    status: associated
    ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
    authmode WPA2/802.11i privacy MIXED deftxkey 2 TKIP 2:128-bit txpowmax 36 protmode CTS d
```

A hozzáférési pont mostantól működik, innentől a kliensek már képesek csatlakozni hozzá, bővebben lásd a 31.3.3.1.3 Szakaszban. A hozzáférési ponthoz tartozó állomásokat az `ifconfig wlan0 list sta` paranccsal tudjuk listázni.

31.3.5.4. WEP titkosítást használó hozzáférési pontok

A WEP titkosítást nem javasoljuk a hozzáférési pontok esetében, mivel nem tartalmaz semmilyen hitelesítési mechanizmust és könnyen feltörhető. Egyes régebbi vezeték nélküli kártyák azonban csak a WEP által nyújtott védelmet ismerik, ezért az ilyenek csak olyan hozzáférési pontokhoz tudnak csatlakozni, amelyek vagy nem használnak hitelesítést és titkosítást, vagy erre a WEP protokollt használják.

A vezeték nélküli eszközt tegyük hozzáférési pont módba és állítsuk be neki a megfelelő SSID-t és IP-címet:

```
# ifconfig wlan0 create wlandev ath0 wlanmode hostap
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 \
    ssid freebsdap wepmode on weptxkey 3 wepkey 3:0x3456789012 mode 11g
```

- A `weptxkey` beállítás után adjuk meg a küldéshez használt WEP kulcsot. Itt a harmadik kulcsot adtuk meg (vegyük észre, hogy a kulcsok számozása az 1 értékkel kezdődik). Ez a paramétert az adatok tényleges titkosításához kell megadni.

- A `wepkey` a kiválasztott WEP kulcs beállítását jelöli, aminek a formátuma `index:kulcs`. Ha itt nem adunk meg indexet, akkor automatikusan az első kulcsot állítjuk be. Ezért talán mondanunk sem kell, hogy az indexet csak akkor kell megadni, ha nem az első kulcsot akarjuk használni.

A `wlan0` felület állapotának megtekintéséhez adjuk ki megint az `ifconfig` parancsot:

```
# ifconfig wlan0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 192.168.0.1 netmask 0xffffffff broadcast 192.168.0.255
    ether 00:11:95:c3:0d:ac
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
    status: running
    ssid freebsdap channel 4 (2427 Mhz) bssid 00:11:95:c3:0d:ac
    country US ecm authmode OPEN privacy ON deftxkey 3 wepkey 3:40-bit
    txpower 21.5 scanvalid 60 protmode CTS wme burst dtimperiod 1 -dfs
```

Egy másik vezeték nélküli gépről most már megpróbálhatjuk megkeresni a hozzáférési pontot:

```
# ifconfig wlan0 create wlandev ath0
SSID                BSSID                CHAN  RATE  S:N    INT  CAPS
freebsdap           00:11:95:c3:0d:ac    1     54M  22:1   100  EPS
```

Láthatjuk, hogy a kliens megtalálta a hozzáférési pontot, és a megfelelő paraméterekkel (kulcs stb.) képes kapcsolódni hozzá a 31.3.3.1.4 Szakaszban leírtak szerint.

31.3.6. A vezetékes és vezeték nélküli hálózatok együttes használata

A vezetékes hálózatok általában jobb teljesítményt nyújtanak és megbízhatóbbak, miközben a vezeték nélküli hálózatok pedig nagyobb rugalmasságot és mozgásteret szolgáltatnak. Ezért a hordozható számítógépek tulajdonosaiban felmerülhet az igény, hogy egyszerre mind a kettőt használva, tetszőlegesen és problémamentesen válthassanak a hálózatok között.

FreeBSD rendszereken ún. “hibatűrő” módon két vagy akár több hálózati interfészt össze tudunk vonni. Ennek köszönhetően az aktív hálózati kapcsolat megszűnésekor rendszerünk önállóan igyekszik mindig a fennmaradó elérhető hálózatok közül a leginkább preferáltabbra váltani.

A hálózati összeköttetések összefűzésével és a hibatűrés konkrét megvalósításával az 31.6 Szakaszban foglalkozunk, ahol a 31-3 Példában láthatjuk is a vezetékes és vezeték nélküli kapcsolatok együttes használatának beállítását.

31.3.7. Hibaelhárítás

Ha valamilyen gondunk lenne a vezeték nélküli hálózatok használatával, akad néhány lépés, amivel esetleg fel tudjuk deríteni a hiba okát.

- Ha nem látjuk a hozzáférési pontot a pásztázás után, ellenőrizzük, hogy a vezeték nélküli eszközt véletlenül nem korlátoztuk-e le bizonyos csatornákra.
- Ha nem tudunk csatlakozni a hozzáférési ponthoz, akkor egyeztessük vele az állomás egyes paramétereit, beleértve a hitelesítési sémát és a biztonsági protokollokat. Minél jobban egyszerűsítsük le a konfigurációkat. Ha

WPA vagy WEP titkosítást használunk, akkor a hozzáférési ponton állítsunk be nyílt hitelesítést és kapcsoljuk ki a titkosítást, majd nézzük meg, hogy így eljut-e hozzánk valamilyen forgalom.

- Ahogy sikerült csatlakozunk a hozzáférési ponthoz, a biztonsági beállításokat olyan egyszerű eszközökkel próbáljuk meg diagnosztizálni, mint például a ping(8).

A `wpa_supplicant` segédprogrammal tudunk nyomkövetést végezni. A `-dd` opció megadásával indítsuk el manuálisan és ellenőrizzük a rendszernaplókat.

- Vannak alacsonyabb szintű nyomkövetési lehetőségek is. A 802.11 protokollt támogató rétegben is tudunk engedélyezni nyomkövetési üzeneteket a `/usr/src/tools/tools/net80211` könyvtárban található `wldebug` program segítségével. Például a

```
# wldebug -i ath0 +scan+auth+debug+assoc
net.wlan.0.debug: 0 => 0xc80000<assoc,auth,scan>
```

paranccsal a hozzáférési pontok kereséséhez és a 802.11 protokollon belül a kapcsolat megszervezéséhez szükséges kézfogásokhoz kapcsolódó konzolüzeneteket tudjuk engedélyezni.

A 802.11 rétegben rengeteg hasznos statisztikát találhatunk. Mindezeket a `wlanstats` eszközzel tudjuk kiírni. Ezeknek a statisztikáknak a 802.11 réteg összes hibáját be kell tudniuk azonosítaniuk. Vigyázzunk azonban, mert az eszközmeghajtókban a 802.11 réteg alatt rejlő bizonyos hibák ilyenkor nem jelennek meg. Az eszközfüggő problémák felderítésével kapcsolatban a megfelelő meghajtó dokumentációját olvassuk át.

Amennyiben a fenti tanácsok mentén sem sikerül orvosolnunk a hibát okát, küldjünk egy hibajelentést és mellékeljük hozzá a fentebb tárgyalt eszközök által gyártott kimeneteket.

31.4. Bluetooth

Írta: Pav Lucistnik.

31.4.1. Bevezetés

A Bluetooth egy olyan vezeték nélküli technológia, amellyel a 2,4 GHz-es frekvenciatartományban tudunk személyi hálózatokat létrehozni 10 méteren belül. Az ilyen típusú hálózatok általában alkalmi jelleggel keletkeznek különféle hordozható eszközök, mint például mobiltelefonok, kézi számítógépek és laptopok között. Eltérően más népszerű vezeték nélküli technológiáktól, például a Wi-Fi-től, a Bluetooth magasabb szintű szolgáltatási profilokat is felajánl: FTP-szerű állománszervereket, az állományok áttolását, hang átküldését, soros vonali emulációt és még sok minden mást.

A FreeBSD-ben megvalósított Bluetooth protokollkészlet a Netgraph rendszerre építkezik (lásd `netgraph(4)`). A Bluetooth alapú USB-s hardverzárak széles körét támogatja az `ng_ubt(4)` meghajtó. A Broadcom BCM2033 chipre épített Bluetooth eszközöket az `ubtbcmfw(4)` és az `ng_ubt(4)` meghajtók támogatják. A 3Com Bluetooth PC Card 3CRWB60-A eszközt az `ng_bt3c(4)` meghajtó támogatja. A soros és UART alapú Bluetooth eszközöket a `sio(4)`, `ng_h4(4)` és `hseriald(8)` ismeri. Ebben a szakaszban a Bluetooth alapú USB-s hardverzárak használatát mutatjuk be.

31.4.2. Az eszköz csatlakoztatása

Alapértelmezés szerint a Bluetooth eszközmeghajtók modulként érhetőek el. Az eszköz csatlakoztatása előtt a megfelelő meghajtót be kell töltenünk a rendszermagba:

```
# kldload ng_ubt
```

Ha a Bluetooth eszköz már a rendszer indításakor is jelen van, akkor a modult az `/boot/loader.conf` állományon keresztül is betölthetjük:

```
ng_ubt_load="YES"
```

Dugjuk be az USB-s hardverzárunkat. Az alábbihoz hasonló kimenet fog keletkezni a konzolon (vagy a rendszernaplóban):

```
ubt0: vendor 0x0a12 product 0x0001, rev 1.10/5.25, addr 2
ubt0: Interface 0 endpoints: interrupt=0x81, bulk-in=0x82, bulk-out=0x2
ubt0: Interface 1 (alt.config 5) endpoints: isoc-in=0x83, isoc-out=0x3,
      wMaxPacketSize=49, nframes=6, buffer size=294
```

Az `/etc/rc.d/bluetooth` szkript fogja végezni a Bluetooth használatához szükséges protokollkészlet elindítását és leállítását. Jó ötlet leállítani az eszköz eltávolítása előtt, de ha elhagyjuk, (általában) nem okoz végzetes hibát. Az indításkor a következő kimenetet kapjuk:

```
# /etc/rc.d/bluetooth start ubt0
BD_ADDR: 00:02:72:00:d4:1a
Features: 0xff 0xff 0xf 00 00 00 00 00
<3-Slot> <5-Slot> <Encryption> <Slot offset>
<Timing accuracy> <Switch> <Hold mode> <Sniff mode>
<Park mode> <RSSI> <Channel quality> <SCO link>
<HV2 packets> <HV3 packets> <u-law log> <A-law log> <CVSD>
<Paging scheme> <Power control> <Transparent SCO data>
Max. ACL packet size: 192 bytes
Number of ACL packets: 8
Max. SCO packet size: 64 bytes
Number of SCO packets: 8
```

31.4.3. Host Controller Interface (HCI)

A Host Controller Interface (HCI) egy parancsfelületet nyújt a működési sáv vezérlőjéhez (baseband controller) és az összeköttetések kezelőjéhez (link manager), valamint hozzáférést a hardverállapot és -vezérlő regiszterekhez. Ez a felület egy egységes módszert szolgáltat a Bluetooth működési sávjához tartozó tulajdonságok eléréséhez. Az eszközön üzemelő HCI réteg a Bluetooth hardverben található HCI firmware-rel vált adatokat és parancsokat. A Host Controller Transport Layer (vagyis a fizikai busz) meghajtója mind a két HCI réteget és a kettejük közti információcserét is elérhetővé teszi.

Az egyes Bluetooth eszközökhöz létrejön egy-egy *hci* típusú Netgraph-beli csomópont. Ez a HCI csomópont általában a Bluetooth eszközmeghajtó csomópontjához (lefelé) és az L2CAP csomópontához (felfelé) csatlakozik. Az összes HCI műveletet a HCI csomóponton kell elvégezni és nem az eszközmeghajtóhoz tartozón. A HCI csomópont alapértelmezett neve a `“devicehci”`. Ezekről többet az `ng_hci(4)` man oldalán tudhatunk meg.

Az egyik legáltalánosabb feladat a Bluetooth eszközök esetében a közelben levő további eszközök felderítése. Ezt a műveletet *tudakozódásnak* (`“inquiry”`) nevezik. A tudakozódást és az összes többi HCI-hez kapcsolódó műveletet a `hccontrol(8)` segédprogrammal tudjuk elvégezni. A lentebb látható példa azt mutatja meg, hogyan tudunk Bluetooth eszközöket keresni egy adott távolságon belül. Az elérhető eszközök listáját néhány másodpercen alatt megkapjuk. A távoli azonban eszközök csak akkor fognak válaszolni, ha *felderíthető* (`“discoverable”`) módban vannak.

```
% hccontrol -n ubt0hci inquiry
Inquiry result, num_responses=1
Inquiry result #0
    BD_ADDR: 00:80:37:29:19:a4
    Page Scan Rep. Mode: 0x1
    Page Scan Period Mode: 00
    Page Scan Mode: 00
    Class: 52:02:04
    Clock offset: 0x78ef
Inquiry complete. Status: No error [00]
```

A BD_ADDR a Bluetooth eszköz egyedi címe, hasonló a hálózati kártyák MAC-címéhez. Erre a címre lesz szükség ahhoz, hogy a továbbiakban kommunikálni tudjunk az eszközzel. Emberek számára értelmezhető nevet is hozzá tudunk rendelni a BD_ADDR címhez. Az `/etc/bluetooth/hosts` állomány tartalmazza a Bluetooth eszközökre vonatkozó információkat. A következő példában azt láthatjuk, hogyan tudunk beszédesebb nevet adni egy távoli eszköznek:

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav T39-ese
```

Amikor tudakozódni kezdünk a távoli Bluetooth eszközök jelenléte felől, a gépünket “sajat.gep.nev (ubt0)” néven fogják látni. Ez a helyi eszközkhöz rendelt név bármikor megváltoztatható.

A Bluetooth rendszer lehetőség ad pont-pont (természetesen csak két Bluetooth egység között) vagy pont-multipont típusú kapcsolatok kiépítésére. A pont-multipont kapcsolat esetén a kapcsolaton több Bluetooth eszköz osztozik. A most következő példában megláthatjuk, hogyan kell az aktív működési sávban lekérdezni a helyi eszköz létrejött kapcsolatait:

```
% hccontrol -n ubt0hci read_connection_list
Remote BD_ADDR      Handle Type Mode Role Encrypt Pending Queue State
00:80:37:29:19:a4    41  ACL    0  MAST    NONE        0      0  OPEN
```

A *kapcsolat azonosítója* (connection handle) akkor hasznos, amikor egy sávbeli kapcsolatot akarunk lezárni. Ezt általában nem kell kézzel megcsinálni. A rendszer magától lezárja az inaktív sávbeli kapcsolatokat.

```
# hccontrol -n ubt0hci disconnect 41
Connection handle: 41
Reason: Connection terminated by local host [0x16]
```

A `hccontrol help` paranccsal tudjuk lekérdezni az elérhető HCI parancsokat. A legtöbb HCI parancs végrehajtásához nem kellenek rendszeradminisztrátori jogosultságok.

31.4.4. Logical Link Control and Adaptation Protocol (L2CAP)

A Logical Link Control and Adaptation Protocol (L2CAP) a kapcsolat-orientált és a kapcsolat nélküli adatszolgáltatásokért felelős a felsőbb rétegek felé, valamint támogatja a protokollok többszörözését, a darabolást és az összerakást. Az L2CAP a magasabb szintű protokollok és az alkalmazások számára egészen 64 kilobyte méretig lehetővé teszi az adatsomagok küldését és fogadását.

A L2CAP a *csatorna* (channel) fogalmára építkezik. A csatorna egy logikai kapcsolatot képvisel a működési sávon belüli kapcsolat felett. Mindegyik csatornához egyetlen protokoll kötődik, egy a többhöz alapon. Több csatorna is

tarthat ugyanahhoz a protokollhoz, de egy csatornán nem használhatunk több protokollt. A csatornákon keresztül érkező L2CAP csomagok ezután a megfelelő felsőbb rétegbeli protokollokhoz kerülnek. Több csatorna osztható ugyanazon a sávbeli kapcsolaton.

Minden Bluetooth eszközhöz létrejön egy *l2cap* típusú Netgraph-csomópont. Az L2CAP csomópont általában egy Bluetooth HCI csomópont (lefelé) és egy Bluetooth socket (felfelé) kapcsolódik. Az L2CAP csomópont alapértelmezett neve *“device_l2cap”*. Erről részletesebben az *ng_l2cap(4)* man oldal világosít fel minket.

Ezen a szinten hasznos parancsnak bizonyulhat az *l2ping(8)*, amivel más eszközöket tudunk pingelni. Előfordulhat, hogy egyes Bluetooth implementációk nem válaszolnak semmilyen felénk küldött adatra, így az alábbi példában is szereplő 0 bytes teljesen normális.

```
# l2ping -a 00:80:37:29:19:a4
0 bytes from 0:80:37:29:19:a4 seq_no=0 time=48.633 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=1 time=37.551 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=2 time=28.324 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=3 time=46.150 ms result=0
```

Az *l2control(8)* segédprogram használható az L2CAP csomópontok különböző műveleteinek kivitelezésére. Ebben a példában a helyi eszközhöz tartozó logikai kapcsolatokat (csatornákat) és sávokat kérdezzük le:

```
% l2control -a 00:02:72:00:d4:1a read_channel_list
L2CAP channels:
Remote BD_ADDR      SCID/ DCID   PSM   IMTU/ OMTU State
00:07:e0:00:0b:ca   66/   64     3   132/  672 OPEN
% l2control -a 00:02:72:00:d4:1a read_connection_list
L2CAP connections:
Remote BD_ADDR      Handle Flags Pending State
00:07:e0:00:0b:ca   41  0           0 OPEN
```

Másik ugyanilyen diagnosztikai eszköz a *btsockstat(1)*. Ha a viselkedését tekintjük, akkor leginkább a *netstat(1)* programra hasonlít, de a Bluetooth hálózatban megjelenő adatszerkezetekkel dolgozik. Az alábbi példa az iménti *l2control(8)* parancs kimenetében szereplő logikai kapcsolatokat mutatja:

```
% btsockstat
Active L2CAP sockets
PCB      Recv-Q Send-Q Local address/PSM      Foreign address  CID   State
c2afe900  0         0 00:02:72:00:d4:1a/3    00:07:e0:00:0b:ca 66    OPEN
Active RFCOMM sessions
L2PCB    PCB      Flag MTU   Out-Q DLCs State
c2afe900 c2b53380 1    127     0    Yes  OPEN
Active RFCOMM sockets
PCB      Recv-Q Send-Q Local address      Foreign address  Chan DLCI State
c2e8bc80  0      250 00:02:72:00:d4:1a 00:07:e0:00:0b:ca 3     6    OPEN
```

31.4.5. Az RFCOMM protokoll

Az RFCOMM protokoll a soros portok emulációját valósítja meg az L2CAP protokollon keresztül. A protokoll az ETSI TS 07.10. RFCOMM szabványán alapszik, és egy egyszerű átviteli protokoll, amelyet a 9 tűs RS-232 (EIA/TIA-232-E) soros portok emulációjára készítettek fel. Az RFCOMM protokoll legfeljebb 60 kapcsolatot (RFCOMM csatorna) párhuzamos használatát támogatja két Bluetooth eszköz között.

Az RFCOMM számára a teljes kommunikációs útvonal két különböző eszközön futó alkalmazást (kommunikációs végpontot) és köztük levő kommunikációs szegmens foglalja magában. Az RFCOMM az adott eszközön a soros portot használó alkalmazások részére készült. A kommunikációs szegmens az egyik eszköztől a másikig vezető Bluetooth alapú összeköttetés (közvetlen kapcsolat).

Közvetlen kapcsolat esetén az RFCOMM csak az eszközök közti kapcsolattal foglalkozik, valamint hálózati kapcsolat esetén az eszköz és a modem közti kapcsolattal. Az RFCOMM más konfigurációkat is támogat, például olyan modulokat, amelyek az egyik oldalon a Bluetooth vezeték nélküli technológián keresztül kommunikálnak, míg a másik oldalon egy vonalas felületet nyújtanak.

A FreeBSD-ben az RFCOMM protokollt Bluetooth foglalatok rétegében valósították meg.

31.4.6. Az eszközök párosítása

Alapértelmezés szerint a Bluetooth kommunikáció nem hitelesítődik és bármelyik eszköz képes bármelyik másikkal felvenni a kapcsolatot. Egy Bluetooth eszköz (például egy mobiltelefon) egy adott szolgáltatáshoz igényelhet hitelesítést (például betárcsázáshoz). A Bluetooth alapú hitelesítés többnyire *PIN kódokkal* történik. A PIN kód egy legfeljebb 16 karakterből álló ASCII karakterlánc. A felhasználóknak mind a két eszközön ugyanazt a PIN kódot kell megadniuk. Miután megadtuk a PIN kódot, az eszközök létrehoznak hozzájuk egy *összeköttetésbeli kulcsot* (link key). Ezután ezt a kulcsot vagy az eszközökön tároljuk vagy pedig valamilyen tartós tárolón. A következő alkalommal mind a két eszközt ezt a korábban elkészített kulcsot fogja használni. Ezt az eljárást nevezzük *párosításnak* (pairing). Ha valamelyik eszköz elveszti az összeköttetés kulcsát, akkor a párosítást meg kell ismételni.

A hcsecd(8) démon felelős az összes Bluetooth alapú hitelesítési kérés lekezeléséért. Az alapértelmezett konfigurációs állománya az `/etc/bluetooth/hcsecd.conf`. Például így tudjuk benne egy mobiltelefonhoz megadni az "1234" PIN kódot:

```
device {
    bdaddr    00:80:37:29:19:a4;
    name      "Pav T39-ese";
    key       nokey;
    pin       "1234";
}
```

Semmilyen korlátozás nincs a PIN kódokra (a méretüktől eltekintve). Egyes eszközökbe (például a Bluetooth fejhallgatók) előre rögzített PIN kódot építettek bele. A `-d` kapcsoló hatására a hcsecd(8) démont az előtérben lehet futtatni, így könnyebben láthatjuk mi történik. A távoli eszközt állítsuk be a párosítás elfogadására és kezdeményezzünk felé egy Bluetooth kapcsolatot. A távoli eszköznek erre kell válaszolnia, hogy elfogadta a párosítást, majd kérni fogja a PIN kódot. Adjuk meg ugyanazt a PIN kódot, mint amit a `hcsecd.conf` állományba is beírtunk. Most már a gépünk és a távoli eszköz párban vannak. A párosítást a távoli eszközről is kezdeményezhetjük.

A FreeBSD 5.5, 6.1 és újabb változataiban az `/etc/rc.conf` állományba a következő sort kell felvenni a **hcsecd** automatikus indításához:

```
hcsecd_enable="YES"
```

Ez pedig a **hcsecd** démon által generált kimenetre példa:

```
hcsecd[16484]: Got Link_Key_Request event from 'ubt0hci', remote bdaddr 0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39', link key d
hcsecd[16484]: Sending Link_Key_Negative_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
hcsecd[16484]: Got PIN_Code_Request event from 'ubt0hci', remote bdaddr 0:80:37:29:19:a4
```

```
hcsec[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39', PIN code e
hcsec[16484]: Sending PIN_Code_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
```

31.4.7. Service Discovery Protocol (SDP)

A Service Discovery Protocol (SDP) segítségével a kliens alkalmazások képesek felderíteni, hogy a szerver alkalmazások részéről milyen szolgáltatások érhetőek el, valamint ezek a szolgáltatások milyen tulajdonságokkal rendelkeznek. A szolgáltatások tulajdonsági közé soroljuk többek között a felajánlott szolgáltatás típusát vagy osztályát, illetve a szolgáltatás kihasználásához szükséges mechanizmusra vagy protokollra vonatkozó információkat.

Az SDP az SDP szerver és az SDP kliens közti kommunikációt foglalja magában. A szerver karbantart egy listát azokról a szolgáltatási rekordokról, amelyek a szerverhez tartozó szolgáltatások jellemzőit írják le. Mindegyik ilyen szolgáltatási rekord egyetlen szolgáltatás adatait tartalmazza. A kliensek egy SDP kéréssel ezeket a szolgáltatási rekordokat kérhetik el az SDP szervertől. Amennyiben a kliens, vagy a hozzá tartozó alkalmazás a szolgáltatás használata mellett dönt, akkor a szolgáltatás használatához a megfelelő szolgáltató felé nyitnia kell egy külön kapcsolatot. Az SDP csak a szolgáltatások és azok tulajdonságainak felderítéséhez ad segítséget, de semmilyen eszközt nem tartalmaz a felhasználásukra.

Általában az SDP kliensek általában valamilyen számunkra kellő tulajdonság alapján keresnek szolgáltatásokat. Ráadásul adódhatnak olyan alkalmak is, amikor a szolgáltatások előzetes ismerete nélkül szeretnénk felderíteni a rendelkezésre álló szolgáltatások típusait. A felajánlott szolgáltatások ilyen típusú feldolgozását nevezzük *böngészésnek* (browsing).

Az `sdpd(8)` Bluetooth SDP szerver és a parancssoros `sdpcontrol(8)` kliens az alap FreeBSD telepítés része. Az alábbi példában egy SDP böngészési kérést adunk ki:

```
% sdpcontrol -a 00:01:03:fc:6e:ec browse
Record Handle: 00000000
Service Class ID List:
    Service Discovery Server (0x1000)
Protocol Descriptor List:
    L2CAP (0x0100)
        Protocol specific parameter #1: u/int/uuid16 1
        Protocol specific parameter #2: u/int/uuid16 1

Record Handle: 0x00000001
Service Class ID List:
    Browse Group Descriptor (0x1001)

Record Handle: 0x00000002
Service Class ID List:
    LAN Access Using PPP (0x1102)
Protocol Descriptor List:
    L2CAP (0x0100)
    RFCOMM (0x0003)
        Protocol specific parameter #1: u/int8/bool 1
Bluetooth Profile Descriptor List:
    LAN Access Using PPP (0x1102) ver. 1.0
```

és így tovább. Mindegyik szolgáltatáshoz hozzátartozik a tulajdonságok egy listája (például RFCOMM csatorna). Lehetséges, hogy szolgáltatástól függően bizonyos tulajdonságokat kell figyelniük. Egyes Bluetooth implementációk nem támogatják a szolgáltatások böngészését és ezért egy üres listát adnak vissza. Ebben az esetben egy konkrét szolgáltatásra tudunk rákeresni. A következő példában az OBEX Object Push (OPUSH) szolgáltatást keressük:

```
% sdpcontrol -a 00:01:03:fc:6e:ec search OPUSH
```

FreeBSD alatt az sdpd(8) szerverrel tudunk szolgáltatásokat felajánlani a Bluetooth klienseknek. A FreeBSD 5.5, 6.1 vagy későbbi változataiban ehhez a következő sort kell megadnunk az `/etc/rc.conf` állományban:

```
sdpd_enable="YES"
```

Ezután az **sdpd** démon így indítható el:

```
# /etc/rc.d/sdpd start
```

A távoli kliensek részére Bluetooth szolgáltatásokat felajánlani kívánó helyi szerver alkalmazásoknak regisztrálniuk kell magukat a helyi SDP démonnál. Például az egyik ilyen alkalmazás az `rfcomm_pppd(8)`, és elindítása után regisztrálni fogja a Bluetooth LAN szolgáltatást a helyi SDP démonnál.

A helyi SDP szerveren regisztrált szolgáltatásokat a helyi vezérlési csatornán keresztül egy browse kéréssel tudjuk lekérdezni:

```
# sdpcontrol -l browse
```

31.4.8. A betárcsázós hálózati és a PPP hálózati hozzáférési (LAN) profilok

A betárcsázós hálózati (Dial-Up Networking, DUN) profil leggyakrabban a modemek és mobiltelefonok között tűnik fel. Ez a profil a következő forgatókönyveket dolgozza fel:

- A számítógépünkkel egy mobiltelefont vagy modemet vezeték nélküli modemként használunk, amivel az internethez vagy más hálózatokhoz csatlakozunk betárcsázással.
- A számítógépünkkel egy mobiltelefonon vagy modemen keresztül fogadunk adathívásokat.

A PPP hálózati hozzáférési (LAN) profil a következő helyzetekben alkalmazható:

- LAN hozzáférés egyetlen Bluetooth eszközhez
- LAN hozzáférés több Bluetooth eszközhez
- Két gép összekötése (a soros vonali kapcsolat emulációval PPP-n keresztül)

FreeBSD alatt mind a két profilt a `pppd(8)` és az `rfcomm_pppd(8)` valósítja meg — egy olyan wrapper eszköz, amely az RFCOMM Bluetooth kapcsolatokat a PPP számára is értelmessé alakítja át. Mielőtt még bármelyik profilt elkezdhenénk használni, egy új PPP címkét kell létrehozni az `/etc/ppp/ppp.conf` állományban. Erre példát az `rfcomm_pppd(8)` man oldalon találhatunk.

A következő példában az `rfcomm_pppd(8)` programot fogjuk használni arra, hogy egy RFCOMM típusú kapcsolatot nyissunk a 00:80:37:29:19:a4 címmel rendelkező távoli Bluetooth eszköz felé. A tényleges RFCOMM csatorna számát SDP-n keresztül a távoli eszköztől kapjuk. Az RFCOMM csatorna kézzel is megadható, és ilyen esetekben az `rfcomm_pppd(8)` nem fog SDP kérést küldeni. A `sdpcontrol(8)` használatával tudjuk lekérdezni a távoli eszközön létrejött RFCOMM csatornát.

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

A PPP hálózati elérés (LAN) szolgáltatás beindításához futni kell a sdppd(8) szervernek. A helyi hálózaton keresztül csatlakozó kliensekhez létre kell hozni egy új bejegyzést az `/etc/ppp/ppp.conf` állományban. Az `rfcomm_pppd(8)` man oldalon találhatunk erre példákat. Végezetül indítsuk el az RFCOMM PPP szervert egy érvényes RFCOMM csatornaszámmal. Az RFCOMM PPP szerver ekkor automatikusan regisztrálja a Bluetooth LAN szolgáltatást a helyi SDP démonnál. A következő példában megmutatjuk, hogyan lehet elindítani egy RFCOMM PPP szervert:

```
# rfcomm_pppd -s -C 7 -l rfcomm-server
```

31.4.9. Az OBEX Object Push (OPUSH) profil

Az OBEX egy széles körben alkalmazott protokoll a mobileszközök közti egyszerű állományvitelre. Legfőképpen az infravörös kommunikációban alkalmazzák, ahol a laptopok vagy PDA-k közti általános állományátvitelre használják, illetve névjegykártyák vagy naptárbejegyzések átküldésére mobiltelefonok között és egyéb PIM alkalmazást futtató eszközök esetében.

Az OBEX szervert és klienst egy külső csomag, az **obexapp** valósítja meg, amelyet az `comms/obexapp` portból érhetünk el.

Az OBEX kliens használható objektumok áttolására vagy lehúzására az OBEX szerverhez. Ez az objektum lehet például egy névjegykártya vagy egy megbeszélés találkozó. Az OBEX kliens SDP-n keresztül tud magának RFCOMM csatornaszámot szerezni. Ezt úgy tehetjük meg, ha a szolgáltatás neve helyett egy RFCOMM csatorna számát adjuk meg. A támogatott szolgáltatások: IrMC, FTRN és OPUSH. Számként RFCOMM csatorna is megadható. Az alábbi példában egy OBEX munkamenetet láthatunk, ahol az eszköz információs objektumát húzzuk le a mobiltelefonról és egy új objektumot (egy névjegykártyát) tolunk fel a telefon könyvtárába.

```
% obexapp -a 00:80:37:29:19:a4 -C IrMC
obex> get telecom/devinfo.txt devinfo-t39.txt
Success, response: OK, Success (0x20)
obex> put new.vcf
Success, response: OK, Success (0x20)
obex> di
Success, response: OK, Success (0x20)
```

Az OBEX objektumok tologatásának támogatásához az sdppd(8) szervernek kell futnia. Továbbá a beérkező objektumok tárolásához létre kell hoznunk még egy könyvtárat is. Ez az könyvtár alapértelmezés szerint a `/var/spool/obex`. Végül indítsuk el az OBEX szervert egy érvényes RFCOMM csatorna számának megadásával. Az OBEX szerver ezután automatikusan regisztrálja az “OBEX Object Push” nevű szolgáltatást a helyi SDP démonnál. Ebben a példában láthatjuk az OBEX szerver indítását:

```
# obexapp -s -C 10
```

31.4.10. Soros vonali profil (SPP)

A sörös vonali profil (Serial Port Profile, SPP) használatával RS232 (vagy ahhoz hasonló) vonali adatátvitelt tudunk emulálni. Ez a profil a régebben fejlesztett alkalmazásokkal birkózik meg, és a Bluetooth technológiával valódi kábel helyett egy virtuális sörös portot képez le.

Az `rfcomm_sppd(1)` segédprogram ezt a soros vonali profilt valósítja meg. Így egy pszeudo terminált tudunk virtuális soros portként használni. Ha nem adunk meg RFCOMM csatornát, akkor az `rfcomm_sppd(1)` képes SDP-n keresztül kérni egyet magának a távoli eszköztől. Ha ezt felül kívánjuk bírálni, akkor a parancssorban megadhatunk akár egy konkrét RFCOMM csatornát is.

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t /dev/tty6
rfcomm_sppd[94692]: Starting on /dev/tty6...
```

Miután csatlakoztunk, a pszeudo terminált tudjuk soros portként használni:

```
# cu -l tty6
```

31.4.11. Hibaelhárítás

31.4.11.1. Nem tudunk csatlakozni a távoli eszközzel

Egyes Bluetooth eszközök nem támogatják a szerepek cseréjét (role switch). Alapértelmezés szerint amikor a FreeBSD elfogad egy új kapcsolatot, megpróbál rajta szerepet cserélni és mesterré válni. Azok az eszközök, amelyek ezt nem támogatják, nem lesznek képesek emiatt csatlakozni. Ez a szerepváltás az új kapcsolatok felépítése során zajlik le, ezért egy távoli eszköztől nem lehet megtudni, hogy ismeri-e ezt a lehetőséget. A helyi oldalon a következő HCI opcióval lehet kikapcsolni a szerepcserét:

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

31.4.11.2. Valami nem megy. Lehet látni valahogy, pontosan mi is történik?

Persze, igen. Egy külső csomag, a **hcidump** segítségével, amely a `comms/hcidump` portból érhető el. A **hcidump** segédprogram a `tcpdump(1)` programhoz hasonlítható. Ezzel lehet a Bluetooth csomagok tartalmát megnézni a terminálon vagy elmenteni ezeket egy állományba.

31.5. Hálózati hidak

Írta: Andrew Thompson.

31.5.1. Bevezetés

Gyakran hasznos lehet anélkül felosztani egy fizikai hálózatot (például egy Ethernet szegmenst) két külön hálózati szegmensre, hogy külön IP-hálózatot kellene létrehozunk és összekötnünk ezeket egy útválasztóval. A két ilyen módon kialakított hálózat összekötő eszközt nevezzük “hálózati hídnak” (bridge). A legalább két hálózati felülettel rendelkező FreeBSD rendszerek képesek hálózati híd szerepét betölteni.

A hálózati híd az eszközök adatkapcsolati rétegben a hozzá tartozó felületein megjelenő (vagyis Ethernet) címének megtanulásával működik. A két hálózat között csak akkor közvetít forgalmat, amikor a forrás és cél nem ugyanabban a hálózatban található.

A hálózati hidak bizonyos szempontból lényegében nagyon kevés porttal rendelkező Ethernet switch-ek.

31.5.2. A hálózati hidak tipikus alkalmazásai

Napjainkban akad néhány igen jellemző szituáció, ahol szükség van a hálózati hidak alkalmazására.

31.5.2.1. Hálózatok összekötése

A hálózati hidak alapvető feladata két vagy több hálózati szegmens összekötése. Az egyszerű hálózati környezet felállítása helyett több okból is felmerülhet a hidak létrehozása: kábelezési megszorítások, tűzfalazás vagy pszeudo hálózatok, például virtuális gépek felületének csatlakoztatása miatt. Egy híd használatával ráadásul össze tudunk kötni egy vezeték nélküli hozzáférési pontként üzemelő felületet egy vezetékes hálózattal.

31.5.2.2. Szűrés vagy forgalomkorlátozás tűzfallal

Sokszor előfordulhat, hogy útválasztás vagy hálózati címfordítás (NAT) nélkül szeretnénk tűzfalat használni.

Példaként képzeljünk el egy olyan kis méretű céget, amely egy DSL vagy ISDN vonalon kapcsolódik az internet-szolgáltatójához. A szolgáltatótól 13, mindenki által használható IP-címet kaptak és a hálózatukban 10 gép van. Ebben a helyzetben egy útválasztást végző tűzfal működtetése nehézkessé válna az alhálózatok problémái miatt.

Egy hídként viselkedő tűzfallal azonban minden IP számozási probléma nélkül egyszerűen be tudjuk dobni a gépeket a DSL/ISDN útválasztó mögé.

31.5.2.3. A hálózat megcsapolása

Egy hálózati híddal úgy kapcsolunk össze két hálózati szegmenst, hogy közben meg tudjuk vizsgálni a kettejük között mozgó Ethernet kereteket. Ezt a híd felületen a `bpf(4)` valamint a `tcpdump(1)` segítségével tudjuk megoldani, vagy úgy, ha egy másik felületen elküldjük az összes keret másolatát (`span`, vagyis feszítő port).

31.5.2.4. VPN az adatkapcsolati rétegben

A két Ethernet hálózatot egy IP alapú összeköttetésen keresztül is össze tudunk kötni, ha a hálózatokat egy EtherIP járaton keresztül kötjük össze híddal, vagy egy OpenVPN-hez hasonló `tap(4)` alapú megoldással.

31.5.2.5. Redundancia az adatkapcsolati rétegben

A hálózatokat több linken keresztül kötjük össze és a redundáns útvonalakat a feszítőfa protokollal (Spanning Tree Protocol, STP). Az Ethernetes hálózatok esetében a megfelelő működéshez a két eszköz között csak egyetlen aktív útvonal létezhet, így a feszítőfa protokoll észleli a hurkokat és a redundáns összeköttetéseket blokkolt állapotba teszi. Amikor azonban az aktív linkek egyike meghibásodik, akkor a protokoll újraszámolja a fát és a hálózati pontjai közti konnektivitást megpróbálja helyreállítani az addig blokkolt linkek ismételt engedélyezésével.

31.5.3. A rendszermag beállításai

Ebben a szakaszban az `if_bridge(4)` hálózati híd implementációval foglalkozunk, de a `Netgraph` segítségével is tudunk hidakat építeni. Ez utóbbiról az `ng_bridge(4)` man oldalon olvashatunk.

Amikor létrehozunk egy hálózati hidat, az `ifconfig(8)` automatikusan betölti a hozzá tartozó meghajtót. Ha viszont a rendszermag beállításait tartalmazó állományba felvesszük a `device if_bridge` sort, akkor akár be is építhetjük a rendszermagba.

A csomagszűrés minden olyan tűzfalal használható, amely a `pfil(9)` rendszerre kapcsolódik. Maga a tűzfal is betölthető modulként, vagy belefördíthető a rendszermagba.

A hálózati híddal forgalmat is tudunk szabályozni az `altq(4)` vagy a `dummynet(4)` segítségével.

31.5.4. A hálózati híd engedélyezése

Hálózati hidak felületek klónozásával hozhatóak létre. A híd létrehozásához használjuk az `ifconfig(8)` programot, és a megfelelő meghajtó automatikusan betöltődik, ha nem lenne még elérhető a rendszermagban.

```
# ifconfig bridge create
bridge0
# ifconfig bridge0
bridge0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 96:3d:4b:f1:79:7a
        id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:00:00:00:00:00 priority 0 ifcost 0 port 0
```

Ekkor létrejön a hálózati hídhez tartozó felület és véletlenszerűen generálódik hozzá egy Ethernetes cím. A `maxaddr` és a `timeout` paraméterek vezérlik, hogy a híd mennyi MAC-címet tartson meg a keretek továbbításáért felelős táblázatban és mennyi másodperc után töröljön automatikusan egy bejegyzést a legutolsó használat után. A többi paraméter a feszítőfa működését irányítja.

Vegyük fel a hídhez tartozó hálózati tagfelületeket. A híd csak akkor fog a tagfelületek között csomagokat továbbküldeni, amikor a híd és a tagok is `up` állapotban vannak:

```
# ifconfig bridge0 addm fxp0 addm fxp1 up
# ifconfig fxp0 up
# ifconfig fxp1 up
```

A híd most már átküldi az Ethernet kereteket a `fxp0` és `fxp1` felületek között. Az iméntiekkel megegyező konfigurációt az `/etc/rc.conf` állományban így alakíthatjuk ki:

```
cloned_interfaces="bridge0"
ifconfig_bridge0="addm fxp0 addm fxp1 up"
ifconfig_fxp0="up"
ifconfig_fxp1="up"
```

Ha a hídhez IP-címet is rendelni akarunk, akkor inkább magánál a hídnál adjuk meg, ne a tagoknál. Ezt statikusan vagy DHCP használatával is megtehetjük:

```
# ifconfig bridge0 inet 192.168.0.1/24
```

A hídhez IPv6 címet is hozzá tudunk rendelni.

31.5.5. Tűzfalazás

Ha engedélyezzük a csomagszűrést, a hídon áthaladó csomagok először a küldő felület érkezési oldalára kerülnek, majd a hídra, végül a megfelelő irányban levő felület küldési oldalára. Bármelyik fázis letiltható. Amikor a csomagok áramlásának iránya fontos számunkra, akkor jobban járunk, ha nem magára a hídra, hanem csak a tagfelületekre állítjuk be a tűzfalat.

A híd számos módosítható beállítással rendelkezik a nem-IP és ARP csomagok átküldésére, valamint arra, hogy az IPFW tűzfal adatkapcsolati réteg szintjén működhessen. Az `if_bridge(4)` man oldal ennek részleteit tárja fel.

31.5.6. Feszítőfák

A híd meghajtója a gyors feszítőfa protokollt (Rapid Spanning Tree Protocol, RSTP avagy 802.1w) valósítja meg, ami visszafelé kompatibilis a korábban említett feszítőfa protokollal. A feszítőfákat a hálózati topológiában felbukkanó hurkok észlelésére és eltávolítására alkalmazzák. Az RSTP azonban a hagyományos STP-nél valamivel gyorsabb konvergenciát ígér, mivel itt a szomszédos switch-ek kicserélik egymás között az adataikat, és így újabb hurkok létrehozása nélkül képesek viszonylag gyorsan egyik állapotból átváltani a másikba.

Az alábbi táblázat a támogatott működési módokat láthatjuk:

Operációs rendszer	STP módok	Alapértelmezés
FreeBSD 5.4—FreeBSD 6.2	STP	STP
FreeBSD 6.3+	RSTP vagy STP	STP
FreeBSD 7.0+	RSTP vagy STP	RSTP

A tagfelületeken az `stp` paranccsal tudjuk engedélyezni a feszítőfák használatát. Az `fxp0` és `fxp1` felületeket összekötő hídfelület esetében tehát így:

```
# ifconfig bridge0 stp fxp0 stp fxp1
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether d6:cf:d5:a0:94:6d
        id 00:01:02:4b:d4:50 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:01:02:4b:d4:50 priority 32768 ifcost 0 port 0
        member: fxp0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 3 priority 128 path cost 200000 proto rstp
                role designated state forwarding
        member: fxp1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 4 priority 128 path cost 200000 proto rstp
                role designated state forwarding
```

Láthatjuk, hogy a híd a feszítőfában megkapta a `00:01:02:4b:d4:50`-es azonosítót és a `32768`-as prioritást. Mivel `root id` értéke is ugyanez, elmondhatjuk, hogy ez a fa gyökereként funkcionáló híd.

Ha a hálózaton már valahol létezik egy másik híd:

```
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 96:3d:4b:f1:79:7a
        id 00:13:d4:9a:06:7a priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4
        member: fxp0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
```

```

port 4 priority 128 path cost 200000 proto rstp
role root state forwarding
member: fxp1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
port 5 priority 128 path cost 200000 proto rstp
role designated state forwarding

```

A root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4 sor mutatja, hogy a fa gyökerét képező híd most a 00:01:02:4b:d4:50 azonosítóval rendelkezik, és ezt a hidat 400000-res költséggel éri el a port 4 (a 4. porton) keresztül, amely jelen esetben az fxp0 felület.

31.5.7. Komolyabb hidak építése

31.5.7.1. A forgalom áramlásának átszerkesztése

A hidak támogatják az ún. megfigyelési módot, ahol a csomagokat a bpf(4) feldolgozásuk után eldobja, így nem folytatódik a feldolgozásuk vagy nem haladnak tovább. Ennek kihasználásával a két vagy több felületen érkező adatokat egyetlen bpf(4) folyamattá tudjuk alakítani. Ez olyan hálózati csapok forgalmának átszerkesztésében hasznos, ahol a két különböző felületen keresztül küldjük ki az RX/TX (fogadás/küldés) jeleket.

Az alábbi paranccsal tudjuk megoldani, hogy négy felületről érkező adatot legyünk képesek egyetlen folyamként olvasni:

```

# ifconfig bridge0 addm fxp0 addm fxp1 addm fxp2 addm fxp3 monitor up
# tcpdump -i bridge0

```

31.5.7.2. Feszítő portok

A hídhoz befutó Ethernet keretek mindegyikéről készül egy másolat, ami egy megadott feszítő porton keresztül megy tovább. Hidanként végtelen számú ilyen feszítő port létezhet, és ha egy felületet feszítő portnak adtunk meg, akkor hagyományos portként már nem használhatjuk. Ez leginkább akkor hasznos, amikor passzívan akarjuk megfigyelni a híddal rendelkező hálózatot a híd valamelyik feszítő portjára csatlakozó gépről.

Küldessük az összes keretről egy másolatot az fxp4 felületre:

```

# ifconfig bridge0 span fxp4

```

31.5.7.3. Privát felületek

A privát felületek (private interface) csak más privát felületek felé küldenek tovább adatot. Így feltétel nélkül tudjuk korlátozni a forgalmat, és sem Ethernet keretek, sem pedig ARP nem megy keresztül rajtuk. Ha viszont szelektíven akarjuk korlátozni a forgalmat, akkor helyette használjunk tűzfalat.

31.5.7.4. Tapadós felületek

Ha a híd egyik tagfelületét tapadósnak (sticky) adjuk meg, akkor a dinamikusan megtanult címek bejegyzései a gyorsítótárba kerülésük után állandósulnak. A tapadós bejegyzések soha nem évülnek el vagy cserélődnek le, még abban az esetben sem, ha utána az adott címet egy másik felületről látjuk. Így a továbbításra vonatkozó táblázatot

nem kell előre feltölteniünk, és a híd egyik oldalán meglátott kliensek nem képesek átvándorolni egy másik hálózati szegmensbe.

Másik ilyen példa a tapadós címek használatára az lehetne, amikor a hidat VLAN-nal kombináljuk, és így egy olyan útválasztót hozunk létre, ahol az ügyfeleink az IP-címtartomány pocséklása nélkül zárhatóak el egymástól. Tegyük fel, hogy az A-ügyfél a vlan100, és a B-ügyfél a vlan101 felületen csatlakozik. A híd IP-címe 192.168.0.1, amely maga is egy internet felé mutató útválasztó.

```
# ifconfig bridge0 addm vlan100 sticky vlan100 addm vlan101 sticky vlan101
# ifconfig bridge0 inet 192.168.0.1/24
```

Mind a két kliens a 192.168.0.1 címet látja alapértelmezett átjáróként, és mivel a híd gyorsítótára tapadós bejegyzéseket tartalmaz, a MAC-címeik meghamisításával nem tudják elcsúszni a másikkal forgalmát.

A VLAN-ok közti bárminemű kommunikációt privát felületek létrehozásával akadályozzuk meg (vagy egy tűzfallal):

```
# ifconfig bridge0 private vlan100 private vlan101
```

Ezzel a megoldással az ügyfeleinket teljesen elszigeteljük egymástól úgy, hogy közben az egész /24 címtartomány külön alhálózatok kialakítása nélkül kiosztható.

31.5.7.5. Címek korlátozása

Korlátozhatóak az egy felület mögül küldeni képes egyedi MAC-címek. Amikor ezen a határon felül érkeznek ismeretlen feladótól csomagok, egészen addig eldobjuk ezeket, amíg egy korábban már regisztrált bejegyzést a rendszer ki nem töröl vagy ki nem veszünk a gyorsítótárból.

A következő példában az vlan100 felületen csatlakozó A-ügyfél számára korlátozzuk le 10-re az Ethernet eszközök számát:

```
# ifconfig bridge0 ifmaxaddr vlan100 10
```

31.5.7.6. SNMP felügyelet

A hidak és az STP paraméterei az alap FreeBSD rendszerben megtalálható SNMP démonnal felügyelhetők. A hídhoz exportált felügyeleti információk (Management Information Base, MIB) megfelelnek az IETF által előírt szabványoknak, így akár tetszőleges SNMP kliens vagy bármilyen más felügyeleti szoftver alkalmas az olvasásukra.

A hidat működtető gépen az /etc/snmp.config állományban engedélyezzük a begemotSnmpdModulePath."bridge" = "/usr/lib/snmp_bridge.so" sort és indítsuk el a **bsnmpd** demont. Itt még szükség lehet más beállítások, például a közösségek nevének (community name) vagy a hozzáférési listák (access list) módosítására is. Ezzel kapcsolatban a bsnmpd(1) és az snmp_bridge(3) man oldalakat lapozzuk fel.

A következő példában a **Net-SNMP** nevű szoftver (net-mgmt/net-snmpp) fogjuk használni a híd elérésére, de ugyanerre a net-mgmt/bsnmptools port is alkalmas. Az SNMP klienst használó gépen egészítsük ki az \$HOME/.snmp/snmp.conf állományt a híd felügyeleti információinak importálásával az **Net-SNMP** rendszerébe:

```
mibdirs +/usr/share/snmp/mibs
mibs +BRIDGE-MIB:RSTP-MIB:BEGEMOT-MIB:BEGEMOT-BRIDGE-MIB
```

Az IETF BRIDGE-MIB (RFC 4188) használatán keresztül így tudjuk elindítani egy híd felügyeletét:

```
% snmpwalk -v 2c -c public bridge1.example.com mib-2.dot1dBridge
BRIDGE-MIB::dot1dBaseBridgeAddress.0 = STRING: 66:fb:9b:6e:5c:44
BRIDGE-MIB::dot1dBaseNumPorts.0 = INTEGER: 1 ports
BRIDGE-MIB::dot1dStpTimeSinceTopologyChange.0 = Timeticks: (189959) 0:31:39.59 centi-seconds
BRIDGE-MIB::dot1dStpTopChanges.0 = Counter32: 2
BRIDGE-MIB::dot1dStpDesignatedRoot.0 = Hex-STRING: 80 00 00 01 02 4B D4 50
...
BRIDGE-MIB::dot1dStpPortState.3 = INTEGER: forwarding(5)
BRIDGE-MIB::dot1dStpPortEnable.3 = INTEGER: enabled(1)
BRIDGE-MIB::dot1dStpPortPathCost.3 = INTEGER: 200000
BRIDGE-MIB::dot1dStpPortDesignatedRoot.3 = Hex-STRING: 80 00 00 01 02 4B D4 50
BRIDGE-MIB::dot1dStpPortDesignatedCost.3 = INTEGER: 0
BRIDGE-MIB::dot1dStpPortDesignatedBridge.3 = Hex-STRING: 80 00 00 01 02 4B D4 50
BRIDGE-MIB::dot1dStpPortDesignatedPort.3 = Hex-STRING: 03 80
BRIDGE-MIB::dot1dStpPortForwardTransitions.3 = Counter32: 1
RSTP-MIB::dot1dStpVersion.0 = INTEGER: rstp(2)
```

A példában látszik, hogy a dot1dStpTopChanges.0 értéke kettő, ami arra utal, hogy az STP híd topológiája kétszer változott. A topológia változása pedig azt jelenti, hogy a hálózaton belül egy vagy több link állapota megváltozott vagy egyszerűen meghibásodott és ezért egy új fát kellett számolni. A dot1dStpTimeSinceTopologyChange.0 érték adja meg, hogy ez pontosan mikor is történt.

Több híd felületének felügyeletéhez a belső BEGEMOT-BRIDGE-MIB parancsot is használhatjuk:

```
% snmpwalk -v 2c -c public bridge1.example.com
enterprises.fokus.begemot.begemotBridge
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseName."bridge0" = STRING: bridge0
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseName."bridge2" = STRING: bridge2
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseAddress."bridge0" = STRING: e:ce:3b:5a:9e:13
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseAddress."bridge2" = STRING: 12:5e:4d:74:d:fc
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseNumPorts."bridge0" = INTEGER: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseNumPorts."bridge2" = INTEGER: 1
...
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTimeSinceTopologyChange."bridge0" = Timeticks: (116927) 0:19:
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTimeSinceTopologyChange."bridge2" = Timeticks: (82773) 0:13:4
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTopChanges."bridge0" = Counter32: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTopChanges."bridge2" = Counter32: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeStpDesignatedRoot."bridge0" = Hex-STRING: 80 00 00 40 95 30 5E 3
BEGEMOT-BRIDGE-MIB::begemotBridgeStpDesignatedRoot."bridge2" = Hex-STRING: 80 00 00 50 8B B8 C6 A
```

Így tudjuk megadni, hogy a hidat mib-2.dot1dBridge részfán keresztül akarjuk megfigyelni:

```
% snmpset -v 2c -c private bridge1.example.com
BEGEMOT-BRIDGE-MIB::begemotBridgeDefaultBridgeIf.0 s bridge2
```

31.6. Linkek összefűzése és hibatűrése

Írta: Andrew Thompson.

31.6.1. Bevezetés

A `lagg(4)` felület lehetővé teszi, hogy több hálózati felületet egyetlen virtuális felületként fűzzünk össze, és ezzel egy hibatűrő és nagysebességű összeköttetést alakítsunk ki.

31.6.2. Működési módok

failover

Csak az elsődlegesként kijelölt porton keresztül fogad és küld adatokat. Amikor ez az elsődleges port elérhetetlenné válik, a következő aktív portot fogja használni. Az elsőként felvett felület válik automatikusan az elsődleges porttá, és az utána felvett összes többit pedig csak hiba esetén használjuk.

Cisco® Fast EtherChannel®

A Cisco Fast EtherChannel (FEC) technológia támogatása. Ez egy statikus beállítás, és nem egyeztetni az összefűzést a többiekkel vagy a linkek felügyeletéhez nem vált kereteket. Ha a switch támogatja az LACP használatát, akkor inkább azt válasszuk.

A FEC a kimenő forgalmat a fejlécekben szereplő protokollok alapján számolt hasítókóddal próbálja szétosztani az aktív portok között, és tetszőleges aktív porton fogad beérkező adatokat. Az említett hasítókódban egy Ethernetes forrás- és célcím szerepel, valamint ha elérhető, akkor egy VLAN címke, illetve az IPv4/IPv6 forrás- és célcím.

LACP

Az IEEE 802.3ad Link Aggregation Control Protocol (LACP) és a Marker Protocol támogatása. Az LACP megpróbálja egyeztetni a többi géppel az összefűzhető linkeket egy vagy több csoportban (Link Aggregated Group, LAG). Mindegyik ilyen csoportban ugyanolyan sebességű portokat találunk, full-duplex működési módban. A forgalmat így a legnagyobb összesebességgel rendelkező csoportban megtalálható portok között osztja el, ami a legtöbb esetben az összes portot magában foglaló csoport. A fizikai konnektivitás megváltozása esetén a linkek összefűződése igen gyorsan alkalmazkodik az új konfigurációhoz.

Az LACP a kimenő forgalmat az aktív portok között osztja szét fejlécekben szereplő protokollok alapján számolt hasítókóddal, és bármelyik aktív portról fogad bejövő forgalmat. A hasítókódban megtalálható az Ethernetes forrás- és célcím, valamint ha elérhető, akkor a VLAN címke, illetve az IPv4/IPv6 forrás- és célcímek.

Loadbalance

Ez a *FEC* mód másik neve.

Round-Robin

A kimenő forgalmat egy körkörös (Round-Robin) elvű ütemezővel osztja szét az aktív portok között és tetszőleges aktív portról fogad bejövő forgalmat. Ez a működési mód megsérti az Ethernet keretek rendezését és csak nagy körülmények mellett alkalmazzuk.

31.6.3. Példák

Példa 31-1. LACP alapú összefűzés egy Cisco® switch-csel

Ebben a példában egy FreeBSD-s gép két felületét kapcsoljuk össze switch-csel egy egyszerű terhelés-kiegyenlítéssel és hibatűréssel beállított linken keresztül. Mivel az Ethernet keretek sorrendje döntő fontosságú, ezért a két állomás között egyazon fizikai linken zajló forgalom maximális sebességét az adott felület kapacitása korlátozza. A küldési algoritmus a lehető legtöbb információ alapján próbálja egymástól megkülönböztetni a forgalmakat és elosztani ezeket a rendelkezésre álló felületek között.

A Cisco switch-en vegyünk fel a *FastEthernet0/1* és *FastEthernet0/2* interfészeket az 1 csoportba (channel group):

```
interface FastEthernet0/1
  channel-group 1 mode active
  channel-protocol lacp
!
interface FastEthernet0/2
  channel-group 1 mode active
  channel-protocol lacp
```

A FreeBSD-s gépen pedig a *fxp0* és *fxp1* használatával hozzunk létre a *lagg(4)* interfészt:

```
# ifconfig lagg0 create
# ifconfig lagg0 up lagproto lacp laggport fxp0 laggport fxp1
```

Ellenőrizzük a felület állapotát:

```
# ifconfig lagg0
```

A *ACTIVE* jelzésű, vagyis aktív állapotú portok az összefűzéshez kialakított csoport azon tagjai, amelyeknél felépült a kapcsolat a távoli switch felé és készen állnak a küldésre és fogadásra. Ha az *ifconfig(8)* programtól részletesebb kimenetet kérünk, akkor láthatjuk a csoportok azonosítóit is:

```
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
  options=8<VLAN_MTU>
  ether 00:05:5d:71:8d:b8
  media: Ethernet autoselect
  status: active
  lagproto lacp
  laggport: fxp1 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
  laggport: fxp0 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
```

A **show lacp neighbor** paranccsal kérdezhetjük le a portok állapotát:

```
switch# show lacp neighbor
Flags:  S - Device is requesting Slow LACPDUs
        F - Device is requesting Fast LACPDUs
        A - Device is in Active mode           P - Device is in Passive mode
```

Channel group 1 neighbors

Partner's information:

Port	Flags	LACP port Priority	Dev ID	Age	Oper Key	Port Number	Port State
Fa0/1	SA	32768	0005.5d71.8db8	29s	0x146	0x3	0x3D

```
Fa0/2      SA      32768      0005.5d71.8db8  29s      0x146      0x4      0x3D
```

Részletesebb kijelzést a **show lacp neighbor detail** paranccsal kaphatunk.

Példa 31-2. A hibatűrési beállítása

A hibatűrési mód arra alkalmas, hogy amikor az elsődleges porton elvesztjük a kapcsolatot, helyette egy másodlagos interfész használatára tudunk áttérni. Hozzuk létre és állítsuk be a *lagg0* interfészt, ahol az *fxp0* legyen a főinterfész, az *fxp1* pedig a tartalék interfész:

```
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport fxp0 laggport fxp1
```

Az így létrejövő interfész nagyjából az alábbi lesz, ahol eltérés a MAC-cím és az eszköz neve:

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=8<VLAN_MTU>
    ether 00:05:5d:71:8d:b8
    media: Ethernet autoselect
    status: active
    laggproto failover
    laggport: fxp1 flags=0<>
    laggport: fxp0 flags=5<MASTER,ACTIVE>
```

A forgalom kezdetben az *fxp0* felületen keresztül érkezik és távozik. Ha az *fxp0* felületen valamiért megszakadna a kapcsolat, helyette az *fxp1* lesz az aktív link. Ha később helyreáll a kapcsolat az elsődleges felületen, akkor újra az lesz aktív link.

Példa 31-3. Hibatűrési beállítása vezetékes és vezeték nélküli hálózatok között

Hordozható számítógépek használata esetén általában érdemesebb a vezeték nélküli kapcsolatot másodlagos interfészként beállítani, így csak akkor használja a rendszer, ha vezetékes hálózat nem érhető el. A *lagg(4)* segítségével egyetlen IP-címmel tudjuk használni mind a két interfészt: a teljesítmény és biztonságosság miatt elsősorban a vezetékes hálózatot használjuk, miközben megmarad a lehetőség az adatok továbbítására a vezeték nélküli kapcsolaton keresztül is.

A beállítás során a vezeték nélküli interfész MAC-címét úgy kell módosítanunk, hogy megegyezzen a *lagg(4)* címével. A *lagg(4)* interfész a saját MAC-címét az elsődleges interfésztől öröklí, amely jelen esetünkben a vezetékes interfész lesz.

A most következő példában a vezetékes hálózatunk lesz az elsődleges interfész (*bge0*), míg a vezeték nélküli (*wlan0*) a másodlagos. A *wlan0* interfészt az *iwn0* interfészből hoztuk létre, és a vezetékes kapcsolat MAC-címét állítjuk be neki. Első lépésként tehát le kell kérdeznünk a vezetékes interfész MAC-címét:

```
# ifconfig bge0
bge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=19b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM, TSO4>
    ether 00:21:70:da:ae:37
    inet6 fe80::221:70ff:feda:ae37%bge0 prefixlen 64 scopeid 0x2
    nd6 options=29<PERFORMNUD, IFDISABLED, AUTO_LINKLOCAL>
```

```
media: Ethernet autoselect (1000baseT <full-duplex>)
status: active
```

A `bge0` helyett természetesen a saját vezetékes hálózati interfészünket kell megadni, és az `ether` kezdetű sorban is saját kártyánk MAC-címe fog megjelenni. Ezután már meg is tudjuk változtatni az `iwn0` címét:

```
# ifconfig iwn0 ether 00:21:70:da:ae:37
```

Aktiváljuk a vezeték nélküli interfészt, de ne állítsunk be neki semmilyen IP-címet:

```
# ifconfig wlan0 create wlandev iwn0 ssid wlan_hálózat up
```

Hozzuk létre a `lagg(4)` interfészt a `bge0` mint elsődleges interfész megadásával, valamint a `wlan0` legyen a szükség esetén használható tartalék:

```
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport bge0 laggport wlan0
```

Az így létrehozott interfész nagyjából így fog megjelenni, egyedüli fontosabb eltérések a MAC-címek és az eszközök nevei:

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=8<VLAN_MTU>
ether 00:21:70:da:ae:37
media: Ethernet autoselect
status: active
laggproto failover
laggport: wlan0 flags=0<>
laggport: bge0 flags=5<MASTER,ACTIVE>
```

Hogy ne kelljen a rendszer minden egyes indítása után ezt a műveletet megismételni, vegyük fel a következő sorokat az `/etc/rc.conf` állományba:

```
ifconfig_bge0="up"
ifconfig_iwn0="ether 00:21:70:da:ae:37"
wlans_iwn0="wlan0"
ifconfig_wlan0="WPA"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto failover laggport bge0 laggport wlan0 DHCP"
```

31.7. Lemez nélküli működés

Frissítette: Jean-François Dockès. Átdolgozta és javította: Alex Dupre.

A FreeBSD képes hálózaton keresztül elindulni és helyi lemez nélkül egy NFS szerver által megosztott állományrendszer csatlakoztatásával működni. Ehhez a szabványos konfigurációs állományok módosításán kívül semmi másra nincs szükségünk. Egy ilyen rendszert viszonylag könnyű beállítani, mivel az összes hozzávaló szinte készen elérhető:

- Rögtön adott legalább két módszer, ha a rendszermagot hálózaton keresztül akarjuk betölteni:
 - PXE: az Intel által fejlesztett Preboot eXecution Environment (“indítás előtti végrehajtási környezet”) nevű rendszer a hálózati kártyákba vagy alaplapokba épített ROM segítségével teszi lehetővé az intelligens rendszerindítást. A `pxeboot(8)` man oldalán olvashatunk erről részletesebben.

- Az **Etherboot** port (`net/etherboot`) olyan ROM-ba programozható kódot készít, amellyel rendszermagokat tudunk hálózaton keresztül betölteni. Ez a kód egyaránt felhasználható egy hálózati rendszerindító PROM beégetéséhez, vagy betölthető a helyi floppy (esetleg merev)lemezről, illetve MS-DOS rendszer alól. Elég sok hálózati kártya támogatja ezt a módot.
- Egy mintaszкриpt (`/usr/share/examples/diskless/clone_root`) is próbálja megkönnyíteni a szerveren a munkaállomás rendszerindító állományrendszerének létrehozását és karbantartását. Ezt a szkriptet valószínűleg némileg módosítani kell, de így is sokat segít az elindulásban.
- Az `/etc` könyvtárban található szabványos rendszerindításhoz használt állományok, amelyekkel a lemez nélküli indulást lehet detektálni és segíteni.
- A lapozás, amennyiben szükséges, NFS vagy helyi lemez segítségével oldható meg.

Számos módon állíthatunk be egy lemez nélküli munkaállomást. Rengeteg részből tevődik össze, és ezek legtöbbje remekül testreszabható az igényeinknek. A továbbiakban egy teljes rendszer összeállításának lehetséges variációit ismertetjük, különös hangsúlyt fektetünk arra, hogy egyszerűek és a hagyományos FreeBSD indítószkriptekkel kompatibilisek maradjanak. A bemutatandó rendszer a következő jellemzőkkel bír:

- A lemez nélküli munkaállomások megosztott `/` és `/usr` állományrendszereket használnak.
A rendszer indításához használt gyökér állományrendszer a szabvány FreeBSD-s gyökér (ez általában a szerveré), ahol néhány állományt felülírtunk a lemez nélküli működéshez vagy azért, mert egyszerűen az adott munkaállomáshoz tartozik.
A gyökér azon részeit, amelyeket írhatóvá kívánunk tenni, `md(4)` alapú állományrendszerekkel lapoljuk felül. Ilyenkor azonban bármilyen rajtuk ejtett változtatás a rendszer újraindításával elveszik.
- A rendszermagot vagy az **Etherboot** vagy a PXE használatával küldessük át és töltjük be, mivel egyes helyzetekben ezekre szükség lesz.

FigyelemA bemutatott rendszer nem biztonságos. Helyezzük a hálózatunk egy jól védett részére, és a többi gép ne tekintse megbízhatónak.

A szakaszban szereplő összes információt a FreeBSD 5.2.1-RELEASE változatával teszteltük.

31.7.1. Háttérinformációk

A lemez nélküli munkaállomások beállítása egyszerre adja magát és könnyen is elvethető. Az elkövetett hibákat olykor számos okból kifolyólag nehéz felismerni. Például:

- A fordítási időben megadott beállítások mást eredményeznek futási időben.
- A hibaüzenetek gyakran titokzatosak vagy esetleg teljesen el is maradnak.

Ezért ha valamennyire tisztában vagyunk a háttérben zajló folyamatokkal, akkor sokkal több eséllyel leszünk képesek megoldani a menet közben felmerülő problémákat.

A rendszernek a sikeres felkapaszkodáshoz több műveletet is végre kell hajtania:

- A gépnek szüksége van olyan induló paraméterekhez, mint például az IP-cím, a végrehajtható állomány neve, a szerver neve, a gyökér elérési útja. Ezeket a DHCP vagy a BOOTP protokollok használatával adhatjuk meg. A DHCP a BOOTP kompatibilis kiterjesztése, ezért ugyanazokat a portokat és alapvető csomagformátumot alkalmazza.

A rendszerüket kizárólag BOOTP használatával is beállíthatjuk. A bootpd(8) szerver az alap FreeBSD rendszer része.

A DHCP azonban rengeteg előnnyel rendelkezik a BOOTP protokollal szemben (áttekinthetőbb konfigurációs állományok, a PXE használatának lehetősége, illetve sok minden más, ami nem csak a lemez nélküli működéshez kellhet), ezért itt alapvetően egy DHCP alapú konfigurációt mutatunk be, de ahol megoldható, megemlíjtük a bootpd(8) esetén alkalmas példákat is. A mintaként szolgáló konfiguráció az **ISC DHCP** szoftvercsomagot használja (a tesztszerverre ennek a 3.0.1.r12 verzióját telepítettük fel).

- A gépnek egy vagy több programot kell a saját memóriájába áttöltenie. Erre vagy a TFTP vagy pedig az NFS alkalmas. A TFTP és az NFS között sok helyen fordítási időben tudunk választani. Gyakori hibaforrás a protokollhoz rosszul megadott állománynevek használata: a TFTP általában az összes állományt a szerverről egyetlen könyvtárból tölti át, ezért arra számít, hogy a neveiket ehhez viszonyítva adjuk meg. Az NFS használata során azonban abszolút elérési utakat kell megadnunk.
- A rendszer indítását lehetővé tevő közbenső programokat és a rendszermagot valahogy inicializálni kell és elindítani. Ezen a területen több fontos változat kapott helyet:
 - A PXE a pxeboot(8) kódját fogja betölteni, ez lényegében a FreeBSD betöltő harmadik fokozatának egy módosított változata. A loader(8) a működéséhez szükséges paramétereket a rendszer indításakor kapja meg, majd a vezérlés átadása előtt ezeket a rendszermag környezetében hagyja. Ebben az esetben akár a **GENERIC** rendszermag is használható.
 - Az **Etherboot** kevesebb előkészítéssel közvetlenül magát a rendszermagot tölti be. Ehhez azonban egy saját rendszermagot kell építeni, külön beállításokkal.

A PXE és az **Etherboot** egyaránt jól használható. Mivel azonban a rendszermagok általában a loader(8) kódjára hagyják a munka legnagyobb részét, ezért ahol lehetséges, a PXE megoldását érdemes alkalmazni.

Tehát ha az alaplap BIOS és a hálózati kártya is támogatja a PXE használatát, akkor válasszunk inkább azt.

- Végezetül a gépnek valamilyen módon hozzá kell tudnia férnie az állományrendszerekhez. Erre többnyire az NFS jöhet szóba.

A további részleket lásd a diskless(8) man oldalon.

31.7.2. Beállítási útmutató

31.7.2.1. Beállítás a ISC DHCP használatával

Az **ISC DHCP** szervere képes a BOOTP és DHCP kéréseket is megválaszolni.

Az **ISC DHCP 3.0** nem az alaprendszer része, ezért a használatához először telepítenünk kell a `net/isc-dhcp30-server` portot vagy a neki megfelelő csomagot.

Ahogy feltelepítettük, le kell futtatnunk az **ISC DHCP** konfigurációs állományát (ezt általában `/usr/local/etc/dhcpd.conf` néven találjuk meg). A most következő, megjegyzésekkel kiegészített példában egy `margaux` nevű gép az **Etherboot**, valamint egy `corbieres` nevű gép PXE használatával akar kapcsolódni:

```
default-lease-time 600;
```

```

max-lease-time 7200;
authoritative;

option domain-name "minta.com";
option domain-name-servers 192.168.4.1;
option routers 192.168.4.1;

subnet 192.168.4.0 netmask 255.255.255.0 {
    use-host-decl-names on; ❶
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.4.255;

    host margaux {
        hardware ethernet 01:23:45:67:89:ab;
        fixed-address margaux.minta.com;
        next-server 192.168.4.4; ❷
        filename "/data/misc/kernel.diskless"; ❸
        option root-path "192.168.4.4:/data/misc/diskless"; ❹
    }
    host corbieres {
        hardware ethernet 00:02:b3:27:62:df;
        fixed-address corbieres.minta.com;
        next-server 192.168.4.4;
        filename "pxeboot";
        option root-path "192.168.4.4:/data/misc/diskless";
    }
}

```

- ❶ Ez a beállítás arra utasítja a **dhcpcd** démon, hogy a lemez nélküli gép hálózati neveként a host deklarációban megadott értéket küldje el. Ezt úgyis meg lehet csinálni, hogy felvesszünk egy `option host-name margaux` részt a host deklarációk közé.
- ❷ A `next-server` direktíva a betöltő vagy a rendszermag betöltéséért felelős TFTP vagy NFS szervert jelöli ki (alapértelmezés szerint ez megegyezik a DHCP szerverrel).
- ❸ A `filename` direktíva azt az állományt adja meg, amelyet az **Etherboot** vagy a PXE a következő végrehajtási lépésben betölt. Ezt a kiválasztott átviteli módnak megfelelően kell megadni. Az **Etherboot** lefordítható az NFS vagy a TFTP használatával is. A FreeBSD port alapból az NFS támogatását tartalmazza. A PXE a TFTP protokollt használja, ezért itt relatív állományneveket adunk meg (ez persze a TFTP szervertől függ, de általában ez a jellemző). Sőt, a PXE a `pxeboot` állományt tölti be, nem is a rendszermagot. Léteznek további érdekes lehetőségek is, mint például a `pxeboot` állomány betöltése a FreeBSD CD-jén található `/boot` könyvtárból (mivel a `pxeboot(8)` a **GENERIC** rendszermagot képes betölteni, ezért a PXE használatával akár egy távoli CD-meghajtóról is indíthatjuk a rendszert).
- ❹ A `root-path` opció a rendszer indításához használt gyöker állományrendszert nevezi meg, amelyet többnyire az NFS jelölési módszere szerint kell megadni. A PXE használata során el lehet hagyni a gép IP-címét egészen addig, amíg nem engedélyezzük a rendszermagban a BOOTP beállítást. Az NFS szervert ekkor megegyezik a TFTP szerverrel.

31.7.2.2. Beállítás a BOOTP használatával

Itt a **bootpd** (egyetlen kliensre korlátozott) beállítását láthatjuk. Ezt az `/etc/bootptab` állományba tesszük.

Ne feledjük, hogy a BOOTP használatához az **Etherboot** portot a `NO_DHCP_SUPPORT` beállítással kell fordítanunk, miközben a PXE esetében kell a DHCP. Egyébként a **bootpd** egyedüli nyilvánvaló előnye csupán annyi, hogy az alaprendszer része.

```
.def100:\
:hn:ht=1:sa=192.168.4.4:vm=rfc1048:\
:sm=255.255.255.0:\
:ds=192.168.4.1:\
:gw=192.168.4.1:\
:hd="/tftpboot":\
:bf="/kernel.diskless":\
:rp="192.168.4.4:/data/misc/diskless":
```

```
margaux:ha=0123456789ab:tc=.def100
```

31.7.2.3. A rendszer előkészítése az Etherboot számára

Az Etherboot honlapján (<http://etherboot.sourceforge.net>) találhatunk egy minden részletre kiterjedő dokumentációt (angolul) (<http://etherboot.sourceforge.net/doc/html/userman/t1.html>), amely elsősorban ugyan a Linux típusú rendszerek számára íródott, de ettől függetlenül még hasznos információkat tartalmaz. A továbbiakban csak annyit szeretnénk körvonalazni, hogy az **Etherboot** miként bírható működésre FreeBSD rendszerekkel.

Először telepítenünk kell a `net/etherboot` csomagot vagy portot.

Az **Etherboot** beállítását (vagyis a TFTP használatának megadását az NFS helyett) az **Etherboot** forrását tartalmazó könyvtárban található `Config` állomány megfelelő átírásával tudjuk megtenni.

Itt most floppyról fogjuk indítani a rendszert. A többi módszerrel (PROM vagy MS-DOS program) kapcsolatban olvassuk el az **Etherboot** dokumentációját.

A rendszerindító lemez elkészítéséhez tesszük egy lemezt annak a gépnek a meghajtójába, ahová az **Etherboot** felkerült. Váltunk az **Etherboot** könyvtárán belül az `src` alkönyvtárba és gépeljük be:

```
# make bin32/eszköztípus.fd0
```

Az `eszköztípus` a lemez nélküli munkaállomás Ethernet kártyájától függ. Az ugyanebben a könyvtárban található `NIC` állományból tudjuk kiolvasni, hogy az adott kártyához melyik `eszköztípus` tartozik.

31.7.2.4. A rendszer indítása PXE használatával

Alapértelmezés szerint a `pxeboot(8)` betöltő a rendszermagot NFS-en keresztül tölti be. Ha az `/etc/make.conf` állományban a `LOADER_TFTP_SUPPORT` beállítást adjuk meg, akkor TFTP támogatással is lefordítható. Ezzel kapcsolatban a `/usr/share/examples/etc/make.conf` állományban található megjegyzéseket érdemes elolvasnunk.

A `make.conf` állományban még további két másik hasznos opciót is találhatunk a soros vonali konzollal üzemelő lemez nélküli gépek számára: az egyik a `BOOT_PXEldr_PROBE_KEYBOARD`, a másik pedig a `BOOT_PXEldr_ALWAYS_SERIAL`.

A gép indításakor úgy tudjuk beüzemelni a PXE használatát, ha a BIOS beállításai között a `Boot from network` opciót választjuk ki, vagy a gép bekapcsolása után lenyomjuk hozzá a megfelelő funkcióbillentyűt.

31.7.2.5. A TFTP és NFS szerverek beállítása

Ha a PXE vagy az **Etherboot** a TFTP protokollt használja, akkor az állományszerveren a **tftpd** démonat kell elindítani:

1. Készítsünk egy könyvtárat, ahonnan majd a **tftpd** küldi az állományokat, például legyen ez a `/tftpboot`.
2. Vegyük fel a következő sort az `/etc/inetd.conf` állományunkba:

```
tftp      dgram    udp      wait      root      /usr/libexec/tftpd      tftpd -l -s /tftpboot
```

Megjegyzés: A tapasztalat szerint egyes PXE verziók a TFTP TCP alapú változatát használják. Ebben az esetben vegyük fel még egy második sort is, ahol a `dgram udp` részt `stream tcp`-re cseréljük.

3. Mondjuk meg az **inetd** démonnak, hogy olvassa újra a konfigurációs állományát. Az alábbi parancs megfelelő működéséhez Az `inetd_enable="YES"` sornak szerepelnie kell az `/etc/rc.conf` állományban:

```
# /etc/rc.d/inetd restart
```

A `tftpboot` könyvtárat bárhova rakhatjuk a szerveren. Viszont az `inetd.conf` és `dhcpd.conf` állományokban ezt ne felejtjük fel megadni.

Minden esetben engedélyeznünk kell az NFS használatát és vele együtt exportálni az NFS szerverről elérni kívánt állományrendszereket.

1. Az `/etc/rc.conf` állományba tegyük bele a következőt:

```
nfs_server_enable="YES"
```

2. Az `/etc/exports` állományban a lemez nélküli rendszereknek szánt gyökérkönyvtárat tegyük elérhetővé (a példában írjuk át a kötet csatlakozási pontját és a *margaux corbieres* helyére állítsuk be a saját lemez nélküli munkaállomásaink neveit:

```
/data/misc -alldirs -ro margaux corbieres
```

3. Kérjük meg a **mountd** démonat, hogy olvassa újra a konfigurációs állományát. Előfordulhat azonban, hogy ehhez először az NFS szolgáltatást kell engedélyezni az `/etc/rc.conf` állományból és újraindítani a gépet.

```
# /etc/rc.d/mountd restart
```

31.7.2.6. Lemez nélküli rendszermag fordítása

Ha az **Etherboot** használata mellett döntünk, akkor a lemez nélküli kliensek számára a rendszermagot a következő beállítások használatával kell újrafordítani (a megszokottak mellett):

```
options      BOOTP          # BOOTP-n keresztül kérünk IP-címet és hálózati nevet
options      BOOTP_NFSROOT  # a BOOTP-től kapott információk alapján csatoljuk a gyökeret NFS-en k
```

Ezek mellett valószínűleg szükségünk lesz a BOOTP_NFSV3, BOOT_COMPAT és BOOTP_WIRED_TO beállítások megadására is (lásd a NOTES állományt).

A beállítások nevei régről származnak és némileg félrevezetőek lehetnek, mivel valójában semmit sem változtatnak a rendszermagban levő DHCP vagy a BOOTP rutinok használatában (egyébként meg lehet adni vagy az egyik vagy a másik protokoll kizárólagos használatát is).

Fordítsuk le a rendszermagot (lásd 8 fejezet), és másoljuk a dhcpd.conf állományban megadott helyre.

Megjegyzés: Amikor a PXE protokollt használjuk, a rendszermagot nem fontos az imént felsorolt paraméterekkel fordítanunk (habár ajánlatos). Az engedélyezésünkkel több DHCP kérés keletkezik a rendszermag elindulása közben, ezért kisebb a kockázata annak, hogy a pxeboot(8) által bizonyos esetekben megszerzett és az új értékek között valamilyen ellentmondás jön létre. A használatuk egyik előnye, hogy így mellékhatásként a hálózati nevünket is megkapjuk. Ellenkező esetben erre is találnunk kellene valamilyen módot, például fenntartani egy-egy rc.conf állományt minden kliensen.

Megjegyzés: Az Etherboot csak akkor lesz képes betölteni a rendszermagot, ha device hinteket is beépítünk. Ezt a következő beállítással tudjuk megoldani (erről bővebben lásd a NOTES állomány megjegyzéseit):

```
hints                "GENERIC.hints"
```

31.7.2.7. A rendszerindító állományrendszer előkészítése

A dhcpd.conf állomány root-path beállításának megfelelően hozzunk létre a rendszer indítására alkalmas gyökér állományrendszert.

31.7.2.7.1. Az állományrendszer feltöltése a make world paranccsal

Ezzel a módszerrel a DESTDIR könyvtárba pillanatok alatt telepíteni tudunk egy teljes szűz rendszert (és nem csak a rendszerindító állományrendszert). Ehhez mindössze csak annyit kell tenni, hogy lefuttatjuk a következő szkriptet:

```
#!/bin/sh
export DESTDIR=/data/misc/diskless
mkdir -p ${DESTDIR}
cd /usr/src; make buildworld && make buildkernel
make installworld && make installkernel
cd /usr/src/etc; make distribution
```

Miután végzett, már csak a DESTDIR könyvtárban található /etc/rc.conf és /etc/fstab állományokat kell az igényeinkhez igazítani.

31.7.2.8. A lapozóterület beállítása

Amennyiben szükséges, a szerveren található lapozóállományt NFS-en keresztül el tudjuk érni.

31.7.2.8.1. Lapozás NFS-sel

A rendszermag maga nem támogatja az NFS alapú lapozás engedélyezését a rendszer indításakor. A lapozóállományt ezért a rendszerindító szkripteken keresztül aktiváljuk, amelyekben csatlakoztatunk egy írható állományrendszert, ahol létrehozunk és engedélyezzük a lapozóállományt. Tetszőleges méretű lapozóállományt például így tudunk készíteni:

```
# dd if=/dev/zero of=/a/lapozóállomány/helye bs=1k count=1 oseek=100000
```

Az engedélyezéséhez pedig a következő sort kell felvenni az `rc.conf` állományba:

```
swapfile=/a/lapozóállomány/helye
```

31.7.2.9. Egyéb problémák

31.7.2.9.1. Írásvédett `/usr` használata

Ha a lemez nélküli munkaállomáson X szerveret akarunk futtatni, akkor az **XDM** konfigurációs állományait kicsit módosítanunk kell, mert alapértelmezés szerint a `/usr` könyvtárban hozza létre a naplót.

31.7.2.9.2. Nem FreeBSD-s szerver használata

Amikor a rendszer indításához használt állományrendszert nem egy FreeBSD alapú számítógépen tároljuk, akkor először ezt egy FreeBSD-s gépen kell elkészíteni, majd a `tar` vagy `cpio` segítségével átmásolni a megfelelő helyre.

Ilyen helyzetekben gyakran gondok adódhatnak olyan speciális állományokkal, mint például amelyek a `/dev` könyvtárban találhatók, mivel a fő- és aleszközazonosítók tárolására szánt méret különbözhet. Ezt úgy oldhatjuk meg, ha exportálunk egy könyvtárat a nem FreeBSD alapú szerveren, ezt csatlakoztatjuk a FreeBSD-s gépen, majd a `devfs(5)` segítségével a eszközeleírókat a felhasználó számára észrevétlen módon foglaljuk le.

31.8. ISDN

Az ISDN technológiai és hardveres háttéréről sokat megtudhatunk Dan Kegel ISDN-ről szóló oldalán (angolul) (<http://www.alumni.caltech.edu/~dank/isdn/>).

Az ISDN használatát röviden így foglalhatnánk össze:

- Ha Európában élünk, akkor minden bizonnyal az ISDN kártyákkal foglalkozó szakaszt érdemes elolvasnunk.
- Ha elsősorban betárcsázós ISDN-nel szeretnénk csatlakozni az internetre egy internet-szolgáltatón keresztül, akkor a terminál adaptereket tárgyaló szakaszt nézzük meg. A szolgáltatók váltásakor ezzel jár a legtöbb rugalmasság és a legkevesebb probléma.
- Ha két helyi hálózat összekötésére használjuk, vagy az internethez egy bérelt ISDN vonalon keresztül kapcsolódunk, akkor egy önálló útválasztó vagy hálózati híd beállításában érdemes gondolkodnunk.

A költség fontos szerepet játszik az elfogadható megoldás kiválasztásában. A most következő lehetőségeket a legolcsóbbtól indulva kezdjük el felsorolni egészen a legdrágáig.

31.8.1. ISDN kártyák

Készítette: Hellmuth Michaelis.

A FreeBSD-ben megtalálható ISDN implementáció csak a DSS1/Q.931 (más néven Euro-ISDN) szabvány szerint gyártott passzív kártyákat támogatja. Ismer azonban egyes olyan aktív kártyákat is, amelyeknél a firmware további más jelkezelési protokollokat is támogat. Ilyen többek közt az elsőként támogatott Primary Rate (PRI) ISDN kártya.

Az **isdn4bsd** szoftver segítségével kapcsolódni tudunk más ISDN útválasztókhoz IP-n keresztül a nyers HDLC felett, vagy szinkron PPP használatával. Mindezeket a rendszermagban található PPP-re vagy az **isppp**-re építkezik.

FreeBSD alatt egyre több PC-s ISDN kártyához készül el a támogatás, és a visszajelzések azt mutatják, hogy Európában és a világ minden részén sikerrel használják ezeket.

A passzív ISDN kártyák közül is leginkább az Infineon (korábban Siemens) gyártmányú ISAC/HSCX/IPAC ISDN chipkészletek támogatottak, de a Cologne chippel rendelkező (de csak ISA buszos) ISDN kártyák, a Winbond W6692 chipes PCI buszos kártyák, és a Tiger300/320/ISAC chipkészletek egyes változatai, valamint néhány gyártófüggő chipkészlettel rendelkező kártya, mint például az AVM Fritz!Card PCI V.1.0 és az AVM Fritz!Card PnP is remekül működik.

Jelenleg a következő aktív ISDN kártyákat támogatja a rendszer: AVM B1 (ISA és PCI) BRI kártyák és az AVM T1 PCI PRI kártyák.

Az **isdn4bsd** dokumentációját a rendszerünkön belül a `/usr/share/examples/isdn/` könyvtárban találhatjuk meg, vagy közvetlenül az **isdn4bsd** honlapján (<http://www.freebsd-support.de/i4b/>), ahol több hivatkozást is találunk tippekre, hibajegyzékekre és bősegebb dokumentációra, például az **isdn4bsd** saját kézikönyvére (<http://people.FreeBSD.org/~hm/>).

Ha szeretnénk egy másik ISDN protokoll támogatásának kifejlesztésében résztvenni, vagy egy jelenleg még nem támogatott ISDN kártyát használhatóvá tenni, esetleg valamilyen más módon segíteni az **isdn4bsd** ügyét, vegyük fel a kapcsolatot Hellmuth Michaelis fejlesztővel.

Az **isdn4bsd** telepítésével, beállításával és hibaelhárításával kapcsolatos kérdéseinket a **freebsd-isdn** (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn>) levelezési listán tehetjük fel.

31.8.2. ISDN terminál adapterek

Az ISDN számára olyanok a terminál adapterek, mint a hagyományos telefonvonalak számára a modemek.

A legtöbb terminál adapter a Hayes-modemek szabványos AT parancskészletét használja, és könnyen be lehet iktatni egy modem helyett.

A terminál adapterek alapvetően ugyanúgy működnek, mint a modemek, kivéve, hogy egy átlagos modemnél jóval nagyobb adatátviteli sebességre képesek. Ezért a PPP kapcsolatunkat pontosan ugyanúgy kell beállítani, mint a modemek esetében. Ne felejtjük a soros pont sebességét a maximális értékre állítani.

A terminál adapterek használatának egyik legnagyobb előnye, hogy segítségükkel dinamikus PPP-n keresztül tudunk az internet-szolgáltatókhoz kapcsolódni. Mivel az IP-címtartomány egyre inkább szűkösebb, a legtöbb szolgáltató nem szívesen oszt ki bárkinek is statikus IP-címet. A legtöbb önálló útválasztó azonban nem képes alkalmazkodni az IP-címek dinamikus kiosztásához.

A terminál adapter az elérhető lehetőségeket és a kapcsolat stabilitását tekintve teljesen a PPP démontól függ. Emiatt egy FreeBSD-s gépet könnyű modemről átállítani az ISDN használatára, ha már egyszer beállítottuk a PPP démont. Ezzel együtt azonban a PPP használata során tapasztalt problémák ugyanúgy ismét felmerülnek.

Ha a maximális stabilitásra van szükségünk, akkor a rendszermag PPP beállítását használjuk, és ne a felhasználói PPP megoldást.

A FreeBSD hivatalosan az alábbi terminál adaptereket ismeri:

- Motorola BitSurfer és Bitsurfer Pro
- Adtran

Valószínűleg a többi terminál adapterrel is képes együttműködni, mivel a terminál adapterek gyártói általában igyekeznek a termékeiket a szabványos modem AT parancskészletével kompatibilissá tenni.

Az igazi probléma a külső terminál adapterekkel adódik, mivel, akárcsak a modemek esetében, egy nagyon jó soros kártyát igényelnek.

A soros eszközök működésének részleteit valamint az aszinkron és szinkron soros portok közti különbségeket a FreeBSD soros hardverekről (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/serial-uart/index.html) szóló cikkében olvashatjuk.

A terminál adaptereken keresztül elérhető sebességet a PC-kben található szabványos (aszinkron) soros port 115,2 Kb/mp-re korlátozza, még 128 Kb/mp-es adatátvitelű kapcsolatok esetében is. Az ISDN által nyújtott 128 Kb/mp kihasználásához a terminál adaptert egy szinkron soros kártyával kell összekötnünk.

Ne higgyük, hogy egy belső terminál adapter megvásárlásával megmenekülünk ettől a gondtól. A belső terminál adapterekbe egyszerűen csak egy sima szabványos PC-s soros portot építettek bele. Mindössze egy soros kábelt és egy konnektort takarítunk meg velük.

A terminál adapterhez csatlakozó szinkron kártyák legalább olyan gyorsak, mint egy önálló útválasztó, és egy egyszerű 386-osra épülő FreeBSD rendszerrel talán még rugalmasabban is kezelhetők.

A terminál adapter plusz szinkron kártya kontra önálló útválasztó kérdése már hitkérdéssé fajult, amiről igen sokat vitatkoztak szerte a levelezési listákon. A teljes okfejtés elolvasásához az archívum (<http://www.FreeBSD.org/search/index.html>) böngészését javasoljuk.

31.8.3. Önálló ISDN hálózati hidak és útválasztók

Az ISDN hidak vagy útválasztók nem egészen a FreeBSD vagy operációs rendszerek területéhez tartoznak. Az útválasztás és a hálózatok hidak alapjainak a számítógépes hálózatokról szóló szakirodalomban járhatunk utána.

Ebben a szakaszban a hálózati híd és az útválasztó kifejezéseket egymás szinonímájaként fogjuk használni.

Ahogy az olcsóbb ISDN útválasztók és hidak árai egyre jobban csökkennek, ezért egyre inkább népszerűbbé válnak. Az ISDN útválasztó egy apró doboz, amelyet közvetlenül a helyi Ethernet hálózatunkra tudunk csatlakoztatni, és a többi útválasztóhoz vagy hídhez kapcsolódik. A benne található szoftverrel képes kommunikálni a PPP vagy más egyéb népszerű protokollokon keresztül.

Az útválasztó egy szabványos terminál adapternél sokkal nagyobb adatátvitelt tesz lehetővé, mivel a teljes szinkron ISDN kapcsolatot képes kihasználni.

Az ISDN útválasztókkal és hidakkal kapcsolatban az egyik legnagyobb problémát a különböző gyártók közti eltérések jelenthetik. Ha egy szolgáltatóhoz akarunk ezen a módon csatlakozni, akkor érdemes előzetesen egyeztetni az igényeinket velük.

Ha két helyi hálózati szegmenst akarunk összekapcsolni, mint például az otthoni és az irodai hálózatot, akkor ez a megoldás jár a legkevesebb karbantartási költséggel. Mivel ekkor mi magunk vásároljuk a kapcsolat mind a két oldalára a felszerelést, biztosak lehetünk benne, hogy az így létrehozott összeköttetés működni fog.

Például, ha egy otthon vagy a vállalat egy fiókjánál levő gépet akarjuk összekötni az igazgatóság hálózatával, akkor a következő felállást érdemes követnünk:

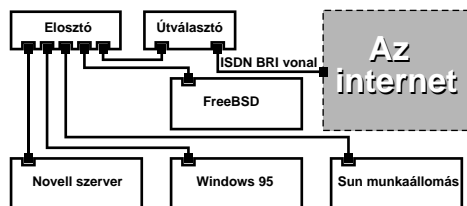
Példa 31-4. Egy otthoni vagy egy fiókbeli hálózat

A hálózat busz topológiájú és 10 Base 2 Ethernetet használ ("thinnet"). Ha szükséges, akkor az útválasztót egy AUI/10BT adó-vevővel csatlakoztassuk a hálózati kábelre.

Ha az otthoni vagy fiókbeli számítógép az egyedüli, akkor egy keresztkötésű sodrott érpár kábellel akár közvetlenül is csatlakozhatunk az útválasztóhoz.

Példa 31-5. Az igazgatósági iroda vagy egy másik helyi hálózat

A hálózat csillag topológiájú, és 10 Base T Ethernet kábelezésű ("sodrott érpár").



A legtöbb útválasztó/híd előnye, hogy *egyszerre 2 egymástól független PPP kapcsolatot tudunk felépíteni velük 2 egymástól független géppel*. Ezt a legtöbb terminál adapter nem támogatja, kivéve azok a (általában drága) típusok, amelyek két soros porttal rendelkeznek. Ezt ne tévesszük össze a csatornák nyálábolásával, az MPP-vel és a többivel.

Ez nagyon hasznos lehet például olyan esetekben, amikor van egy dedikált ISDN kapcsolatunk az irodában, amelyet ugyan szeretnénk megcsapolni, de nem szeretnénk a másik ISDN vonalat is elrabolni. Az irodában levő A útválasztó képes a dedikált B csatornájú kapcsolaton (64 Kb/mp) keresztül elérni az internetet, miközben a másik B csatornát ettől független adatkapcsolatra használja. A második B csatorna így használható betárcsázásra, kitércsázásra vagy a másik B csatornával együtt dinamikus nyálábolásra (MPP stb.) a nagyobb sávszélesség elérése érdekében.

Az Ethernetes híd nem IP alapú forgalmat is képes továbbítani, ezért rajta keresztül akár IPX vagy SPX és más egyéb protokollokat is használni tudunk.

31.9. Hálózati címfordítás

Írta: Chern Lee.

31.9.1. Áttekintés

A FreeBSD hálózati címfordításért felelős démonprogramja, a natd(8) (Network Address Translation daemon), a beérkező nyers IP csomagokat dolgozza fel, és a helyi gépek forráscímét kicserélve visszailleszti ezeket a csomagokat a kimenő folyamba. A natd(8) mindezt úgy teszi a forrás IP-címekkel és portokkal, hogy amikor az adat visszaérkezik, akkor képes lesz megmondani a csomag eredeti küldőjét és visszaküldeni neki a választ.

A hálózati címfordítást általában az internet-kapcsolatok megosztásánál alkalmazzuk.

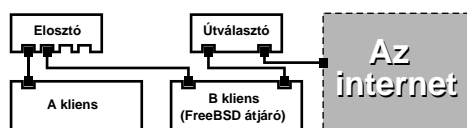
31.9.2. A hálózat felépítése

Az IPv4 világában egyre jobban fogyó IP-címek és az egyre növekvő számú, nagysebességre vágyó, például kábeles vagy DSL-es fogyasztók miatt az igény is egyre nagyobb az internet-kapcsolatok megosztására. Ha több számítógéppel szeretnénk egyetlen kapcsolaton és egy IP-címen keresztül kapcsolódni az internetre, akkor ehhez a natd(8) tökéletes választás.

Az esetek többségében a felhasználók egy kábeles vagy DSL vonalra csatlakoznak, melyhez egyetlen IP-cím tartozik, és ezen a gépen keresztül szeretnék elérni az internetet a helyi hálózaton levő többi gépről.

Ezt úgy tudjuk elérni, ha az internethez kapcsolódó FreeBSD-s gépet átjárónak állítjuk be. Ebben az átjáróban legalább két hálózati felületnek kell léteznie — az egyikkel az internetes útválasztóhoz, a másikkal pedig a helyi hálózathoz kapcsolódik. A belső hálózaton levő gépek egy hub vagy egy switch segítségével csatlakoznak egymáshoz.

Megjegyzés: Több módon is el tudjuk érni a belső hálózatról az internetet egy FreeBSD-s átjárón keresztül. Ebben a példában most csak olyan átjárókkal foglalkozunk, amelyekben legalább két hálózati kártya található.



Egy ehhez hasonló beállítás igen gyakori a megosztott internet-kapcsolatok esetében. A helyi hálózat egyik gépe csatlakozik az internetre. A többi gép ezen az "átjárón" keresztül éri el az internetet.

31.9.3. A rendszerbetöltő beállítása

A natd(8) működéséhez szükséges címfordítási támogatást a GENERIC típusú rendszermagok nem tartalmazzák, viszont a `/boot/loader.conf` megfelelő paraméterezésével a rendszer betöltése közben ezt hozzá tudjuk adni:

```
ipfw_load="YES"
ipdivert_load="YES"
```

Valamint a `net.inet.ip.fw.default_to_accept` változót állítsuk az 1 értékre.

```
net.inet.ip.fw.default_to_accept="1"
```

Megjegyzés: Ez utóbbi beállítást leginkább a tűzfal és a címfordítást végző átjáró próbálgatásakor érdemes alkalmazni. Ilyenkor ugyanis az ipfw(8) alapértelmezett módon az `allow ip from any to any` (minden forgalom engedélyezett) szabályt követi, és nem pedig a kevésbé barátságos `deny ip from any to any` (minden forgalom tiltott) szabályt. A rendszer újraindításakor így valamivel nehezebb lesz kizárnunk magunkat a szabályok megadása során.

31.9.4. A rendszermag beállítása

Amikor viszont nincs lehetőségünk modulok használatára, vagy szeretnénk minden igényelt funkciót beépíteni a rendszermagba, akkor a rendszermag beállításait tartalmazó állományban a következőket kell megadnunk:

```
options IPFIREWALL
options IPDIVERT
```

A fentiek mellett még ezeket a lehetőségeket tudjuk választani:

```
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_VERBOSE
```

31.9.5. A rendszerindítás beállítása

A tűzfal és a hálózati címfordítás beindításához a következőknek kell az `/etc/rc.conf` állományban lennie:

```
gateway_enable="YES" ❶
firewall_enable="YES" ❷
firewall_type="OPEN" ❸
natd_enable="YES"
natd_interface="fxp0" ❹
natd_flags="" ❺
```

- ❶ A gépet átjárónak állítja be. Hatása megegyezik a `sysctl net.inet.ip.forwarding=1` parancs kiadásával.
- ❷ A rendszer indításakor engedélyezi az `/etc/rc.firewall` állományban szereplő tűzfalszabályok használatát.
- ❸ Egy olyan előre definiált tűzfalat ad meg, amely alaphól mindent beenged. Az `/etc/rc.firewall` állományban találhatjuk a többi típust.
- ❹ Megadja, hogy melyik felületen továbbítsunk csomagokat az internet felé (ez a felület csatlakozik az internetre).
- ❺ Itt szerepel minden további paraméter, amelyet még az indításkor át kell adnunk a `natd(8)` démonnak.

Amikor megadjuk ezeket a beállításokat az `/etc/rc.conf` állományban, pontosan ugyanaz történik, mintha a `natd -interface fxp0` parancsot adtunk volna ki a rendszer indításakor. Ez tehát manuálisan is elindítható.

Megjegyzés: Ha túlságosan sok paramétert akarunk egyszerre beállítani `natd(8)` használatához, akkor akár egy külön konfigurációs állományt is megadhatunk. Ebben az esetben a konfigurációs állományt a következő módon kell megjelölni az `/etc/rc.conf` állományban:

```
natd_flags="-f /etc/natd.conf"
```

Ekkor a `/etc/natd.conf` állomány fogja tartalmazni a beállításokat, soronként egyet. Például a következő szakaszban ez lesz a tartalma:

```
redirect_port tcp 192.168.0.2:6667 6667
redirect_port tcp 192.168.0.3:80 80
```

A konfigurációs állományról és az `-f` opció használatával kapcsolatban olvassuk el a `natd(8)` man oldalát.

A helyi hálózaton mindegyik gépnek az RFC 1918 (<ftp://ftp.isi.edu/in-notes/rfc1918.txt>) által megadott privát IP-címterekből származó címet kell használnia, és az alapértelmezett átjárónak mindenhol a **natd** démont futtató gép IP-címét kell megadni.

Például a belső hálózaton található A és B kliensek IP-címei rendre `192.168.0.2` és `192.168.0.3`, míg a `natd(8)` démont futtató gép belső címe `192.168.0.1`. Az A és a B kliens alapértelmezett átjáróját a **natd** gépre, vagyis a `192.168.0.1` címre kell beállítanunk. A **natd** gép külső, avagy internetes felülete semmilyen további módosítást nem igényel a `natd(8)` működéséhez.

31.9.6. A portok átirányítása

A `natd(8)` alkalmazásának hátránya, hogy a belső hálózatra csatlakozó kliensek az internetről nem érhetőek el. Tehát a helyi hálózat kliensei képesek elérni a külvilágot, de az visszafelé már nem igaz. Ez akkor jelent igazából problémát, ha az egyik belső kliensen szolgáltatásokat akarunk futtatni. A probléma egyik egyszerű megoldása, ha a **natd** használatával az internet felől egyszerűen átirányítunk bizonyos portokat a megfelelő belső kliensre.

Például tegyük fel, hogy az A kliens egy IRC szerver, míg a B kliens egy webszervert futtat. Ez akkor fog működni, ha a szolgáltatásokhoz tartozó 6667 (IRC) és 80 (web) portokat átirányítjuk a hozzájuk tartozó gépek felé.

Ehhez a `natd(8)` démonnak a `-redirect_port` paramétert kell átadni. A pontos felírás így néz ki:

```
-redirect_port protokoll célIP:célPORT[-célPORT]
                [külsőIP:]külsőPORT[-külsőPORT]
                [távoliIP[:távoliPORT[-távoliPORT]]]
```

A fenti példában tehát ezt kell megadnunk:

```
-redirect_port tcp 192.168.0.2:6667 6667
-redirect_port tcp 192.168.0.3:80 80
```

Így az egyes külső `tcp` portokat átirányítjuk a belső hálózat gépei felé.

A `-redirect_port` paraméternek akár egész porttartományokat is megadhatunk. Például a `tcp 192.168.0.2:2000-3000 2000-3000` megadásával az összes 2000-től 3000-ig terjedő port csatlakozását leképezzük az A kliens 2000 és 3000 közti portjaira.

Ezek a beállítások a `natd(8)` közvetlen futtatásakor adhatóak meg, esetleg az `/etc/rc.conf` állományban az `natd_flags=""` opció keresztül, vagy egy külön konfigurációs állományban.

A többi beállítási lehetőséget a `natd(8)` man oldalán ismerhetjük meg.

31.9.7. A címek átirányítása

A címek átirányítása abban az esetben hasznos, amikor több IP-cím áll rendelkezésünkre, de ezek egy géphez tartoznak. Ilyenkor az `natd(8)` képes a belső hálózat egyes gépeihez saját külső IP-címet rendelni. A `natd(8)` a belső hálózat kliensei által küldött csomagokban kicseréli a címüket a megfelelő külső IP-címmel, illetve az ezekre a címekre érkező forgalmat továbbítja a megfelelő belső kliens irányába. Ezt a megoldást statikus hálózati címfordításnak is nevezzük. Például a `128.1.1.2` és a `128.1.1.3` IP-címek a **natd** démont futtató átjáróhoz tartoznak. A `128.1.1.1` cím használható a **natd** alapú átjáró külső IP-címeként, miközben a `128.1.1.2` és a `128.1.1.3` címeket a belső hálózaton elérhető A és B kliensek felé közvetítjük.

A `-redirect_address` felírása tehát a következő:

```
-redirect_address helyiIP publikusIP
```

helyiIP

A helyi hálózaton található kliens saját IP-címe.

publikusIP

A klienshez tartozó megfelelő külső IP-cím.

Az iménti példában a pontos paraméterek ezek lesznek:

```
-redirect_address 192.168.0.2 128.1.1.2
-redirect_address 192.168.0.3 128.1.1.3
```

A `-redirect_port` opcióhoz hasonlóan ez is megadható az `/etc/rc.conf` állományban az `natd_flags=""` beállításon keresztül vagy egy külön konfigurációs állományban. A címek átirányításával nincs szüksége a portok átirányítására, mivel az adott IP-címhez tartozó összes forgalmat átirányítjuk.

A **natd** démont futtató gépen a külső IP-címeket aktiválni kell és a külső felülethez kell rendelni. A `rc.conf(5)` man oldalon járhatunk utána, hogy mindezt hogyan is tudjuk megcsinálni.

31.10. Párhuzamos vonali IP (PLIP)

A párhuzamos vonali IP (Parallel Line IP, PLIP) a TCP/IP protokoll használatát valósítja meg párhuzamos porton keresztül. Olyan gépek számára lehet hasznos, amelyekben nincs hálózati kártya, vagy esetleg laptopoknál. Ebben a szakaszban a következőket tárgyaljuk:

- Párhuzamos (laplink) kábel készítése
- Két számítógép összekapcsolása a PLIP segítségével

31.10.1. Párhuzamos kábel készítése

Párhuzamos kábelt a legtöbb számítástechnikai boltban tudunk vásárolni. Ha mégsem tudnánk sehol sem beszerezni, vagy egyszerűen tudni szeretnénk, hogyan lehet ilyet készíteni, akkor az alábbi táblázatban láthatjuk, hogy miként tudunk egy hétköznapi nyomtatókábelt átalakítani a céljainkra.

Táblázat 31-1. A párhuzamos kábel hálózati használatra alkalmas bekötése

A-név	A-vég	B-vég	Leírás	Post/Bit
-------	-------	-------	--------	----------

A-név	A-vég	B-vég	Leírás	Post/Bit
DATA0 -ERROR	2 15	15 2	Adat	0/0x01 1/0x08
DATA1 +SLCT	3 13	13 3	Adat	0/0x02 1/0x10
DATA2 +PE	4 12	12 4	Adat	0/0x04 1/0x20
DATA3 -ACK	5 10	10 5	Vál. imp.	0/0x08 1/0x40
DATA4 BUSY	6 11	11 6	Adat	0/0x10 1/0x80
GND	18-25	18-25	Föld	-

31.10.2. A PLIP beállítása

Először is szereznünk kell valahonnan egy laplink kábelt. Ha ez megvan, akkor mind a két gépen ellenőrizzük, hogy a rendszermag tartalmazza az lpt(4) meghajtót:

```
# grep lp /var/run/dmesg.boot
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
```

A párhuzamos portnak megszakítással vezéreltnek kell lennie ("interrupt driven"), és az /boot/device.hints állományban szerepelnie kell nagyjából a következő soroknak:

```
hint.ppc.0.at="isa"
hint.ppc.0.irq="7"
```

Ezután nézzük meg, hogy a rendszermag beállításait tartalmazó állományban megjelenik-e a device plip sor, vagy a plip.ko modul betöltődött-e. Akármelyik is történt, a párhuzamos hálózati felület most már a rendelkezésünkre áll, és az ifconfig(8) paranccsal ezt meg is tudjuk nézni:

```
# ifconfig plip0
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
```

A laplink kábelt csatlakoztassuk mind a két számítógéphez.

Mind a két a hálózati felület paramétereit root felhasználóként hangoljuk be. Például, ha az *egyikgép* nevű gépet akarjuk a *másikgép* nevű géphez csatlakoztatni:

```
egyikgép <-----> másikgép
IP-cím      10.0.0.1      10.0.0.2
```

Az *egyikgép* felületét így állítsuk be:

```
# ifconfig plip0 10.0.0.1 10.0.0.2
```

A *másikgép* felületét így állítsuk be:

```
# ifconfig plip0 10.0.0.2 10.0.0.1
```

Ezt követően már egy működő kapcsolatnak kell felépülnie. Az egyéb részletek kapcsán az lp(4) és az lpt(4) man oldalait nézzük át.

Ezt a két gépet vegyük fel az /etc/hosts állományba is:

```

127.0.0.1          localhost.saját.tartomány localhost
10.0.0.1          egyikgép.saját.tartomány egyikgép
10.0.0.2          másikkép.saját.tartomány

```

A kapcsolat működőképességéről úgy tudunk meggyőződni, ha az egyik gépről megpróbáljuk pingelni a másikat. Például az *egyikgép* esetében:

```

# ifconfig plip0
plip0: flags=8851<UP,POINTOPOINT,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.0.0.1 --> 10.0.0.2 netmask 0xff000000
# netstat -r
Routing tables

Internet:

Destination          Gateway              Flags      Refs      Use      Netif Expire
máskép               egyikgép            UH          0          0        plip0

# ping -c 4 máskép
PING máskép (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: icmp_seq=0 ttl=255 time=2.774 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=255 time=2.530 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=255 time=2.556 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=255 time=2.714 ms

--- máskép ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.530/2.643/2.774/0.103 ms

```

31.11. Az IPv6

Eredetileg írta: Aaron Kaplan. Átszervezte és kiegészítette: Tom Rhodes. Tovább bővítette: Brad Davis.

Az IPv6 (másik néven az IPng, vagy a “az internet következő generációs protokollja”, “IP next generation”) a jól ismert IP protokoll (avagy az IPv4) új változata. Hasonlóan a jelenleg működő összes többi BSD rendszerhez, a FreeBSD is tartalmazza a KAME IPv6 referencia implementációt. Ezért ha ezzel szeretnénk kísérletezni, akkor ehhez a FreeBSD minden eszköz biztosít számunkra. Ez a szakasz az IPv6 beállítását és használatát mutatja be.

Az 1990-es évek elején az IPv4-es címterek rohamos mértékű kimerülését figyelték meg. Az internet jelenlegi bővülési üteme mellett két nagyobb aggodalomnak adott okot:

- A címek elfogyása. Napjainkban efelől egyre kevesebb a kétség, mivel az RFC 1918 által megfogalmazott privát címterek (10.0.0.0/8, 172.16.0.0/12, és 192.168.0.0/16), valamint a hálózati címfordítás (Network Address Translation, NAT) használata igen elterjedt.
- Az útválasztási táblázatok méretének növekedése. Ez még manapság is aggasztó.

Az IPv6 ezeket és még más egyéb problémákat a következő módon igyekszik megoldani:

- A 128 bites címtér használata. Más szóval, elméletben összesen 340 282 366 920 938 463 374 607 431 768 211 456 darab címet képes kiosztani. Ez azt jelenti, hogy bolygónk minden egyes négyzetméterére megközelítőleg $6,67 \cdot 10^{27}$ IPv6 típusú cím jut.

- Az útválasztók a saját táblázataikban csak a hálózatok összevont címeit tárolják el, ezáltal egy átlagos útválasztási táblázatban található bejegyzések száma 8192 alá csökken.

Az IPv6 emellett még rengeteg más előnyös lehetőséget is kínál:

- A címek automatikus beállítása (lásd RFC 2462 (<http://www.ietf.org/rfc/rfc2462.txt>))
- Anycast (bárkiküldés, vagyis “egy a sokból”)
- Kötelező (mandatory) multicast
- IPsec (IP szintű védelem)
- Egyszerűsített fejléc
- Mobil IP
- IPv6-IPv4 közti átjárhatóság

Ha mindezekről többet szeretnénk megtudni, akkor erre érdemes továbblépünk:

- Az IPv6 áttekintése a playground.sun.com (<http://playground.sun.com/pub/ipng/html/ipng-main.html>) honlapon
- KAME.net (<http://www.kame.net>)

31.11.1. Az IPv6 címek háttere

Az IPv6 címeknek több típusa létezik: a unicast (egyesküldés), az anycast (bárkiküldés) és a multicast (többesküldés).

A unicasthez használt címek jól ismert címek. Az így elküldött csomag pontosan ahhoz a felülethez érkezik meg, amelyhez az adott cím tartozik.

Az anycasthez használt címek felírásukban tökéletesen megegyeznek a unicast esetével, de valójában felületek egy csoportját címezik. Az anycastre beállított címekre küldött csomagok mindig a(z útválasztó szerinti) legközelebb levő felülethez érkeznek meg. Az anycastet az útválasztók számára találták ki.

A multicasthez használt címek felületek egy csoportját nevezik meg. A multicast címekre érkező csomagokat a csoport minden egyes tagja megkapja.

Megjegyzés: Az IPv4 esetében az üzenetszórásra szánt (általában az xxx.xxx.xxx.255 formátumú) címeket az IPv6 esetében multicast címekkel fejezzük ki.

Táblázat 31-2. Fenntartott IPv6 címek

IPv6 cím	Az előtag hossza (bitekben)	Leírás	Megjegyzés
::	128 bit	nem specifikált	Vö. a 0.0.0.0 címmel az IPv4 esetében.
:::1	128 bit	saját cím	Vö. a 127.0.0.1 címmel az IPv4 esetében.

IPv6 cím	Az előtag hossza (bitekben)	Leírás	Megjegyzés
::00:xx:xx:xx:xx	96 bit	IPv4 beágyazása	Az alsó 32 bit egy IPv4 formátumú cím. Ezt "IPv4 kompatibilis IPv6 címnek" is nevezik.
::ff:xx:xx:xx:xx	96 bit	IPv4-re leképzett IPv6 címek	Az alsó 32 bit egy IPv4 címet jelöl. Olyan gépeknél használatos, amelyek nem támogatják az IPv6 protokollt.
fe80:: - feb::	10 bit	helyi összeköttetés	Vö. az IPv4 loopback címeivel.
fec0:: - fef::	10 bit	helyi cím	
ff::	8 bit	multicast	
001 (2-es alapú)	3 bit	globális unicast	Az összes globális unicast címet ebből a tartományból osztjuk ki. Az első 3 bit értéke "001".

31.11.2. Az IPv6 címek olvasása

Az IPv6 címek kanonikus formája így ábrázolható: x:x:x:x:x:x:x:x, ahol mindegyik "x" egy 16 bites hexadecimalis érték. Például: FEBC:A574:382B:23C1:AA49:4592:4EFE:9982.

Gyakran a címek hosszú nullákból álló sorozatokat tartalmaznak, ezért mindegyik ilyen sorozatot rövidíteni tudjuk a "::" jelöléssel. Rajtuk kívül még az egyes hexadecimalis csoportokban a bevezető nullák is elhagyhatóak. Például az fe80::1 cím kanonikus formája: fe80:0000:0000:0000:0000:0000:0000:0001.

A harmadik forma szerint az utolsó 32 bites részt írjuk fel a megszokott (decimális) IPv4 stílusú pontozással, ahol tehát a "." választja el a tagokat. Így például a 2002::10.0.0.1 felírás a 2002:0000:0000:0000:0000:0000:0a00:0001 kanonikus (hexadecimalis) ábrázolásnak feleltethető meg, ami pedig egyszerűen 2002::a00:1 alakban is megadható.

Mostanra már minden bizonnyal a kedves olvasó érteni fogja a következőt:

```
# ifconfig
```

```
rl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    inet 10.0.0.10 netmask 0xffffffff broadcast 10.0.0.255
    inet6 fe80::200:21ff:fe03:8e1%rl0 prefixlen 64 scopeid 0x1
    ether 00:00:21:03:08:e1
    media: Ethernet autoselect (100baseTX )
    status: active
```

A fe80::200:21ff:fe03:8e1%rl0 cím az automatikusan beállított helyi összeköttetés címe. Ez az automatikus beállítás részeként a MAC-címből jött létre.

Az IPv6 címek szerkezetéről további részleteket az RFC 3513 (<http://www.ietf.org/rfc/rfc3513.txt>)-ban találunk.

31.11.3. Kapcsolódás

Jelenleg négy módon tudunk más IPv6-os géphez és hálózathoz csatlakozni:

- Kérjünk a hálózati elérésünkért felelős illetékesektől IPv6 alapú hálózatot. A részletek tekintetében vegyük fel a kapcsolatot az internet-szolgáltatónkkal.
- A SixXS (<http://www.sixxs.net>) a világ minden táján kínál végpontokkal rendelkező tunneleket.
- Egy 6-ból-4 (RFC 3068 (<http://www.ietf.org/rfc/rfc3068.txt>)) típusú tunnellal.
- Ha betárcsázós kapcsolatunk van, akkor használjuk a `net/freenet6` portot.

31.11.4. A nevek feloldása az IPv6 világában

IPv6 alatt régebben két típusa volt a nevek feloldásáért felelős rekordoknak. Az IETF az A6 rekordokat időközben elavultnak nyilvánította. Ezért manapság már az AAAA rekordok tekinthetők szabványosnak.

Az AAAA rekordok használata magától értetődik. A hálózati nevükhöz az alábbi módon tudunk IPv6 címet rendelni az elsődleges zónát leíró állományban:

```
SAJÁTNEV          AAAA      SAJÁTIPv6CÍM
```

Ha nem rendelkezünk saját névfeloldási zónával, akkor erre kérjük meg a névfeloldást végző szolgáltatónkat. A **bind** jelenlegi változatai (8.3 és 9), valamint a `dns/djbdns` (IPv6 támogatására vonatkozó javítással) támogatják az AAAA rekordokat.

31.11.5. Az `/etc/rc.conf` szükséges módosításai

31.11.5.1. Az IPv6 kliensek beállításai

Ezek a beállítások egy helyi hálózaton levő gépre vonatkoznak, nem pedig egy útválasztóra. Az `rtsol(8)` az alábbi megadásával fogja automatikusan beállítani a felületeinket a rendszer indításakor:

```
ipv6_enable="YES"
```

Ha az `fxp0` felülethez statikusan akarunk IP-címet rendelni, például a

2001:471:1f11:251:290:27ff:fee0:2093 címet, akkor ehhez a következőt kell megadni:

```
ipv6_ifconfig_fxp0="2001:471:1f11:251:290:27ff:fee0:2093"
```

Az `/etc/rc.conf` állományban az alapértelmezett átjárót a következő módon tudjuk a 2001:471:1f11:251::1 címre beállítani:

```
ipv6_defaultrouter="2001:471:1f11:251::1"
```

31.11.5.2. Az IPv6 útválasztók és átjárók beállítása

Itt most a tunnelt biztosító szolgáltató által mutatott irányt követjük, és olyan formára alakítjuk, amely megmarad az újraindítás után is. A rendszer indításakor az `/etc/rc.conf` állományban valami ilyesmit kell megadni a járat visszaállításához:

Soroljuk fel a beállítandó általános tunnel alapú felületeket, ilyen lehet például a `gif0`:

```
gif_interfaces="gif0"
```

A felületnek állítsunk be egy helyi végpontot a `SAJÁT_IPv4_CÍM` megadásával, valamint egy távoli végpontot a `TÁVOLI_IPv4_CÍM` megadásával:

```
gifconfig_gif0="SAJÁT_IPv4_CÍM TÁVOLI_IPv4_CÍM"
```

Az IPv6 tunnelünk végpontjához kapott cím aktiválásához az alábbi kell még megadnunk:

```
ipv6_ifconfig_gif0="SAJÁT_KAPOTT_IPv6_TUNNEL_VÉGPONTJÁNAK_CÍME"
```

Ezután már csak az alapértelmezett útvonalat kell beállítani az IPv6 számára. Ez az IPv6 járat másik oldala:

```
ipv6_defaultrouter="SAJÁT_IPv6_TÁVOLI_TUNNEL_VÉGPONTJÁNAK_CÍME"
```

31.11.5.3. Az IPv6 tunnel beállításai

Amennyiben a szerver IPv6 alapú forgalmat közvetít a hálózatunk és a világ között, az `/etc/rc.conf` állományba a következőt kell felvennünk:

```
ipv6_gateway_enable="YES"
```

31.11.6. Az útválasztók kihirdetése és automatikus konfigurációja

Ebben a szakaszban az `rtadvd(8)` beállításával fogjuk az alapértelmezett IPv6 útvonalat kihirdetni.

Az `rtadvd(8)` engedélyezéséhez az alábbi sort kell betennünk az `/etc/rc.conf` állományba:

```
rtadvd_enable="YES"
```

Emellett még fontos megadnunk azt a felületet, ahol az IPv6 útválasztó kérelmezését végezzük. Ha erre a feladatra például az `fxp0` felületet választjuk, akkor erről az `rtadvd(8)` így értesíthető:

```
rtadvd_interfaces="fxp0"
```

Most pedig készítenünk kell hozzá egy konfigurációt is, vagyis az `/etc/rtadvd.conf` állományt. Íme erre egy példa:

```
fxp0:\n      :addrs#1:addr="2001:471:1f11:246::":prefixlen#64:tc=ether:
```

Az `fxp0` felületet természetesen cseréljük ki a sajátunkkal.

Ezután a `2001:471:1f11:246::` címre helyére írjuk be a saját kiosztásunk előtagját.

Egy egész /64 alhálózat esetén nem is kell többet megadni. Minden más helyzetben az előtag hosszára prefixlen# vonatkozó értéket is be kell még állítanunk.

31.12. Az Aszinkron adatátviteli mód (ATM)

Készítette: Harti Brandt.

31.12.1. A klasszikus IP-címek beállítása ATM felett (állandó)

A klasszikus IP ATM felett (Classical IP over ATM, CLIP) a legegyszerűbb módszer az IP-címek használatára az Aszinkron adatátviteli móddal (Asynchronous Transfer Mode, ATM) együtt. Kapcsolt és állandó kapcsolatok (Switched Virtual Channel, SVC és Permanent Virtual Channel, PVC) esetén egyaránt megfelelő. Ebben a szakaszban ez utóbbival fogunk foglalkozni.

31.12.1.1. A teljesen hálószerű konfigurációk

A CLIP beállítását állandó csatornákon például úgy tudjuk megoldani, ha az összes gépet külön ezekre a célokra szánt állandó csatornákkal összekapcsoljuk egymással. Ez az egyszerű megoldás azonban nagyobb számú gép esetében már nem eléggé hatékony. A következő példában csupán négy gépet kötünk hálózatba, melyik mindegyike egy ATM kártyával csatlakozik az ATM hálózatra. Ehhez elsőként tervezzük meg az IP-címek kiosztását és a gépek közti ATM kapcsolatokat. A példában ez az alábbiak szerint alakul:

Gép	IP-cím
A-gep	192.168.173.1
B-gep	192.168.173.2
C-gep	192.168.173.3
D-gep	192.168.173.4

A teljes hálózat felépítéséhez minden egyes pár között egy-egy ATM kapcsolatra lesz szükségünk:

Gépek	VPI.VCI pár
A-gep - B-gep	0.100
A-gep - C-gep	0.101
A-gep - D-gep	0.102
B-gep - C-gep	0.103
B-gep - D-gep	0.104
C-gep - D-gep	0.105

A kapcsolatok egyes végein szereplő VPI és VCI értékek természetesen eltérhetnek, de ezeket mi most az egyszerűség kedvéért egyenlőnek tekintettük. A következő lépésben minden gépen állítsuk be az ATM felület:

```
A-gep# ifconfig hatm0 192.168.173.1 up
B-gep# ifconfig hatm0 192.168.173.2 up
C-gep# ifconfig hatm0 192.168.173.3 up
D-gep# ifconfig hatm0 192.168.173.4 up
```

Ha feltételezzük, hogy minden gépen a `hatm0` az ATM felület neve. Most pedig az A-gep-en állítsuk be az állandó csatornákat. (Itt most feltesszük, hogy az ATM switch-eken mindezt már elvégeztük. A switch kézikönyvében erről részletesebb leírást is találhatunk.)

```
A-gep# atmconfig natm add 192.168.173.2 hatm0 0 100 llc/snap ubr
A-gep# atmconfig natm add 192.168.173.3 hatm0 0 101 llc/snap ubr
A-gep# atmconfig natm add 192.168.173.4 hatm0 0 102 llc/snap ubr

B-gep# atmconfig natm add 192.168.173.1 hatm0 0 100 llc/snap ubr
B-gep# atmconfig natm add 192.168.173.3 hatm0 0 103 llc/snap ubr
B-gep# atmconfig natm add 192.168.173.4 hatm0 0 104 llc/snap ubr

C-gep# atmconfig natm add 192.168.173.1 hatm0 0 101 llc/snap ubr
C-gep# atmconfig natm add 192.168.173.2 hatm0 0 103 llc/snap ubr
C-gep# atmconfig natm add 192.168.173.4 hatm0 0 105 llc/snap ubr

D-gep# atmconfig natm add 192.168.173.1 hatm0 0 102 llc/snap ubr
D-gep# atmconfig natm add 192.168.173.2 hatm0 0 104 llc/snap ubr
D-gep# atmconfig natm add 192.168.173.3 hatm0 0 105 llc/snap ubr
```

Természetesen nem csak UBR használható, hanem minden más olyan forgalmazási beállítás, amit az ATM kártyáink ismernek. Itt most a forgalmi beállítás nevét a hozzá tartozó konkrét paraméterek követik. Az `atmconfig(8)` segédprogram használatához így kérhetünk segítséget:

```
# atmconfig help natm add
```

Olvassuk el az `atmconfig(8)` man oldalát.

Ugyanez a beállítás az `/etc/rc.conf` állomány használatával is elvégezhető. Az A-gep esetében mindez így nézne ki:

```
network_interfaces="lo0 hatm0"
ifconfig_hatm0="inet 192.168.173.1 up"
natm_static_routes="B-gep C-gep D-gep"
route_B-gep="192.168.173.2 hatm0 0 100 llc/snap ubr"
route_C-gep="192.168.173.3 hatm0 0 101 llc/snap ubr"
route_D-gep="192.168.173.4 hatm0 0 102 llc/snap ubr"
```

A CLIP útvonalak pillanatnyi állapota így kérdezhető le:

```
A-gep# atmconfig natm show
```

31.13. A Közös cím redundancia protokoll (CARP)

Írta: Tom Rhodes.

A Közös cím redundancia protokoll (Common Address Redundancy Protocol, avagy CARP) segítségével több gép képes egyazon IP-címen osztozni. Bizonyos konfigurációkban ez a terhelés elosztására (terhelés-kiegyenlítésre) vagy a rendelkezésre állás növelésére (hibatűrésre) alkalmazható. A benne szereplő gépek akár eltérő IP-címmel is rendelkezhetnek, ahogy azt majd a példában is láthatjuk.

A CARP támogatásának engedélyezéséhez a FreeBSD rendszermagját a következő beállítással kell újrafordítanunk:

```
device carp
```

A CARP által biztosított lehetőségek ezután már elérhetőek, és számos `sysctl` változón keresztül állíthatóak:

Változó	Leírás
<code>net.inet.carp.allow</code>	A beérkező CARP csomagok elfogadása. Alapértelmezés szerint engedélyezett.
<code>net.inet.carp.preempt</code>	Ezzel a beállítással az adott gépen az összes CARP felület leáll, ha közülük bármelyik is működésképtelenné válik. Alapértelmezés szerint tiltott.
<code>net.inet.carp.log</code>	A 0 értékkel kikapcsoljuk a naplózást. Az 1 értékkel a rossz CARP csomagok naplózását engedélyezzük. Az ettől nagyobb értékek esetén pedig a CARP felületek változásait naplózzuk. Az alapértelmezett értéke az 1.
<code>net.inet.carp.arbalance</code>	Az ARP protokoll segítségével próbálja meg a helyi hálózati forgalmat mentesíteni a terheléstől. Alapértelmezés szerint tiltott.
<code>net.inet.carp.suppress_preempt</code>	Ez a változó írásvédett, és a megszakítás elnyomásának állapotát mutatja. A megszakítás elnyomható, ha a felület egyik linkje nem működik. A 0 érték arra utal, hogy a megszakítást nem nyomták el. Minden probléma növeli ennek a változónak az értékét.

A CARP eszközök maguk az `ifconfig` paranccsal készíthetők el:

```
# ifconfig carp0 create
```

Egy valós környezetben az ilyen felületeknek egy VHID néven ismert egyedi azonosítóval kell rendelkezniük. Ez a VHID vagy más néven a virtuális gépazonosító (azaz Virtual Host Identification) fogja a gépünket a hálózat többi elemétől megkülönböztetni.

31.13.1. A CARP felhasználása a rendelkezésre állás javításában

A CARP használatának egyik módja, ahogy arra már korábban is utaltunk, a szerverek rendelkezésre állásának feljavítása. Ebben a példában három géppel fogunk hibatűrést biztosítani, melyik mindegyike egyedi IP-címmel rendelkezik és ugyanazt a webes tartalmat szolgáltatják. A gépeket egy Round Robin rendszerű (körbejáró) névfeloldással együtt használjuk. A tartalék gépünknek lesz még további két CARP felülete, külön a szerver IP-címeihez tartozó egyes webes tartalmakhoz. Amikor valami meghibásodik, a tartalék szerver átveszi a meghibásodott gép IP-címét. Ilyenkor a hiba teljesen észrevétlen marad a felhasználók számára. A tartalék szerveren a többi szerverrel egyező tartalomnak és szolgáltatásoknak kell megjelennie, hogy bármikor át tudja tőlük venni a forgalmat.

A hálózati neveiktől és a virtuális azonosítóiktól eltekintve a két gépet ugyanúgy kell beállítani. Ebben a példában a gépeket most az `a-gep.minta.org` és `b-gep.minta.org` nevekkel láttuk el. Először is a CARP beállításához el kell helyeznünk a megfelelő hivatkozásokat az `rc.conf` állományban. Az `a-gep.minta.org` esetében az `rc.conf` állomány a következő sorokat tartalmazza:

```
hostname="a-gep.minta.org"
ifconfig_fxp0="inet 192.168.1.3 netmask 255.255.255.0"
cloned_interfaces="carp0"
ifconfig_carp0="vhid 1 pass testpass 192.168.1.50/24"
```

Miközben a `b-gep.minta.org` az `rc.conf` állományában ezeket adjuk meg:

```
hostname="b-gep.minta.org"
ifconfig_fxp0="inet 192.168.1.4 netmask 255.255.255.0"
cloned_interfaces="carp0"
ifconfig_carp0="vhid 2 pass testpass 192.168.1.51/24"
```

Megjegyzés: Nagyon fontos, hogy az `ifconfig` parancs `pass` paraméterével megadott jelszavak megegyezzenek. A `carp` eszközök csak a megfelelő jelszót birtokló gépeket fogadják el. A virtuális gépazonosítónak azonban minden esetben el kell térnie.

A harmadik, `szolgaltato.minta.org` címmel rendelkező gépet fogjuk felkészíteni az előbbi gépek meghibásodására felkészíteni. Ennek a gépnek két `carp` eszközre lesz szüksége, melyek az egyes gépeket kezelik. Az ehhez illeszkedő sorok valahogy így fognak kinézni az `rc.conf` állományban:

```
hostname="szolgaltato.minta.org"
ifconfig_fxp0="inet 192.168.1.5 netmask 255.255.255.0"
cloned_interfaces="carp0 carp1"
ifconfig_carp0="vhid 1 advskew 100 pass testpass 192.168.1.50/24"
ifconfig_carp1="vhid 2 advskew 100 pass testpass 192.168.1.51/24"
```

Két `carp` eszköz használatával a `szolgaltato.minta.org` képes észlelni és átvenni bármelyik olyan gép IP-címét, amely nem válaszol.

Megjegyzés: Az alap FreeBSD rendszermag használata esetén *előfordulhat*, hogy a megszakítás (a “preemption” opció) engedélyezett. Amennyiben így lenne, a `szolgaltato.minta.org` nem fogja minden esetben fogja rendesen visszaadni az IP-címet az eredeti tulajdonosának. Ilyenkor a rendszergazdának kell ezt manuálisan megtennie. Tehát a következő parancsot kell kiadnia a `szolgaltato.minta.org` gépen:

```
# ifconfig carp0 down && ifconfig carp0 up
```

Ezt az adott géphez tartozó `carp` felülettel kell megcsinálni.

Innentől a CARP már teljesen engedélyezhető és készen áll a tesztelésre. A teszteléshez vagy a hálózati rendszert kell újraindítani, vagy a gépeket.

További információkat a `carp(4)` man oldalán találhatunk.

V. Függelék

Függelék A. A FreeBSD beszerzése

A.1. CD és DVD kiadók

A.1.1. Kiskereskedelmi dobozos termékek

A FreeBSD beszerezhető számos kiskereskedőtől dobozos termék formájában is (FreeBSD CD-k, egyéb szoftverek és nyomtatott dokumentáció):

- CompUSA
WWW: <http://www.compusa.com/>
- Frys Electronics
WWW: <http://www.frys.com/>

A.1.2. CD- és DVD-készletek

FreeBSD CD- és DVD-készletek rengeteg helyről rendelhetőek:

- FreeBSD Mall, Inc.
700 Harvest Park Ste F
Brentwood, CA 94513
Egyesült Államok
Telefon: +1 925 240-6652
Fax: +1 925 674-0821
e-mail: info@freebsdmail.com
WWW: <http://www.freebsdmail.com/>
- Dr. Hinner EDV
St. Augustinus-Str. 10
D-81825 München
Németország
Telefon: (089) 428 419
WWW: <http://www.hinner.de/linux/freebsd.html>
- Ikarios
22-24 rue Voltaire
92000 Nanterre
Franciaország
WWW: <http://ikarios.com/form/#freebsd>
- JMC Software
Írország
Telefon: 353 1 6291282
WWW: <http://www.thelinuxmall.com>
- The Linux Emporium
Hilliard House, Lester Way

Wallingford
OX10 9TA
Egyesült Királyság
Telefon: +44 1491 837010
Fax: +44 1491 837016
WWW: <http://www.linuxemporium.co.uk/products/bsd/>

- Linux+ DVD Magazine
Lewartowskiego 6
Warsaw
00-190
Lengyelország
Telefon: +48 22 860 18 18
e-mail: <editors@lpmagazine.org>
WWW: <http://www.lpmagazine.org/>
- Linux System Labs Australia
21 Ray Drive
Balwyn North
VIC - 3104
Ausztrália
Telefon: +61 3 9857 5918
Fax: +61 3 9857 8974
WWW: <http://www.lsl.com.au>
- LinuxCenter.Ru
Galernaya utca, 55
Szentpétervár
190000
Oroszország
Telefon: +7-812-3125208
e-mail: <info@linuxcenter.ru>
WWW: <http://linuxcenter.ru/shop/freebsd>

A.1.3. Terjesztők

Ha viszonteladók vagyunk és szeretnénk CD-s FreeBSD termékeket forgalmazni, akkor az alábbi terjesztők valamelyikével vegyük fel a kapcsolatot:

- Cylogistics
809B Cuesta Dr., #2149
Mountain View, CA 94040
Egyesült Államok
Telefon: +1 650 694-4949
Fax: +1 650 694-4953
e-mail: <sales@cylogistics.com>
WWW: <http://www.cylogistics.com/>
- Ingram Micro

1600 E. St. Andrew Place
Santa Ana, CA 92705-4926
Egyesült Államok
Telefon: 1 (800) 456-8000
WWW: <http://www.ingrammicro.com/>

- Kudzu, LLC
7375 Washington Ave. S.
Edina, MN 55439
Egyesült Államok
Telefon: +1 952 947-0822
Fax: +1 952 947-0876
e-mail: <sales@kudzunterprises.com>
- LinuxCenter.Kz
Uszty-Kamenogorszk
Kazahsztán
Telefon: +7-705-501-6001
e-mail: <info@linuxcenter.kz>
WWW: <http://linuxcenter.kz/page.php?page=fr>
- LinuxCenter.Ru
Galernaya utca, 55
Szentpétervár
190000
Oroszország
Telefon: +7-812-3125208
e-mail: <info@linuxcenter.ru>
WWW: <http://linuxcenter.ru/freebsd>
- Navarre Corp
7400 49th Ave South
New Hope, MN 55428
Egyesült Államok
Telefon: +1 763 535-8333
Fax: +1 763 535-0341
WWW: <http://www.navarre.com/>

A.2. FTP oldalak

A FreeBSD hivatalos forrásai anonim FTP-n keresztül is elérhetőek különféle tükrözésekről. Az <ftp://ftp.FreeBSD.org/pub/FreeBSD/> oldal ugyan jó minőségű kapcsolattal rendelkezik és rengeteg felhasználót is enged egyidejűleg kapcsolódni, azonban valószínűleg jobban járunk, ha egy “hozzánk közelebbi” tükrözést választunk (különösen abban az esetben, amikor mi magunk is egy tükrözést akarunk készíteni).

A FreeBSD tükrözések adatbázisá (<http://mirrorlist.FreeBSD.org/>)ban az itt megtalálhatónál sokkal pontosabb leltárt kaphatunk az elérhető tükrözésekről, mivel közvetlenül a névfeloldás segítségével állapítja meg a szükséges adatokat és nem egy rögzített listát tárol.

Emellett az alábbi tükrözésekről a FreeBSD elérhető anonim FTP-n keresztül is. Amennyiben az anonim FTP használata mellett döntenénk, igyekezzünk a hozzánk legközelebb levő szervert használni. Az "Elsődleges tükrözésekként" feltüntetett oldalak általában a teljes FreeBSD archívumot tartalmazzák (az összes jelenleg elérhető változatot az összes architektúrára), de a környékünkön vagy országunkban elhelyezkedő tükörszerverekről többnyire gyorsabban tudunk majd letölteni. A regionális oldalakon gyakorta csak a népszerűbb architektúrákon futó népszerűbb változatokat találjuk meg, nem a teljes FreeBSD archívumot. Minden szerver elérhető anonim FTP-vel, de közülük néhány még további más módszereket is támogat. Az egyes oldalak által ismert konkrét módszereket a nevük után zárójelben közöljük.

Központi szerverek, Elsődleges tükrözések, Amerikai Egyesült Államok, Ausztrália, Ausztria, Brazília, Cseh Köztársaság, Dánia, Dél-Afrika, Egyesült Királyság, Észtország, Finnország, Franciaország, Görögország, Hollandia, Hongkong, Írország, Izland, Japán, Kanada, Kína, Korea, Lengyelország, Lettország, Litvánia, Németország, Norvégia, Örményország, Oroszország, Spanyolország, Svájc, Svédország, Szaúd-Arábia, Szlovák Köztársaság, Szlovénia, Tajvan, Új-Zéland, Ukrajna.

(Dátum: UTC)

Központi szerverek

- <ftp://ftp.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp.FreeBSD.org/pub/FreeBSD/>))

Elsődleges tükrözések

Bármilyen gond esetén a következő címet kell értesíteni: [<mirror-admin@FreeBSD.org>](mailto:mirror-admin@FreeBSD.org).

- <ftp://ftp1.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp4.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp4.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp5.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp10.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp10.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp10.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp11.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp13.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp14.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp14.FreeBSD.org/pub/FreeBSD/>))

Amerikai Egyesült Államok

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@us.FreeBSD.org>](mailto:hostmaster@us.FreeBSD.org).

- <ftp://ftp1.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.us.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp4.us.FreeBSD.org/pub/FreeBSD/>) / [httpv6 \(http://ftp4.us.FreeBSD.org/pub/FreeBSD/\)](http://ftp4.us.FreeBSD.org/pub/FreeBSD/))
- <ftp://ftp5.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp10.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp11.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp13.us.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp13.us.FreeBSD.org/pub/FreeBSD/>) / rsync)
- <ftp://ftp14.us.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp14.us.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp15.us.FreeBSD.org/pub/FreeBSD/> (ftp)

Ausztrália

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@au.FreeBSD.org>](mailto:hostmaster@au.FreeBSD.org).

- <ftp://ftp.au.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.au.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.au.FreeBSD.org/pub/FreeBSD/> (ftp)

Ausztria

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@at.FreeBSD.org>](mailto:hostmaster@at.FreeBSD.org).

- <ftp://ftp.at.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp.at.FreeBSD.org/pub/FreeBSD/>) / [httpv6 \(http://ftp.at.FreeBSD.org/pub/FreeBSD/\)](http://ftp.at.FreeBSD.org/pub/FreeBSD/))

Brazília

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@br.FreeBSD.org>](mailto:hostmaster@br.FreeBSD.org).

- <ftp://ftp2.br.FreeBSD.org/FreeBSD/> (ftp / http (<http://ftp2.br.FreeBSD.org/>))
- <ftp://ftp3.br.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)
- <ftp://ftp4.br.FreeBSD.org/pub/FreeBSD/> (ftp)

Cseh Köztársaság

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@cz.FreeBSD.org>](mailto:hostmaster@cz.FreeBSD.org).

- <ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/>) / [http](http://ftp.cz.FreeBSD.org/pub/FreeBSD/) (<http://ftp.cz.FreeBSD.org/pub/FreeBSD/>) / [httpv6](http://ftp2.cz.FreeBSD.org/pub/FreeBSD/) (<http://ftp2.cz.FreeBSD.org/pub/FreeBSD/>) / rsync / rsyncv6)
- <ftp://ftp2.cz.FreeBSD.org/pub/FreeBSD/> (ftp / [http](http://ftp2.cz.FreeBSD.org/pub/FreeBSD/) (<http://ftp2.cz.FreeBSD.org/pub/FreeBSD/>))

Dánia

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@dk.FreeBSD.org>](mailto:hostmaster@dk.FreeBSD.org).

- <ftp://ftp.dk.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / [http](http://ftp.dk.FreeBSD.org/pub/FreeBSD/) (<http://ftp.dk.FreeBSD.org/pub/FreeBSD/>) / [httpv6](http://ftp2.dk.FreeBSD.org/pub/FreeBSD/) (<http://ftp2.dk.FreeBSD.org/pub/FreeBSD/>))

Dél-Afrika

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@za.FreeBSD.org>](mailto:hostmaster@za.FreeBSD.org).

- <ftp://ftp.za.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.za.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.za.FreeBSD.org/pub/FreeBSD/> (ftp)

Egyesült Királyság

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@uk.FreeBSD.org>](mailto:hostmaster@uk.FreeBSD.org).

- <ftp://ftp.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.uk.FreeBSD.org/pub/FreeBSD/> (ftp / [http](http://ftp2.uk.FreeBSD.org/) (<http://ftp2.uk.FreeBSD.org/>) / rsync (<http://ftp2.uk.FreeBSD.org/ftp.freebsd.org/pub/FreeBSD/>))
- <ftp://ftp3.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.uk.FreeBSD.org/pub/FreeBSD/> (ftp)

Észtország

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@ee.FreeBSD.org>](mailto:hostmaster@ee.FreeBSD.org).

- <ftp://ftp.ee.FreeBSD.org/pub/FreeBSD/> (ftp)

Finnország

Bármilyen gond esetén a következő címet kell értesíteni: [<hostmaster@fi.FreeBSD.org>](mailto:hostmaster@fi.FreeBSD.org).

- <ftp://ftp.fi.FreeBSD.org/pub/FreeBSD/> (ftp)

Franciaország

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@fr.FreeBSD.org>.

- <ftp://ftp.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp1.fr.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp1.fr.FreeBSD.org/pub/FreeBSD/>) / rsync)
- <ftp://ftp3.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.fr.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp4.fr.FreeBSD.org/pub/FreeBSD/>) / http (<http://ftp4.fr.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp4.fr.FreeBSD.org/pub/FreeBSD/>) / rsync (rsync://ftp4.fr.FreeBSD.org/FreeBSD/) / rsyncv6 (rsync://ftp4.fr.FreeBSD.org/FreeBSD/))
- <ftp://ftp5.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.fr.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)
- <ftp://ftp7.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.fr.FreeBSD.org/pub/FreeBSD/> (ftp)

Görögország

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@gr.FreeBSD.org>.

- <ftp://ftp.gr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.gr.FreeBSD.org/pub/FreeBSD/> (ftp)

Hollandia

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@nl.FreeBSD.org>.

- <ftp://ftp.nl.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.nl.FreeBSD.org/os/FreeBSD/>) / rsync)
- <ftp://ftp2.nl.FreeBSD.org/pub/FreeBSD/> (ftp)

Hongkong

- <ftp://ftp.hk.FreeBSD.org/pub/FreeBSD/> (ftp)

Írország

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@ie.FreeBSD.org>.

- <ftp://ftp3.ie.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp3.ie.FreeBSD.org/pub/FreeBSD/>) / rsync)

Izland

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@is.FreeBSD.org>.

- <ftp://ftp.is.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)

Japán

Bármilyen gond esetén a következő címet kell értesíteni: `<hostmaster@jp.FreeBSD.org>`.

- `ftp://ftp.jp.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp2.jp.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp3.jp.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp4.jp.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp5.jp.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp6.jp.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp7.jp.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp8.jp.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp9.jp.FreeBSD.org/pub/FreeBSD/` (ftp)

Kanada

Bármilyen gond esetén a következő címet kell értesíteni: `<hostmaster@ca.FreeBSD.org>`.

- `ftp://ftp.ca.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp2.ca.FreeBSD.org/pub/FreeBSD/` (ftp)

Kína

Bármilyen gond esetén a következő címet kell értesíteni: `<hostmaster@cn.FreeBSD.org>`.

- `ftp://ftp.cn.FreeBSD.org/pub/FreeBSD/` (ftp)

Korea

Bármilyen gond esetén a következő címet kell értesíteni: `<hostmaster@kr.FreeBSD.org>`.

- `ftp://ftp.kr.FreeBSD.org/pub/FreeBSD/` (ftp / rsync)
- `ftp://ftp2.kr.FreeBSD.org/pub/FreeBSD/` (ftp / http (`http://ftp2.kr.FreeBSD.org/pub/FreeBSD/`))

Lengyelország

Bármilyen gond esetén a következő címet kell értesíteni: `<hostmaster@pl.FreeBSD.org>`.

- `ftp://ftp.pl.FreeBSD.org/pub/FreeBSD/` (ftp)
- `ftp://ftp2.pl.FreeBSD.org/pub/FreeBSD/` (ftp / ftpv6 (`ftp://ftp2.pl.FreeBSD.org/pub/FreeBSD/`) / http (`http://ftp2.pl.FreeBSD.org/pub/FreeBSD/`) / httpv6 (`http://ftp2.pl.FreeBSD.org/pub/FreeBSD/`) / rsync / rsyncv6)

Lettország

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@lv.FreeBSD.org>.

- <ftp://ftp.lv.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.lv.FreeBSD.org/pub/FreeBSD/>))

Litvánia

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@lt.FreeBSD.org>.

- <ftp://ftp.lt.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.lt.FreeBSD.org/pub/FreeBSD/>))

Németország

Bármilyen gond esetén a következő címet kell értesíteni: <de-bsd-hubs@de.FreeBSD.org>.

- <ftp://ftp.de.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp1.de.FreeBSD.org/freebsd/> (ftp / http (<http://www1.de.FreeBSD.org/freebsd/>) / rsync (<rsync://rsync3.de.FreeBSD.org/freebsd/>))
- <ftp://ftp2.de.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp2.de.FreeBSD.org/pub/FreeBSD/>) / rsync)
- <ftp://ftp4.de.FreeBSD.org/FreeBSD/> (ftp / http (<http://ftp4.de.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp5.de.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.de.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp7.de.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp8.de.FreeBSD.org/pub/FreeBSD/> (ftp)

Norvégia

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@no.FreeBSD.org>.

- <ftp://ftp.no.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)

Örményország

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@am.FreeBSD.org>.

- <ftp://ftp1.am.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp1.am.FreeBSD.org/pub/FreeBSD/>) / rsync)

Oroszország

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@ru.FreeBSD.org>.

- <ftp://ftp.ru.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.ru.FreeBSD.org/FreeBSD/>) / rsync)
- <ftp://ftp2.ru.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp2.ru.FreeBSD.org/pub/FreeBSD/>) / rsync)
- <ftp://ftp4.ru.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.ru.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp5.ru.FreeBSD.org/pub/FreeBSD/>) / rsync)
- <ftp://ftp6.ru.FreeBSD.org/pub/FreeBSD/> (ftp)

Spanyolország

Bármilyen gond esetén a következő címet kell értesíteni: `<hostmaster@es.FreeBSD.org>`.

- `ftp://ftp.es.FreeBSD.org/pub/FreeBSD/` (`ftp` / `http` (`http://ftp.es.FreeBSD.org/pub/FreeBSD/`))
- `ftp://ftp3.es.FreeBSD.org/pub/FreeBSD/` (`ftp`)

Svájc

Bármilyen gond esetén a következő címet kell értesíteni: `<hostmaster@ch.FreeBSD.org>`.

- `ftp://ftp.ch.FreeBSD.org/pub/FreeBSD/` (`ftp` / `http` (`http://ftp.ch.FreeBSD.org/pub/FreeBSD/`))

Svédország

Bármilyen gond esetén a következő címet kell értesíteni: `<hostmaster@se.FreeBSD.org>`.

- `ftp://ftp.se.FreeBSD.org/pub/FreeBSD/` (`ftp`)
- `ftp://ftp2.se.FreeBSD.org/pub/FreeBSD/` (`ftp` / `rsync` (`rsync://ftp2.se.FreeBSD.org/`))
- `ftp://ftp3.se.FreeBSD.org/pub/FreeBSD/` (`ftp`)
- `ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/` (`ftp` / `ftp6` (`ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/`) / `http` (`http://ftp4.se.FreeBSD.org/pub/FreeBSD/`) / `http6` (`http://ftp4.se.FreeBSD.org/pub/FreeBSD/`) / `rsync` (`rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/`) / `rsync6` (`rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/`))
- `ftp://ftp5.se.FreeBSD.org/pub/FreeBSD/` (`ftp` / `http` (`http://ftp5.se.FreeBSD.org/`) / `rsync` (`rsync://ftp5.se.FreeBSD.org/freebsd/`))
- `ftp://ftp6.se.FreeBSD.org/pub/FreeBSD` (`ftp` / `http` (`http://ftp6.se.FreeBSD.org/pub/FreeBSD/`))

Szaúd-Arábia

Bármilyen gond esetén a következő címet kell értesíteni: `<ftpadmin@isu.net.sa>`.

- `ftp://ftp.isu.net.sa/pub/ftp.freebsd.org/` (`ftp`)

Szlovák Köztársaság

Bármilyen gond esetén a következő címet kell értesíteni: `<hostmaster@sk.FreeBSD.org>`.

- `ftp://ftp.sk.FreeBSD.org/pub/FreeBSD/` (`ftp` / `ftp6` (`ftp://ftp.sk.FreeBSD.org/pub/FreeBSD/`) / `http` (`http://ftp.sk.FreeBSD.org/pub/FreeBSD/`) / `http6` (`http://ftp.sk.FreeBSD.org/pub/FreeBSD/`))
- `ftp://ftp2.sk.FreeBSD.org/pub/FreeBSD/` (`ftp` / `ftp6` (`ftp://ftp2.sk.FreeBSD.org/pub/FreeBSD/`) / `http` (`http://ftp2.sk.FreeBSD.org/pub/FreeBSD/`) / `http6` (`http://ftp2.sk.FreeBSD.org/pub/FreeBSD/`))

Szlovénia

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@si.FreeBSD.org>.

- <ftp://ftp.si.FreeBSD.org/pub/FreeBSD/> (ftp)

Tajvan

Bármilyen gond esetén a következő címet kell értesíteni: <hostmaster@tw.FreeBSD.org>.

- <ftp://ftp.tw.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp.tw.FreeBSD.org/pub/FreeBSD/>) / rsync / rsyncv6)
- <ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/>) / http (<http://ftp2.tw.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp2.tw.FreeBSD.org/pub/FreeBSD/>) / rsync / rsyncv6)
- <ftp://ftp3.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.tw.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp6.tw.FreeBSD.org/>) / rsync)
- <ftp://ftp7.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp9.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp10.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp11.tw.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp11.tw.FreeBSD.org/FreeBSD/>))
- <ftp://ftp12.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp13.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp14.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp15.tw.FreeBSD.org/pub/FreeBSD/> (ftp)

Új-Zéland

- <ftp://ftp.nz.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.nz.FreeBSD.org/pub/FreeBSD/>))

Ukrajna

- <ftp://ftp.ua.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.ua.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp7.ua.FreeBSD.org/pub/FreeBSD/> (ftp)

A.3. BitTorrent

Az egyes kiadásokhoz tartozó alap CD-készletek BitTorrent segítségével is elérhetőek. A lemezek képeire hivatkozó torrent állományokat a <http://torrents.freebsd.org:8080> címről tölthetjük le.

A BitTorrent kliens telepíthető a `net-p2p/py-bittorrent` portból vagy csomagból.

Miután sikeresen letöltöttük BitTorrenten keresztül a lemezképeket, a 18.6.3 Szakasz nyújthat segítséget abban, hogy kell ezeket lemezre írni.

A.4. Anonim CVS

A.4.1. Bevezetés

Az anonim CVS (vagy más néven *anoncvs*) a FreeBSD-hez mellékelt CVS-es segédprogramok által nyújtott olyan lehetőség, amivel távoli CVS repositorykkal tudunk szinkronizálni. Több más dolog mellett lehetővé teszi a FreeBSD felhasználói számára, hogy kiemelt jogosultságok nélkül képesek legyenek olvasással kapcsolatos CVS műveleteket végrehajtani a FreeBSD Projekt hivatalos anoncvs szerverein. A használatához egyszerűen csak a kiválasztott anoncvs szervert kell beállítani a `CVSROOT` környezeti változó értékének, ahol aztán a `cvs login` parancsnak a szerver által ismert “anoncvs” jelszót kell megadni. Ezután a `cvs(1)` paranccsal a többi CVS szerverhez hasonlóan lehetőségünk nyílik hozzáférni.

Megjegyzés: A `cvs login` parancs a bejelentkezésekhez szükséges jelszavakat a `HOME` könyvtárunkban levő `.cvspass` állományban tárolja. Ha ez az állomány nem létezik, akkor a `cvs login` első használatakor hibát kapunk. Ilyenkor csak hozzunk létre egy üres `.cvspass` állományt, majd próbálkozzunk újra.

Habár azt mondhatnánk, hogy a **CVSup** és az *anoncvs* lényegében egyazon feladatot oldják meg, mind a két esetben léteznek olyan kompromisszumok, amelyek befolyásolhatják a felhasználó választását a két szinkronizációs módszer között. Dióhéjban ezt úgy tudnánk összefoglalni, hogy a **CVSup** a hálózati erőforrásokat hatékonyabban kihasználja és kettejük közül ez a fejlettebb, azonban ennek meg kell fizetnünk az árát. A **CVSup** használatához először ugyanis telepítenünk kell és be kell állítanunk egy speciális klienst, illetve az adatokat a **CVSup** által *gyűjteményeknek* (collection) nevezett, viszonylag nagy méretű egységekben érhetjük el.

Ezzel szemben az **anoncvs** használata során a megfelelő CVS modul nevének felhasználásával tetszőlegesen megvizsgálhatunk önálló állományokat vagy akár programokat (mint az `ls` vagy a `grep`). Természetesen az **anoncvs** segítségével csupán az olvasást igénylő CVS műveleteket végezhetjük el, ezért ha a FreeBSD Projekt keretein belül fejleszteni is szeretnénk, akkor inkább érdemes a **CVSup** alkalmazást választani.

A.4.2. Az anonim CVS használata

A `cvs(1)` parancsot nagyon könnyű beállítani az anonim CVS repositoryk használatához, hiszen mindössze annyit kell tennünk, hogy a `CVSROOT` környezeti változó értékének megadjuk a FreeBSD Projekt valamelyik *anoncvs* szerverét. Ezen sorok írásának pillanatában a következő szerverek érhetőek el:

- *Franciaország*: `:pserver:anoncvs@anoncvs.fr.FreeBSD.org:/home/ncvs` (pserver módban a jelszó “anoncvs”, az SSH pedig nincs jelszó)

- *Tajvan:* :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs (pserver módban a cvs login használatával tetszőleges jelszó megadható, az SSH esetén pedig nincs jelszó)

```
SSH2 HostKey: 1024 02:ed:1b:17:d6:97:2b:58:5e:5c:e2:da:3b:89:88:26 /etc/ssh/ssh_host_rsa_key.pub
SSH2 HostKey: 1024 e8:3b:29:7b:ca:9f:ac:e9:45:cb:c8:17:ae:9b:eb:55 /etc/ssh/ssh_host_dsa_key.pub
```

- *Egyesült Államok:* anoncvs@anoncvs1.FreeBSD.org:/home/ncvs (csak SSH v2 — nincs jelszó)

```
SSH2 HostKey: 2048 53:1f:15:a3:72:5c:43:f6:44:0e:6a:e9:bb:f8:01:62 /etc/ssh/ssh_host_dsa_key.pub
```

Mivel a CVS használatával “kikérhetjük” (check out) tulajdonképpen a FreeBSD forrásainak akármelyik eddigi (vagy majd ezután keletkező) változatát, érdemes megismerkednünk a cvs(1) által alkalmazott revízió (revision) (az -r opcióval állítható) fogalmával és a FreeBSD Projekt repositoryjain belül engedélyezett értékeivel.

Címkéket (tag) két esetben használhatunk: a revíziók és az ágak esetén. A revíziós címkék mindig egy adott revízióra hivatkoznak, ami állandóan ugyanazt jelenti. Ezzel szemben az ágak címkéi a fejlesztés adott irányú menetének az adott pillanatban legfrissebb revízióját hivatkozzák. Mivel az ágak címkéi nem egy adott revízióra vonatkoznak, ezért elmondhatjuk róluk, hogy naponta változik a jelentésük.

Az A.7 Szakasz tartalmazza a felhasználók számára fontos revíziós címkéket. Ezek azonban nem igazak a Portgyűjteményre, mivel a Portgyűjteménynek nincs egyszerre több fejlesztési iránya.

Egy ág címkéjének megadásával általában az adott irányhoz tartozó állományok legfrissebb változatát kapjuk meg. Ha viszont az állományok egy korábbi változatára lenne szükségünk, akkor a -D dátum opció megadásával meg tudjuk adni annak időpontját. Erről részletesebben a cvs(1) man oldalán olvashatunk.

A.4.3. Példák

Habár a továbbhaladáshoz mindenképpen javasoljuk a cvs(1) man oldalának részletes áttanulmányozását, mutatunk néhány gyors példát az anonim CVS használatának tömör illusztrálására:

Példa A-1. Valami (az ls(1)) kikérése a -CURRENT ágból

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
"Jelszóként" ezután bármit megadhatunk.
% cvs co ls
```

Példa A-2. Az src/ fa kikérése SSH-n keresztül

```
% cvs -d anoncvs@anoncvs1.FreeBSD.org:/home/ncvs co src
The authenticity of host 'anoncvs1.freebsd.org (216.87.78.137)' can't be established.
DSA key fingerprint is 53:1f:15:a3:72:5c:43:f6:44:0e:6a:e9:bb:f8:01:62.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'anoncvs1.freebsd.org' (DSA) to the list of known hosts.
```

Példa A-3. Az ls(1) 6-STABLE ágban szereplő változatának kikérése

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
Amikor kéri, "jelszóként" bármit megadhatunk.
% cvs co -rRELEASE_6 ls
```

Példa A-4. Az ls(1) változásainak (Unified Diff formátumú) listázása

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
Itt "jelszóként" bármit megadhatunk.
% cvs rdiff -u -rRELENG_5_3_0_RELEASE -rRELENG_5_4_0_RELEASE ls
```

Példa A-5. A használható modulok nevének kiderítése

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
Ezután "jelszóként" bármit megadhatunk.
% cvs co modules
% more modules/modules
```

A.4.4. Egyéb helyek

A következő helyeken találhatunk még hasznos információkat a CVS használatáról:

- A CVS bemutatása (<http://users.csc.calpoly.edu/~gfisher/classes/308/handouts/cvs-basics.html>) (forrás: Kalifornia Állami Műszaki Egyetem).
- A CVS honlapja (<http://ximbiot.com/cvs/wiki/>), a CVS fejlesztésével és alkalmazásával foglalkozó közösség oldala.
- A CVSweb (<http://www.FreeBSD.org/cgi/cvsweb.cgi>) a FreeBSD Projekt által használt CVS rendszerének webes felülete.

A.5. A CTM használata

A CTM használatára a távoli könyvtárakat tudunk egy központi változattal szinkronban tartani. Eredetileg a FreeBSD forrásaihoz fejlesztették ki, de idővel mások más célokra is alkalmasnak találhatják majd. Az eltérések (delták) feldolgozásával kapcsolatban kevéske dokumentáció áll rendelkezésre, ezért a `ctm-users` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-users>) levelezési listát érdemes felkeresni, ha többet szeretnénk megtudni a CTM egyéb célú alkalmazásairól.

A.5.1. Miért használnánk a CTM-et?

A CTM segítségével a FreeBSD forrásainak helyi másolatát hozhatjuk létre. A források több különböző "kivitelben" is hozzáférhetőek. A CTM minden esetben képes eleget tenni az igényeinknek, akár az egész CVS fát, akár annak egy részét kívánjuk csak figyelemmel követni. Ha netalán FreeBSD fejlesztők lennénk, és híján vagyunk vagy éppen gyenge TCP/IP kapcsolattal rendelkezünk, esetleg egyszerűen csak automatikusan értesülni szeretnénk a

változásokról, a **CTM**-et nekünk találták ki. A leggyorsabban fejlődő ágakból is naponta legfeljebb három deltát fogunk kapni, azonban érdemes megfontolni a változások automatikus elküldését levélben. A szükséges frissítések méretét mindig igyekszünk minimalizálni. Ez egyébként általában alig 5 KB, de néha (tízből egyszer) előfordul, hogy 10 és 50 KB között van, és időnként 100 KB vagy afeletti mennyiségű frissítés is érkezik.

Amikor a fejlesztők által használt forrásokat töltjük le, magunknak kell gondoskodnunk a menet közben felmerülő különböző problémák megoldásáról. Ez kiváltképp igaz abban az esetben, amikor az aktuális, vagy hivatalos nevén “CURRENT” ágot követjük. Mielőtt azonban egy ilyenbe belevágnánk, érdemes fellapozni a FreeBSD legfrissebb változatának használatáról szóló fejezetet.

A.5.2. Mire van szükségünk a CTM használatához?

A működéshez két komponens szükséges: a **CTM** kliensprogramja és hozzá a kezdeti delták (amivel majd letöltjük a “CURRENT” forrásait).

A **CTM** program már a 2.0 kiadástól kezdve a FreeBSD része, és a források között a `/usr/src/usr.sbin/ctm` könyvtárban találjuk meg (amennyiben felraktuk).

A **CTM** működéséhez kellő “deltákat” két módon, FTP-n vagy e-mailen keresztül szerezhetjük be. Ha el tudunk érni interneten levő FTP oldalakat, akkor az alábbi FTP helyeken találunk a **CTM**-hez használható adatokat:

<ftp://ftp.FreeBSD.org/pub/FreeBSD/CTM/>

valamint lásd a tükrözéseket.

FTP-n keresztül lépünk be a könyvtárba, töltjük le a `README` nevű állományt és kövessük a benne szereplő utasításokat.

Ha viszont e-mailen keresztül akarjuk megszerezni a deltákat:

Iratkozzunk fel a **CTM** terjesztési listáinak egyikére. A `ctm-cvs-cur` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-cvs-cur>) lista az egész CVS-fát, míg a `ctm-src-cur` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-src-cur>) a fő fejlesztési ágot teszi elérhetővé. A `ctm-src-4` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-src-4>) a 4.X kiadásaihoz ágakat tartalmazza, és így tovább. (Ha nem tudjuk, hogyan kell feliratkozni egy levelezési listára, akkor kattintsunk a lista nevére vagy kövessük a <http://lists.FreeBSD.org/mailman/listinfo> linket, majd kattintsunk arra a listára, ahova fel akarunk iratkozni. Ezen az oldalon az összes, a felíratkozáshoz nélkülözhetetlen információnak szerepelnie kell.)

Miután elkezdenek megérkezni a **CTM**-frissítéseket tartalmazó levelek, a tartalmukat a `ctm_rmail` programmal tudjuk kicsomagolni és felhasználni. Az `/etc/aliases` állományba akár közvetlenül is beírhatjuk a `ctm_rmail` programot, és ezzel az önállósítani tudjuk a levélben érkező frissítések feldolgozását. A `ctm_rmail` man oldalán olvashatjuk ennek részleteit.

Megjegyzés: Nem számít, milyen módon jutunk hozzá a **CTM** által használt deltákhoz, minden esetben fel kell iratkoznunk a `ctm-announce` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-announce>) levelezési listára. Az elkövetkezendőkben ez lesz az egyetlen hely, ahová a **CTM** rendszer működtetésével kapcsolatos bejelentések beküldésre kerülnek. A felíratkozáshoz kattintsunk a fenti lista nevére és kövessük a mellette szereplő utasításokat.

A.5.3. A CTM első használata

Mielőtt nekilátnánk a CTM-hez tartozó delták használatának, először el kell jutnunk egy kiindulási ponthoz, ahonnan majd létre tudjuk hozni a rákövetkező deltákat.

Ehhez elsőként vegyük számba, pontosan mink is van. Általában mindenki egy “üres” könyvtárral kezd. Ilyenkor egy kezdeti “Empty” (mint “üres”) elnevezésű deltával tudjuk megkezdeni az CTM által ismert fa szinkronizálását. Erre a célra lesznek majd szintén alkalmasak a “megkezdett” delták is, amelyek valamikor a CD-re fognak felkerülni.

Mivel a fák maguk több tíz megabyte-nyi méretűek, ezért érdemes inkább valami kéznél levő eszközzel megkezdeni a folyamatot. Ha van -RELEASE verziójú CD-nk, akkor másoljuk le róla és bontsuk ki a kiindulásként használt forrásokat. Ezzel jelentős mennyiségű adat átvitelét takaríthatjuk meg.

A “kezdő” deltákat könnyen megismerjük a szám után x karakterrel leválasztott nevükről (például `src-cur.3210XEmpty.gz`). Az x után szereplő megnevezés a kezdeti “kiindulás” (seed) fokának felel meg. Az Empty egy üres könyvtárra utal. A szabályok szerint az Empty állapotból 100 deltánként jön létre újabb (kiindulásra alkalmas) alapváltozat. Ezek azonban nagyon nagyok is lehetnek. A 70 vagy 80 megabyte-os gzippel csomagolt adatok gyakoriak az XEmpty delták esetén.

Miután kiválasztottuk a számunkra megfelelő alapváltozatot, szükségünk lesz a tőle nagyobb sorszámú összes deltára is.

A.5.4. A CTM használata a hétköznapiakban

A delták felhasználásához egyszerűen csak ennyit kell tennünk:

```
# cd /ahol/tárolni/akarjuk/az/adatokat
# ctm -v -v /ahol/tároljuk/a/deltákat/src-xxx.*
```

A CTM képes értelmezni a gzip által csomagolt adatokat, ezért nincs szükség a delták előzetes kitömörítésére, amivel tárhelyet tudunk spórolni.

Hacsak nem tekinti tökéletesen biztonságosnak az egész folyamatot, akkor a CTM nem fog módosítani a fán. A deltákat a CTM -c kapcsolójával is ellenőrizhetjük, aminek során egyáltalán nem fog módosulni a forrásfa. Ekkor egyszerűen csak ellenőrzi a delták sértetlenségét és megnézi, hogy minden rendben zajlana-e az alkalmazásuk során.

A CTM-nek vannak még további kapcsolói is, melyekről bővebben a man oldalakból és a forráskódokból tájékozódhatunk.

Most már minden megvan, ami kellhet. Amikor kapunk egy újabb deltát, a forrásaink frissítéséhez csak futtassuk át a CTM-en.

Ne töröljük le azokat a deltákat, melyeket nehezen tudtunk letölteni. Helyette érdemes inkább megtartani ezeket arra az esetre, ha valami rossz történne. Még ha csak floppylemezek is állnak rendelkezésünkre, mindenképpen másoljuk le ezeket az fdwrite paranccsal.

A.5.5. A saját változtatásaink megtartása

Fejlesztőként biztosan szeretnénk kísérletezni és állományokat megváltoztatni a forrásfában. A CTM a helyben elkövetett változtatásokat csak korlátozottan támogatja: az ize nevű állomány meglétének vizsgálata előtt az ize.ctm állományt fogja keresni. Ha létezik, akkor a CTM az ize helyett ezen fog dolgozni.

Ezzel a viselkedéssel nyerjük a saját változtatásaink megtartásának egyszerű módját: csak másoljuk le `.ctm` kiterjesztéssel a módosítani tervezett állományokat. Ezután már szabadon módosíthatjuk a forrásokat, miközben a **CTM** a `.ctm` kiterjesztésű állományokat folyamatosan szinkronban tartja.

A.5.6. A CTM egyéb érdekes beállításai

A.5.6.1. Derítsük ki pontosan miket is fog érinteni a frissítés

A **CTM** által a forrásokon elvégzendő változtatások listáját az `-l` kapcsolóval kérdezhetjük le.

Ez akkor esik kézre, ha szeretnénk feljegyezni a bekövetkező változásokat, vagy bármilyen módon elő- vagy utófeldolgozni a módosított állományokat, esetleg szimplán elővigyázatosak akarunk lenni.

A.5.6.2. Biztonsági másolat készítése a frissítés előtt

Néha egyszerűen csak szeretnénk az összes érintett állományról biztonsági másolatot készíteni a **CTM** által elvégzett frissítés előtt.

A `-B mentés-állomány` beállítás megadásával az adott **CTM** delta által módosítandó összes állomány tárolásra kerül a `mentés-állomány` nevű állományba.

A.5.6.3. A frissíthető állományok korlátozása

Egyes esetekben érdeünkben állhat leszűkíteni a **CTM** által eszközölt frissítések hatáskörét, vagy egyszerűen csak néhány állomány szinkronizálására van szükségünk.

A **CTM** számára feldolgozható állományok listáját reguláris kifejezés formájában az `-e` és `-x` opciók mentén határozhatjuk meg.

Például ha a `lib/libc/Makefile` állomány az összegyűjtött **CTM** delták szerinti legfrissebb verziójához kívánunk hozzájutni, akkor futtassuk az alábbi parancsot:

```
# cd /akarhova/ahova/ki/akarjuk/bontani/
# ctm -e '^lib/libc/Makefile' ~ctm/src-xxx.*
```

A **CTM** deltákban megadott minden egyes állomány esetén az `-e` az `-x` opciók a parancssorban történt megadásuk sorrendjében kerülnek feldolgozásra. Egy állományt kizárólag csak akkor dolgoz fel a **CTM**, ha az az `-e` és `-x` opciók kiértékelése után is indokolt.

A.5.7. További tervek a CTM-mel kapcsolatban

Rengeteg van:

- Valamiféle hitelesítés bevezetése a **CTM** rendszerbe, amivel észlelhetőek a meghamisított **CTM**-frissítések.
- A **CTM** beállításainak letisztázása, mivel eléggé megtévesztőek és nehézkesen használhatóak.

A.5.8. Egyebek

Léteznek delták a portok gyűjteményéhez is, azonban még nem mutatkozott túlzottan nagy érdeklődés irántuk.

A.5.9. CTM tükrözések

A CTM/FreeBSD anonim FTP-n keresztül elérhető az alábbi tüköroldalak valamelyikéről. Amennyiben ezen a módon kívánjuk letölteni a CTM rendszerhez tartozó állományokat, először próbálkozzunk a hozzánk legközelebb levő szerverrel.

Ha bármilyen gond merülne fel, értesítsük a ctm-users (<http://lists.FreeBSD.org/mailman/listinfo/ctm-users>) levelezési listát.

Kalifornia, Bay Area (hivatalos forrás)

- <ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CTM/>

Dél-Afrika (a korábbi delták biztonsági másolatai)

- <ftp://ftp.za.FreeBSD.org/pub/FreeBSD/CTM/>

Tajvan/R.O.C.

- <ftp://ctm.tw.FreeBSD.org/pub/FreeBSD/development/CTM/>
- <ftp://ctm2.tw.FreeBSD.org/pub/FreeBSD/development/CTM/>
- <ftp://ctm3.tw.FreeBSD.org/pub/FreeBSD/development/CTM/>

Ha nem találtunk volna hozzánk közel eső tükrözést, vagy ha talált tükör nem elég friss, akkor próbálkozzunk egy olyan keresőmotor használatával, mint például az alltheweb (<http://www.alltheweb.com/>).

A.6. A CVSup használata

A.6.1. Bevezetés

A CVSup távoli szervereken található központi repositorykban levő források terjesztésére és a rajtuk keresztüli frissítésre alkalmas programcsomag. A FreeBSD forrásait egy CVS repositoryban tartják karban Kaliforniában egy fejlesztéseket tároló központi számítógépen. A CVSup segítségével a FreeBSD felhasználói könnyen szinkronban tudják vele tartani a saját forrásaikat.

A **CVSup** az ún. *lehúzással* frissít. Ilyenkor a kliensek csak akkor kérnek a szervertől frissítéseket, amikor szükségük van rá, miközben a szerver passzívan várja a frissítési kérelmeket. Ennek megfelelően tehát minden esetben a kliens kezdeményezi a frissítést, a szerver pedig önmagától sosem küld ilyeneket kérés nélkül. A felhasználóknak így vagy maguknak kell meghívniuk a **CVSup** kliensét, vagy a frissítések rendszeres automatikus letöltéséhez be kell állítaniuk a `cron` rendszerprogramot.

A **CVSup** kifejezés ebben az írásmódban az egész programcsomagra utal. Fő alkotórészei a a felhasználó gépén futó `cvsup` nevű kliens, és a FreeBSD tüköroldalain futó `cvsupd` nevű szerver.

A FreeBSD dokumentációjának és levelezési listáinak fürkészése során rengeteg hivatkozást találhatunk egy **sup** nevű alkalmazásra. A **sup** a **CVSup** elődje volt, és hasonló célokat szolgált. A **CVSup** használat tekintetében nagyon hasonlít a **sup**-hoz, és ami azt illeti, a a `sup` konfigurációs állományával visszafele kompatibilis formátumot használ. Mivel a **CVSup** sokkal gyorsabb és rugalmasabb, a **sup**ot már nem használja a FreeBSD Projekt.

Megjegyzés: A **csup** a **CVSup** C nyelven újraírt változata. Legnagyobb előnye, hogy gyorsabb és nincs szüksége a Modula-3 nyelv futtató környezetére, ezért azt nem kell a használatához telepíteni. Ráadásul, ha a FreeBSD 6.2 vagy annál későbbi változatát használjuk, akkor minden további nélkül a rendelkezésünkre áll, hiszen az alaprendszer része. A FreeBSD korábbi verzióinak alaprendszerei ugyan nem tartalmazzák a `csup(1)` parancsot, viszont a `net/csup` port vagy csomag segítségével pillanatok alatt telepíteni tudjuk. Amennyiben a **csup** mellett tennénk le a voksunkat, a szakasz fennmaradó részében egyszerűen hagyjuk ki a **CVSup** telepítéséről szóló lépéseket és a **CVSup** hivatkozásait helyettesítsük a **csup** programmal.

A.6.2. Telepítés

A **CVSup** telepítésének legegyszerűbb módja a FreeBSD csomaggyűjteményében található előrefordított `net/cvsup` csomag használata. Ha viszont inkább forrásból akarjuk telepíteni a **CVSup**ot, akkor helyette használjuk a `net/cvsup` portot. De legyünk elővigyázatosak: a `net/cvsup` portnak szüksége van a Modula-3 rendszerre, aminek letöltése és lefordítása pedig meglehetősen sok időt és tárhelyet igényel.

Megjegyzés: Ha olyan gépen akarjuk használni a **CVSup**ot, ahol nincs **XFree86**, **Xorg** vagy bármilyen más ilyen szerver, akkor használjuk a `net/cvsup-without-gui` portot, ami nem tartalmazza a hozzá tartozó grafikus felületet.

Ha a FreeBSD 6.1 vagy korábbi változatain szeretnénk telepíteni a **csup**ot, használjuk a FreeBSD csomaggyűjteményében megtalálható `net/csup` csomagot. Ha viszont forrásból kívánjuk telepíteni a **csup** programot, akkor helyette használjuk a `net/csup` portot.

A.6.3. A CVSup beállítása

A **CVSup** működését a `supfile` elnevezésű állomány vezérli. A `/usr/share/examples/cvsup/` (`/usr/share/examples/cvsup/`) könyvtárban találhatunk néhány példát a `supfile` állományokra.

A `supfile` állományban szereplő információk a **CVSup** használatával kapcsolatban a következő kérdéseket válaszolják meg:

- Milyen állományokat akarunk letölteni?

- Milyen verziókra van szükségünk?
- Honnan akarjuk ezeket beszerezni?
- Hova akarjuk rakni a számítógépünkön?
- Hova akarjuk rakni az állapotot tároló állományokat?

Az imént feltett kérdésekre a következő szakaszokban összeállítandó `supfile` segítségével fogunk válaszolni. Ehhez először bemutatjuk a `supfile` formátumú állományok általános szerkezetét.

A `supfile` állományok szöveget tartalmaznak. A megjegyzések `#` karakterrel kezdődnek és a sor végéig tartanak. A kizárólag csak megjegyzéseket tartalmazó vagy üres sorok nem kerülnek feldolgozásra.

Az összes többi fennmaradó sorban pedig azokat az állományokat írjuk le, amelyeket a felhasználó le akar tölteni. Az ilyen fajtájú sorok egy "gyűjtemény" (collection) névvel kezdődnek, ami állományok egy szerver által meghatározott logikai csoportjára utal. A gyűjtemény neve ennek megfelelően elárulja a szervernek, hogy pontosan milyen állományokra van szükségünk. Ezután következik whitespace-szel elválasztva nulla vagy több mező, amelyek a korábban feltett kérdéseinket válaszolják meg rendre. Ezeknek a mezőknek két típusa létezik: a beállításokat és a konkrét értéket tároló mezők. A beállításokat tároló mezők különböző kulcsszavakat tartalmaznak, például a `delete` (törlés) vagy `compress` (tömörítés). Az értéket tároló mezők is egy kulcsszóval kezdődnek, azonban utána közvetlenül egy `=` (egyenlőségjel) jön, amelyet egy második szó követ szorosan. Így például a `release=cvs` pontosan egy ilyen értékmező lesz.

Egy `supfile` általában egynél több gyűjtemény letöltését írja le. Ezért az ilyen állományok felépítésének egyik módja, ha az egyes gyűjteményhez explicite megadjuk a hozzá tartozó mezőket. Azonban így a `supfile` állományok gyorsan megnövekednek és kényelmetlenné válnak, mivel a legtöbb gyűjtemény esetén szinte ugyanazokat a mezőket kellene megadnunk. A **CVSup** az ilyen típusú bonyodalmak elkerülésére egy alapértelmezési megoldást javasol. A `*default` nevű álgyűjteménnyel kezdődő sorok segítségével meg tudunk adni olyan beállításokat és értékeket, amelyek az utána következő gyűjtemények számára alapértelmezésnek fognak számítani a `supfile` állományban. Az itt megadott alapértelmezések természetesen az egyes gyűjteményekben tetszőleges módon felülbírálhatóak, a mezők magán a gyűjteményen belüli megadásával. Az állományban az alapértelmezések is megváltoztathatóak vagy bővíthetők további `*default` sorok hozzáadásával.

Mindezek tudatában most már megkezdhetjük a FreeBSD-CURRENT ág tartalmának letöltésére és frissen tartására alkalmas `supfile` állomány összeállítását.

- Milyen állományokat akarunk letölteni?

A **CVSup**on keresztül elérhető állományok "gyűjteményeknek" hívott nevesített csoportokra bontva érhetőek el. A hivatkozható gyűjtemények leírását a következő szakaszban találjuk. Ebben a példában most szeretnénk letölteni az egész FreeBSD rendszer forrását. Ezt a `src-all` nevű gyűjteményre hivatkozva érhetjük el. A `supfile` állományunk létrehozásának első lépéseként soronként egyet megadva felsoroljuk a letölteni kívánt gyűjteményeket (jelen esetünkben csak egyetlen egyet):

```
src-all
```

- Milyen verziókra van szükségünk?

A **CVSup** használatával tulajdonképpen a források összes valaha létezett verziójához hozzá tudunk férni. Ez annak köszönhető, hogy a **cvsupd** szerver közvetlenül a CVS repositoryból dolgozik, ami pedig az összes verziót tartalmazza. A `tag=` és `date=` értékmezők segítségével adhatjuk meg az igényelt verziókat.

Figyelem! Legyünk óvatosak azonban a `tag=` mezők helyes megadásával. Egyes címkék ugyanis csak bizonyos állománygyűjtemények esetén élnek. Ha hibás vagy elírt címkét adunk meg, akkor a **CVSup** törölni

fog olyan állományokat, amelyeket valószínűleg nem kellene. A `ports-*` gyűjtemények esetében pedig kifejezetten csak a `tag=.` mezők használhatóak!

A `tag=` mezők a tárházban található szimbolikus címkéket nevezik meg. A címkéknek két típusa van: a revíziókhöz és az ágakhoz tartozó címkék. A revíziós címkék mindig egy adott revíziót hivatkoznak, jelentésük állandó. Ezzel szemben az ágak címkéi egy adott fejlesztési ág adott időpontjában elérhető revíziót címkézi. Mivel az ágak címkéi nem egy konkrét revízióra vonatkoznak, ezért akár olyanra is utalhatnak, ami pillanatnyilag még nem is létezik.

Az A.7 Szakaszban megtalálhatjuk a fontosabb ágak címkéit. A **CVSup** konfigurációs állományában a címkéket a `tag=` előtaggal kell bevezetni (így tehát a `RELENG_4` címke hivatkozása `tag=RELENG_4` lesz). Ne felejtsük el, hogy a Portgyűjtemény esetében csak `tag=.` mező megadásának van értelme.

Figyelem!gyekezünk pontosan lemásolni a címkék neveit, mivel a **CVSup** nem képes megkülönböztetni az érvényes és az érvénytelen címkéket. Ha véletlen elírjuk a címkét, akkor a **CVSup** úgy fog viselkedni, mintha olyan érvényes címkére hivatkozhatunk volna, amihez nem tartoznak állományok. Ennek következtében pedig egyszerűen letörli a már meglevő forrásainkat.

Egy ág címkéjének megadása során általában az adott fejlesztési vonal legfrissebb verzióját kapjuk meg. Ha viszont az adott ág valamelyik korábbi változatára lenne szükségünk, akkor a `date=` értékmező felhasználásával meg tudjuk adni a hozzá tartozó dátumot. Ennek működéséről a `cvsup(1)` man oldala részletesebben értekezik.

A példában mi most a FreeBSD-CURRENT verziót akarjuk letölteni. Ezért a következő sort tesszük a `supfile` állományunk elejére:

```
*default tag=.
```

Ha nem adunk meg sem `tag=`, sem pedig `date=` mezőket, akkor egy fontos eset következik be. Ilyenkor ugyanis egy konkrét verzió helyett közvetlenül a szerver CVS repository-jából kapjuk meg az állományokat, az összes kiegészítő információjukkal együtt. A fejlesztők általában ezt a típusú megoldást kedvelik, mivel így a saját rendszerükön is könnyen karban tudnak tartani egy példányt, amiben tudnak keresni a revíziók között és ki tudják kérni akár az állományok korábbi változatait is. Természetesen ennek függvényében jóval több tárhelyre van szükségük.

- Honnan akarjuk ezeket beszerezni?

A `host=` mező beállításával közöljük a `cvsup` klienssel, honnan töltsse le a frissítéseket. A **CVSup** tükrözések közül bármelyik megfelel erre a célra, habár leginkább azt érdemes választani, ami a kibertérben a hozzánk legközelebb esik. A példában most egy kitalált FreeBSD terjesztési oldalt választunk, a `cvsup99.FreeBSD.org`-ot:

```
*default host=cvsup99.FreeBSD.org
```

A **CVSup** futtatása előtt tehát ne felejtsük el megváltoztatni ezt a létező számítógép hálózati nevére. A `cvsup` futtatásakor a `-h hálózati név` opció megadásával lehetőségünk ennek felülbírálására.

- Hova akarjuk rakni a számítógépünkön?

A `prefix=` mező adja meg a `cvsup` számára, hogy hova tegye a kapott állományokat. A példában a forrásokat közvetlenül a forrásokat tároló központi könyvtárba, a `/usr/src` könyvtárba tettük. Mivel a `src` könyvtár neve már hallgatólágosan benne foglaltatik a letöltésre kiválasztott gyűjtemény nevében, ezért itt csak ennyit kell megadnunk:

```
*default prefix=/usr
```

- Hova akarjuk rakni az állapotot tároló állományokat?

A **CVSup** kliens egy “bázisnak” (base) nevezett könyvtárban folyamatosan fenntart bizonyos állományokban állapotokat (status file). Ezek a már letöltött állományok nyilvántartásával segítik a **CVSup** hatékony munkavégzését. Mi most a szabványos bázist, a /var/db könyvtárat fogjuk használni:

```
*default base=/var/db
```

Amennyiben még nem létezne a bázisként használni kívánt könyvtár, ideje létrehozunk. A `cvsup` ugyanis egy nem létező könyvtár esetén nem lesz hajlandó működni.

- További beállítások a `supfile` állományban:

Általában még egy sor szokott szerepelni a `supfile` állományokban:

```
*default release=cvs delete use-rel-suffix compress
```

A `release=cvs` mező jelzi, hogy a szervernek a FreeBSD fő CVS repositoryból kell kikeresnie az információkat. Tulajdonképpen majdnem mindig erről van szó, és az itt megadható többi lehetőség ismertetése most egyébként is meghaladná a szakasz határait.

A `delete` hatására a **CVSup** képes lesz állományokat törölni. Mindig érdemes megadnunk, hiszen a **CVSup** csak így tudja teljes mértékben frissentartani a forrásokat. A **CVSup** természetesen csak azokat az állományokat igyekszik letörölni, amelyek miatt valóban felelős. A kóbor állományokat nem fogja bántani.

A `use-rel-suffix` hatása egy igazi... Rejtély. Ha tényleg érdekel minket a működése, lapozzuk fel bátran a `cvsup(1)` man oldalát. Nyugodtan adjuk meg és különösebben ne törődjünk vele.

A `compress` beállítás segítségével a kommunikációs csatornán vándorló adatokat tudjuk gzip-szerű módon tömöríteni. Ha a hálózati kapcsolatunk sebessége meghaladja a 1,5 Mbitet másodpercenként (T1), akkor ezt már nem érdemes használni, viszont minden más esetben lényeges gyorsulást hozhat.

- Összegezzük az eddigieket:

Íme a példaként összerakott `supfile` állományunk teljes tartalma:

```
*default tag=.
*default host=cvsup99.FreeBSD.org
*default prefix=/usr
*default base=/var/db
*default release=cvs delete use-rel-suffix compress

src-all
```

A.6.3.1. A `refuse` állomány

Ahogy arról már korábban szó esett, a **CVSup** *lehúzással* frissít. Ez alapvetően annyit jelent, hogy feltárcsázunk egy **CVSup** szervert, aki a következőt mondja nekünk: “A következőket tudod tőlem letölteni...”, amire a kliensünk ezt válaszolja: “Rendben, akkor nekem kell ez, ez, ez meg ez.” Alapértelmezés szerint a **CVSup** kliense azokat az állományokat fogja letölteni, amelyeket a konfigurációs állományban szereplő gyűjtemények és címkék által megneveztünk. Ez azonban nem mindig felel meg az igényeinknek, különösen akkor, amikor a `doc`, `ports` vagy `www` fákat akarjuk letölteni — az emberek többsége ugyanis nem beszél négy vagy öt nyelven, ezért nincs is szükségük a nyelvfüggő állományok letöltésére. A Portgyűjtemény letöltése során a `ports-all` helyett egyszerűen egyenként is felsorolhatjuk a számunkra érdekes kategóriákat (például `ports-astrology`, `ports-biology` stb). Azonban mivel a `doc` és a `www` fákhoz nincsenek nyelvfüggő gyűjtemények, ezért elő kell halásznunk a **CVSup** egyik remek funkcióját, a `refuse` állományt.

A `refuse` állománnyal lényegében arra utasítjuk a **CVSup** alkalmazást, hogy a gyűjteményekből ne töltsen le az összes állományt. Úgy is fogalmazhatnánk, hogy javaslatára a kliens *visszautasít* (`refuse`) bizonyos szerverről érkező állományokat. Ezeket a visszautasításokat tároló `refuse` állományt a *bázis/sup/* könyvtárban találhatjuk meg (illetve ha még nincsenek, akkor ide kell rakunk ezeket). Itt a *bázis* a `supfile` állományban megadott `base=` mezőre utal, ami a példánkban a `/var/db` könyvtár volt. Ennek megfelelően tehát a `refuse` állomány a `/var/db/sup/refuse` lesz.

A `refuse` állomány felépítése igen egyszerű: a letölteni nem kívánt állományok és könyvtárak neveit tartalmazza. Például ha az angolul mellett esetleg még beszélünk egy kevés németet is, de nincs szükségünk az angol dokumentáció német fordítására sem, akkor a következőket írjuk a `refuse` állományba:

```
doc/bn_*
doc/da_*
doc/de_*
doc/el_*
doc/es_*
doc/fr_*
doc/hu_*
doc/it_*
doc/ja_*
doc/mn_*
doc/nl_*
doc/no_*
doc/pl_*
doc/pt_*
doc/ru_*
doc/sr_*
doc/tr_*
doc/zh_*
```

és így tovább a többi nyelvre is (melyeket a FreeBSD CVS repository (<http://www.FreeBSD.org/cgi/cvsweb.cgi/>) böngészésével deríthetjük ki).

Ezzel az alkalmas funkcióval a lassú vagy drága internetes kapcsolattal rendelkező felhasználók nagyon jól tudnak gazdálkodni, mivel így nem kell letölteniük az egyáltalán nem használt állományokat. A `refuse` állományokról és a **CVSup** más hasonlóan elegáns funkcióiról a saját man oldaláról tudhatunk meg többet.

A.6.4. A CVSup futtatása

Most már készen állunk egy próba frissítés elvégzésére. A parancssorban nem sok mindent kell beírunk ehhez:

```
# cvsup supfile
```

ahol a `supfile` a frissen létrehozott `supfile` állományunk neve lesz. Feltételezve, hogy a parancsot X11 alatt adtuk ki, az `cvsup` erre feldob egy grafikus ablakot néhány gombbal. Nyomjuk meg a `go` feliratú gombot és dőlünk hátra.

Mivel a példában a `/usr/src` könyvtárunk frissítését állítottuk be, az állományok aktualizálásához szükséges jogosultságok biztosításához a `cvsup` programot `root` felhasználóként kell elindítanunk. Teljesen érthető, ha egy kicsit izgatottak vagyunk ezekben a pillanatokban, hiszen az előbb hoztunk létre egy általunk eddig ismeretlen programhoz egy konfigurációs állományt. Ezért megemlítenénk, hogy ilyenkor először mindig próbáljuk ki a

konfigurációkat, mielőtt azok bármilyen módosítást végeznének a fontos állományainkon. Ehhez hozzunk létre valahol egy üres könyvtárat, majd adjuk meg a parancssorban ennek a nevét:

```
# mkdir /var/tmp/proba
# cvsup supfile /var/tmp/proba
```

Az így megadott könyvtárba kerülnek a frissítés eredményeképpen keletkező állományok. A **CVSup** először megvizsgálja a `/usr/src` könyvtárban található állományokat, viszont egyiküket sem módosítja vagy törli. A frissítések helyett a `/var/tmp/proba/usr/src` könyvtárba fognak kerülni. A **CVSup** emellett még a báziskönyvtárban tárolt állapotokat sem fogja megváltoztatni. A módosított állományok új változatai a megadott könyvtárba jönnek létre. Mivel a `/usr/src` könyvtárat ehhez csak olvasni fogjuk, a próba lefuttatásához még `root` felhasználónak sem kell lennünk.

Ha nem használunk X11-et vagy egyszerűen csak nincs szükségünk a grafikus felületre, a parancssorban pár további opció megadásával így is kiadhatjuk a `cvsup` parancsot:

```
# cvsup -g -L 2 supfile
```

A `-g` hatására a **CVSup** nem hozza be a grafikus felületét. Ha nem talál X11-et, akkor ez természetesen automatikus, de ellenkező esetben ezt is meg kell adnunk.

Az `-L 2` megadásával a **CVSup** az összes elvégzendő frissítésről részletes értesítést ad. A részletességnek három foka van, `-L 0`-tól indulva egészen `-L 2`-ig. Itt az alapértelmezett érték a 0, amivel a hibaiüzenetek kivételével egyetlen üzenetet sem kapunk.

Rengeteg egyéb beállítás adható még meg, ezeket a `cvsup -H` kiadásával kérdezhetjük le. A beállítások pontosabb leírását a man oldalon találjuk meg.

Miután elégedetten tapasztaltuk, hogy a frissítés remekül működik, a `cron(8)` segítségével próbáljuk meg az egész folyamatot önműködővé tenni a **CVSup** szabályos időközönkénti futtatásával. Ekkor viszont magától értetődik, hogy a **CVSup** számára ne engedjük használni a grafikus felületet.

A.6.5. A CVSup állománygyűjteményei

A **CVSup** révén elérhető állománygyűjtemények egy hierarchikus rendszert alkotnak. Van néhány nagyobb állománygyűjtemény, amelyek kisebb al-állománygyűjteményekre bonthatóak. A nagyobb gyűjtemények letöltése ezért a kisebb algyűjtemények letöltésével egyenlő. A gyűjtemények közt fennálló hierarchikus rendszer a lentebb szereplő lista behúzásaiban érhető tetten.

A leggyakrabban használt gyűjtemények a `src-all` és a `ports-all` neveket viselik. A többi gyűjteményt általában csak kevesen és csak speciális célokra használják, ezért egyes tükrözéseken nem feltétlenül találjuk meg mindegyiküket.

```
cvs-all release=cvs
```

A FreeBSD fő CVS repositoryja, beleértve a titkosításhoz tartozó kódokat is.

```
distrib release=cvs
```

A FreeBSD terjesztéséhez és tükrözéséhez kapcsolódó állományok.

`doc-all release=cvs`

A FreeBSD kézikönyvének és a többi dokumentáció forrásai. Nem tartalmazza a FreeBSD honlapjának forrásait.

`ports-all release=cvs`

A FreeBSD portgyűjteménye.

Fontos: Ha nem akarjuk a `ports-all` egészét (vagyis a teljes portfát) frissíteni, csak a lentebb szereplő egyes algyűjteményeket letölteni, akkor *soha* ne feledkezzünk meg a `ports-base` megadásáról! Amikor valami változik a portok működésében, akkor a `ports-base` által képviselt algyűjteményben szereplő állományokat igen gyorsan elkezdik használni a “valódi” portok. Ezért ha csak a “valódi” portokat frissítjük, amelyek viszont igényt tartanak néhány újabb funkcióra is, akkor könnyen fordítási hibára vagy különböző rejtélyes hibaüzenetekbe futhatunk. Emiatt *legeslegelőször* mindig tegyünk róla, hogy a `ports-base` algyűjteményünk a lehető legfrissebb legyen.

Fontos: Ha a `ports/INDEX` állomány egy saját példányát kívánjuk létrehozni, akkor ahhoz a `ports-all` gyűjteményt (tehát a teljes portfát) le *kell* kérnünk. A `ports/INDEX` állományt a portfá egy része alapján nem készíthetjük el. Erről bővebben lásd a GYIK (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/applications.html#MAKE-INDEX)-ot.

`ports-accessibility release=cvs`

A fogyatékos felhasználókat segítő szoftverek.

`ports-arabic release=cvs`

Arab nyelvi támogatás.

`ports-archivers release=cvs`

Archiváló eszközök.

`ports-astro release=cvs`

Csillagászathoz tartozó portok.

`ports-audio release=cvs`

Hangtámogatás.

`ports-base release=cvs`

A Portgyűjtemény saját infrastruktúrája — az `Mk/`, `Tools/` és `/usr/ports` különféle alkönyvtáraiban elhelyezkedő állományok.

Megjegyzés: Ne hagyjuk figyelmen kívül a fenti fontos figyelmeztetést sem: ezt az algyűjteményt *mindig* a FreeBSD Portgyűjteményével együtt frissítsük!

ports-benchmarks release=cvs

Teljesítménytesztek.

ports-biology release=cvs

Biológia.

ports-cad release=cvs

Számítógépes tervezőeszközök (CAD).

ports-chinese release=cvs

Kínai nyelvi támogatás.

ports-comms release=cvs

Kommunikációs szoftverek.

ports-converters release=cvs

Karakterkódolások közti átalakítók.

ports-databases release=cvs

Adatbázisok.

ports-deskutils release=cvs

A számítógép feltalálása előtt is már létező eszközök.

ports-devel release=cvs

Fejlesztőeszközök.

ports-dns release=cvs

Névfeloldással kapcsolatos szoftverek.

ports-editors release=cvs

Szövegszerkesztők.

ports-emulators release=cvs

Más operációs rendszerek emulátorai.

ports-finance release=cvs

Pénzügyi, gazdasági és hasonló alkalmazások.

ports-ftp release=cvs

FTP kliensek és szerverek.

`ports-games release=cvs`

Játékok.

`ports-german release=cvs`

Német nyelvi támogatás.

`ports-graphics release=cvs`

Grafikus segédeszközök.

`ports-hebrew release=cvs`

Héber nyelvi támogatás.

`ports-hungarian release=cvs`

Magyar nyelvi támogatás.

`ports-irc release=cvs`

IRC-vel kapcsolatos programok.

`ports-japanese release=cvs`

Japán nyelvi támogatás.

`ports-java release=cvs`

Java segédeszközök.

`ports-korean release=cvs`

Koreai nyelvi támogatás.

`ports-lang release=cvs`

Programozási nyelvek.

`ports-mail release=cvs`

Levelező programok.

`ports-math release=cvs`

Numerikus számításokkal foglalkozó programok.

`ports-mbone release=cvs`

MBone alkalmazások.

`ports-misc release=cvs`

Egyéb segédprogramok.

`ports-multimedia release=cvs`

Multimediás szoftverek.

`ports-net release=cvs`

Hálózati szoftverek.

`ports-net-im release=cvs`

Üzenetküldő (Instant Messaging, IM) szoftverek.

`ports-net-mgmt release=cvs`

Hálózati karbantartó szoftverek.

`ports-net-p2p release=cvs`

Egyenrangú (Peer to Peer, P2P) hálózatok.

`ports-news release=cvs`

USENET hírszoftverek.

`ports-palm release=cvs`

A Palm™ sorozat szoftveres támogatása.

`ports-polish release=cvs`

Lengyel nyelvi támogatás.

`ports-ports-mgmt release=cvs`

A portok és csomagok karbantartását végző segédeszközök.

`ports-portuguese release=cvs`

Portugál nyelvi támogatás.

`ports-print release=cvs`

Nyomdai programok.

`ports-russian release=cvs`

Orosz nyelvi támogatás.

`ports-science release=cvs`

Tudományos programok.

`ports-security release=cvs`

Biztonsági segédprogramok.

`ports-shells release=cvs`

Parancsértelmezők.

`ports-sysutils release=cvs`

Rendszerprogramok.

`ports-textproc release=cvs`

Szövegfeldolgozást segítő eszközök (kivéve az asztali kiadványszerkesztést).

`ports-ukrainian release=cvs`

Ukrán nyelvi támogatás.

`ports-vietnamese release=cvs`

Vietnámi nyelvi támogatás.

`ports-www release=cvs`

A világhálóhoz tartozó szoftverek.

`ports-x11 release=cvs`

Az X Window System működését segítő portok.

`ports-x11-clocks release=cvs`

X11 órák.

`ports-x11-drivers release=cvs`

X11 meghajtók.

`ports-x11-fm release=cvs`

X11 állománykezelők.

`ports-x11-fonts release=cvs`

X11 betűtípusok és a hozzájuk tartozó segédprogramok.

`ports-x11-toolkits release=cvs`

X11 eszközenszerek.

`ports-x11-servers release=cvs`

X11 szerverek.

`ports-x11-themes release=cvs`

X11 témák.

`ports-x11-wm release=cvs`

X11 ablakkezelők.

`projects-all release=cvs`

A FreeBSD projektek forrásainak repositoryja.

`src-all release=cvs`

A FreeBSD fontosabb forrásai, a titkosításhoz tartozó kódokkal együtt.

`src-base release=cvs`

A `/usr/src` könyvtárban levő egyéb állományok.

`src-bin release=cvs`

Az egyfelhasználós módban használható segédeszközök (`/usr/src/bin`).

`src-cddl release=cvs`

A CDDL licenc szerint terjesztett segédprogramok és függvénykönyvtárak (`/usr/src/cddl`).

`src-contrib release=cvs`

A FreeBSD Projekten kívül fejlesztett segédprogramok és függvénykönyvtárak, viszonylag kevés módosítással (`/usr/src/contrib`).

`src-crypto release=cvs`

A FreeBSD Projekten kívül fejlesztett, titkosítással kapcsolatos segédprogramok és függvénykönyvtárak, viszonylag kevés módosítással (`/usr/src/crypto`).

`src-eBones release=cvs`

Kerberos és DES (`/usr/src/eBones`). A FreeBSD jelenlegi változatai nem használják.

`src-etc release=cvs`

A rendszer beállításait tartalmazó állományok (`/usr/src/etc`).

`src-games release=cvs`

Játékok (`/usr/src/games`).

`src-gnu release=cvs`

A GPL licenc szerint terjesztett segédprogramok (`/usr/src/gnu`).

`src-include release=cvs`

(C nyelvi) Header állományok (`/usr/src/include`).

`src-kerberos5 release=cvs`

A Kerberos5 biztonsági csomag (`/usr/src/kerberos5`).

`src-kerberosIV release=cvs`

A KerberosIV biztonsági csomag (`/usr/src/kerberosIV`).

`src-lib release=cvs`

Függvénykönyvtárak (`/usr/src/lib`).

`src-libexec release=cvs`

Más programok által futtatott rendszerprogramok (`/usr/src/libexec`).

`src-release release=cvs`

A FreeBSD kiadások elkészítéséhez szükséges állományok (`/usr/src/release`).

`src-rescue release=cvs`

Statikusan linkelt programok vészhelyzet esetére, lásd `rescue(8)` (`/usr/src/rescue`).

`src-sbin release=cvs`

Egyfelhasználós módban használható rendszereszközök (`/usr/src/sbin`).

`src-secure release=cvs`

Titkosítással foglalkozó függvénykönyvtárak és parancsok (`/usr/src/secure`).

`src-share release=cvs`

Több rendszer között megosztható állományok (`/usr/src/share`).

`src-sys release=cvs`

A rendszermag (`/usr/src/sys`).

`src-sys-crypto release=cvs`

A rendszermagban levő titkosítással foglalkozó kód (`/usr/src/sys/crypto`).

`src-tools release=cvs`

A FreeBSD karbantartására való különböző segédprogramok (`/usr/src/tools`).

`src-usrbin release=cvs`

Felhasználói segédprogramok (`/usr/src/usr.bin`).

`src-usrsbin release=cvs`

Rendszerszintű segédprogramok (`/usr/src/usr.sbin`).

`www release=cvs`

A FreeBSD Projekt honlapjának forráskódja.

`distrib release=self`

A **CVSup** szerver saját konfigurációs állományai. A **CVSup** tükrözései használják.

`gnats release=current`

A GNATS hibanyilvántartó adatbázis.

mail-archive release=current

A FreeBSD levelezési listáinak archívuma.

www release=current

A FreeBSD Projekt honlapjának generált állományai (de nem a forrásai). A WWW tükrözések használják.

A.6.6. Bővebb információk

A **CVSup** részletesebb bemutatását és a hozzá tartozó GYIK-ot A CVSup honlapján (<http://www.cvsup.org>) találjuk meg.

A **CVSup** FreeBSD-re vonatkozó tárgyalása a FreeBSD technical discussions levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>)n történik. Itt és az FreeBSD announcements levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce>)n jelentik be a szoftver újabb változatait.

A **CVSup** alkalmazással kapcsolatos kérdéseket és hibajelentéseket illetően a CVSup GYIK (<http://www.cvsup.org/faq.html#bugreports>)-ot érdemes megnéznünk.

A.6.7. CVSup oldalak

A FreeBSD CVSup szerverei az alábbi oldalakon érhetőek el:

Központi szerverek, Elsődleges tükrözések, Amerikai Egyesült Államok, Ausztrália, Brazília, Cseh Köztársaság, Dánia, Dél-Afrika, Észtország, Finnország, Franciaország, Hollandia, Írország, Japán, Korea, Lengyelország, Lettország, Litvánia, Németország, Norvégia, Olaszország, Örményország, Oroszország, Spanyolország, Svájc, Svédország, Szlovák Köztársaság, Szlovénia, Tajvan, Ukrajna.

(Dátum: UTC)

Központi szerverek

- cvsup.FreeBSD.org

Elsődleges tükrözések

- cvsup1.FreeBSD.org
- cvsup3.FreeBSD.org
- cvsup4.FreeBSD.org
- cvsup5.FreeBSD.org
- cvsup6.FreeBSD.org
- cvsup7.FreeBSD.org
- cvsup9.FreeBSD.org
- cvsup10.FreeBSD.org

- cvsup11.FreeBSD.org
- cvsup12.FreeBSD.org
- cvsup14.FreeBSD.org
- cvsup15.FreeBSD.org
- cvsup18.FreeBSD.org

Amerikai Egyesült Államok

- cvsup1.us.FreeBSD.org
- cvsup3.us.FreeBSD.org
- cvsup4.us.FreeBSD.org
- cvsup5.us.FreeBSD.org
- cvsup7.us.FreeBSD.org
- cvsup9.us.FreeBSD.org
- cvsup11.us.FreeBSD.org
- cvsup12.us.FreeBSD.org
- cvsup13.us.FreeBSD.org
- cvsup14.us.FreeBSD.org
- cvsup15.us.FreeBSD.org
- cvsup18.us.FreeBSD.org

Ausztrália

- cvsup.au.FreeBSD.org

Brazília

- cvsup2.br.FreeBSD.org

Cseh Köztársaság

- cvsup.cz.FreeBSD.org

Dánia

- cvsup.dk.FreeBSD.org

Dél-Afrika

- cvsup.za.FreeBSD.org

Észtország

- cvsup.ee.FreeBSD.org

Finnország

- cvsup.fi.FreeBSD.org

Franciaország

- cvsup3.fr.FreeBSD.org
- cvsup5.fr.FreeBSD.org
- cvsup8.fr.FreeBSD.org

Hollandia

- cvsup.nl.FreeBSD.org
- cvsup2.nl.FreeBSD.org
- cvsup3.nl.FreeBSD.org

Írország

- cvsup.ie.FreeBSD.org
- cvsup2.ie.FreeBSD.org

Japán

- cvsup.jp.FreeBSD.org
- cvsup2.jp.FreeBSD.org
- cvsup3.jp.FreeBSD.org
- cvsup4.jp.FreeBSD.org
- cvsup5.jp.FreeBSD.org
- cvsup6.jp.FreeBSD.org

Korea

- cvsup.kr.FreeBSD.org

Lengyelország

- cvsup.pl.FreeBSD.org

Lettország

- cvsup.lv.FreeBSD.org

Litvánia

- cvsup.lt.FreeBSD.org

Németország

- cvsup.de.FreeBSD.org
- cvsup2.de.FreeBSD.org
- cvsup4.de.FreeBSD.org
- cvsup5.de.FreeBSD.org

Norvégia

- cvsup.no.FreeBSD.org

Olaszország

- cvsup.it.FreeBSD.org

Örményország

- cvsup1.am.FreeBSD.org

Oroszország

- cvsup3.ru.FreeBSD.org
- cvsup5.ru.FreeBSD.org
- cvsup6.ru.FreeBSD.org

Spanyolország

- cvsup.es.FreeBSD.org
- cvsup2.es.FreeBSD.org
- cvsup3.es.FreeBSD.org

Svájc

- cvsup.ch.FreeBSD.org

Svédország

- cvsup.se.FreeBSD.org
- cvsup2.se.FreeBSD.org

Szlovák Köztársaság

- cvsup.sk.FreeBSD.org

Szlovénia

- cvsup.si.FreeBSD.org

Tajvan

- cvsup.tw.FreeBSD.org
- cvsup3.tw.FreeBSD.org
- cvsup6.tw.FreeBSD.org
- cvsup10.tw.FreeBSD.org
- cvsup11.tw.FreeBSD.org
- cvsup12.tw.FreeBSD.org
- cvsup13.tw.FreeBSD.org

Ukrajna

- cvsup5.ua.FreeBSD.org
- cvsup6.ua.FreeBSD.org

A.7. CVS címkék

Meg kell adnunk egy revízió címkéjét, amikor a **cv**s vagy **CVSup** használatával letöltjük vagy frissítjük a forrásokat. A revíziós címkék a FreeBSD egyik fejlesztési irányát vagy egy adott időpontbeli állapotát hivatkozzák. Az előbbi egy “ág címkéje”, míg az utóbbi pedig egy “kiadás címkéje”.

A.7.1. Az ágak címkéi

A HEAD kivételével (amely mindig egy érvényes címke) az összes címke csak a `src/` fára vonatkozik. A `ports/`, `doc/` és `www/` fák nem tartalmaznak ágakat.

HEAD

A fő fejlesztési ág, avagy a FreeBSD-CURRENT szimbolikus neve. Ha nem adunk meg revíziót, ez lesz az alapértelmezés.

A **CVSup** számára ezt . címke jelzi (itt most nem mondatvégi pontot jelöli, hanem a . karaktert).

Megjegyzés: A CVS számára ez lesz az alapértelmezett érték, ha nem adunk meg konkrét revíziós címkét. Többnyire *nem* túlzottan jó ötlet egy STABLE változatot használó gépen a CURRENT verziójú források kikérése, kivéve hacsak nem ez a szándékunk.

RELENG_8

A FreeBSD-8.X fejlesztési ága, más néven a FreeBSD 8-STABLE

RELENG_8_1

A FreeBSD-8.1 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_8_0

A FreeBSD-8.0 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_7

A FreeBSD-7.X fejlesztési ága, más néven a FreeBSD 7-STABLE

RELENG_7_3

A FreeBSD-7.3 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_7_2

A FreeBSD-7.2 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_7_1

A FreeBSD-7.1 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_7_0

A FreeBSD-7.0 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6

A FreeBSD-6.X fejlesztési ága, más néven a FreeBSD 6-STABLE

RELENG_6_4

A FreeBSD-6.4 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6_3

A FreeBSD-6.3 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6_2

A FreeBSD-6.2 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6_1

A FreeBSD-6.1 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6_0

A FreeBSD-6.0 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5

A FreeBSD-5.X fejlesztési ág, más néven a FreeBSD 5-STABLE.

RELENG_5_5

A FreeBSD-5.5 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_4

A FreeBSD-5.4 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_3

A FreeBSD-5.3 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_2

A FreeBSD-5.2 és FreeBSD-5.2.1 kiadások ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_1

A FreeBSD-5.1 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_0

A FreeBSD-5.0 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4

A FreeBSD-4.X fejlesztési ága, más néven a FreeBSD 4-STABLE.

RELENG_4_11

A FreeBSD-4.11 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_10

A FreeBSD-4.10 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_9

A FreeBSD-4.9 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_8

A FreeBSD-4.8 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_7

A FreeBSD-4.7 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_6

A FreeBSD-4.6 és FreeBSD-4.6.2 kiadások ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_5

A FreeBSD-4.5 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_4

A FreeBSD-4.4 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_3

A FreeBSD-4.3 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_3

A FreeBSD-3.X fejlesztési ága, más néven a 3.X-STABLE.

RELENG_2_2

A FreeBSD-2.2.X fejlesztési ága, más néven a 2.2-STABLE. Ez az ág manapság már elavult.

A.7.2. A kiadások címkéi

Ezek a címkék a FreeBSD egyes kiadásainak dátumára hivatkoznak. Egy kiadás előkészítésének és terjesztésének folyamatáról részleteiben a kiadásokat összefoglaló lapról (<http://www.FreeBSD.org/releng/>) és a kiadások építéséről szóló cikkből (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/releng/release-proc.html) tájékozódhatunk. Az `src` fában `RELENG_` kezdetű címkéket találunk. A `ports` és `doc` fákban a címkék nevei a `RELEASE` előtaggal kezdődnek. Végezetül a `www` fában nincsenek kiadásokhoz tartozó címkék.

RELENG_8_1_0_RELEASE

FreeBSD 8.1

RELENG_8_0_0_RELEASE

FreeBSD 8.0

RELENG_7_3_0_RELEASE

FreeBSD 7.3

RELENG_7_2_0_RELEASE

FreeBSD 7.2

RELENG_7_1_0_RELEASE

FreeBSD 7.1

RELENG_7_0_0_RELEASE

FreeBSD 7.0

RELENG_6_4_0_RELEASE

FreeBSD 6.4

RELENG_6_3_0_RELEASE

FreeBSD 6.3

RELENG_6_2_0_RELEASE

FreeBSD 6.2

RELENG_6_1_0_RELEASE

FreeBSD 6.1

RELENG_6_0_0_RELEASE

FreeBSD 6.0

RELENG_5_5_0_RELEASE

FreeBSD 5.5

RELENG_5_4_0_RELEASE

FreeBSD 5.4

RELENG_4_11_0_RELEASE

FreeBSD 4.11

RELENG_5_3_0_RELEASE

FreeBSD 5.3

RELENG_4_10_0_RELEASE

FreeBSD 4.10

RELENG_5_2_1_RELEASE

FreeBSD 5.2.1

RELENG_5_2_0_RELEASE

FreeBSD 5.2

RELENG_4_9_0_RELEASE

FreeBSD 4.9

RELENG_5_1_0_RELEASE

FreeBSD 5.1

RELENG_4_8_0_RELEASE

FreeBSD 4.8

RELENG_5_0_0_RELEASE

FreeBSD 5.0

RELENG_4_7_0_RELEASE

FreeBSD 4.7

RELENG_4_6_2_RELEASE

FreeBSD 4.6.2

RELENG_4_6_1_RELEASE

FreeBSD 4.6.1

RELENG_4_6_0_RELEASE

FreeBSD 4.6

RELENG_4_5_0_RELEASE

FreeBSD 4.5

RELENG_4_4_0_RELEASE

FreeBSD 4.4

RELENG_4_3_0_RELEASE

FreeBSD 4.3

RELENG_4_2_0_RELEASE

FreeBSD 4.2

RELENG_4_1_1_RELEASE

FreeBSD 4.1.1

RELENG_4_1_0_RELEASE

FreeBSD 4.1

RELENG_4_0_0_RELEASE

FreeBSD 4.0

RELENG_3_5_0_RELEASE

FreeBSD 3.5

RELENG_3_4_0_RELEASE

FreeBSD 3.4

RELENG_3_3_0_RELEASE

FreeBSD 3.3

RELENG_3_2_0_RELEASE

FreeBSD 3.2

RELENG_3_1_0_RELEASE

FreeBSD 3.1

RELENG_3_0_0_RELEASE

FreeBSD 3.0

RELENG_2_2_8_RELEASE

FreeBSD 2.2.8

RELENG_2_2_7_RELEASE

FreeBSD 2.2.7

RELENG_2_2_6_RELEASE

FreeBSD 2.2.6

RELENG_2_2_5_RELEASE

FreeBSD 2.2.5

RELENG_2_2_2_RELEASE

FreeBSD 2.2.2

RELENG_2_2_1_RELEASE

FreeBSD 2.2.1

RELENG_2_2_0_RELEASE

FreeBSD 2.2.0

A.8. AFS oldalak

A FreeBSD a következő szerverein érhető el AFS:

Svédország

Az állományok a következő helyen érhetőek el: `/afs/stacken.kth.se/ftp/pub/FreeBSD/`

<code>stacken.kth.se</code>	<code># Stacken Computer Club, KTH, Svédország</code>
<code>130.237.234.43</code>	<code>#hot.stacken.kth.se</code>
<code>130.237.237.230</code>	<code>#fishburger.stacken.kth.se</code>
<code>130.237.234.3</code>	<code>#milko.stacken.kth.se</code>

Karbantartó: <ftp@stacken.kth.se>

A.9. Rsync oldalak

A most következő oldalakon a FreeBSD-t érhetjük el az rsync protokollal. Az **rsync** segédprogram működésében leginkább a rcp(1) parancshoz hasonlít, de sokkal több beállítással rendelkezik, és az rsync távoli frissítéseket kezelő protokollja segítségével csak az állományok csoportjai között levő eltéréseket küldi át, amivel a hálózaton keresztüli szinkronizáció rendkívül felgyorsítható. Ez olyankor jelent számunkra a legtöbbet, ha a FreeBSD FTP szerverének vagy CVS repositoryjának egyik tükrözését tartjuk karban. Az **rsync** több operációs rendszerre is elérhető, és FreeBSD-n a net/rsync port vagy csomag tartalmazza.

Cseh Köztársaság

rsync://ftp.cz.FreeBSD.org/

Elérhető gyűjtemények:

- ftp: a FreeBSD FTP szerverének részleges tükrözése.
- FreeBSD: a FreeBSD FTP szerverének teljes tükrözése.

Hollandia

rsync://ftp.nl.FreeBSD.org/

Elérhető gyűjtemények:

- FreeBSD: a FreeBSD FTP szerverének teljes tükrözése.

Oroszország

rsync://ftp.mtu.ru

Elérhető gyűjtemények:

- FreeBSD: A FreeBSD FTP szerver teljes tartalma.
- FreeBSD-gnats: A GNATS hibanyilvántartó adatbázis.
- FreeBSD-Archive: A FreeBSD archívumait tároló FTP szerver tükrözése.

Tajvan

rsync://ftp.tw.FreeBSD.org/

rsync://ftp2.tw.FreeBSD.org/

rsync://ftp6.tw.FreeBSD.org/

Elérhető gyűjtemények:

- FreeBSD: a FreeBSD FTP szerverének teljes tükrözése.

Egyesült Királyság

`rsync://rsync.mirrorservice.org/`

Elérhető gyűjtemények:

- `sites/ftp.freebsd.org`: a FreeBSD FTP szerverének teljes tükrözése.

Amerikai Egyesült Államok

`rsync://ftp-master.FreeBSD.org/`

Ezt a szerveret csak az elsődleges FreeBSD tükrözéseknek szabad használniuk.

Elérhető gyűjtemények:

- `FreeBSD`: a FreeBSD FTP szerverének központi archívuma.
- `acl`: a FreeBSD központi ACL listája.

`rsync://ftp13.FreeBSD.org/`

Elérhető gyűjtemények:

- `FreeBSD`: a FreeBSD FTP szerver teljes tükrözése.

Függelék B. Irodalomjegyzék

Míg a man oldalak a FreeBSD operációs rendszer egyes önálló részeit tárgyalják, ismert a tény, hogy arról egyáltalán nem szólnak, miképpen illeszkednek egymáshoz ezek az alkotóelemek, és ezáltal hogyan működik maga az operációs rendszer. Erre a célra egyedül csak egy jó UNIX-os rendszeradminisztrációs szakkönyv és egy jó felhasználói kézikönyv alkalmas.

B.1. A FreeBSD-ről szóló könyvek és folyóiratok

Idegennyelvű könyvek és folyóiratok:

- Using FreeBSD (<http://jdli.tw.FreeBSD.org/publication/book/freebsd2/index.htm>) (kínai). Drmaster (<http://www.drmaster.com.tw/>), 1997. ISBN 9-578-39435-7.
- FreeBSD Unleashed (kínai fordítás). China Machine Press (<http://www.hzbook.com/>). ISBN 7-111-10201-0.
- FreeBSD From Scratch (1. kiadás, kínai). China Machine Press. ISBN 7-111-07482-3.
- FreeBSD From Scratch (2. kiadás, kínai). China Machine Press. ISBN 7-111-10286-X.
- FreeBSD Handbook (2. kiadás, kínai). Posts & Telecom Press (<http://www.ptpress.com.cn/>). ISBN 7-115-10541-3.
- FreeBSD 3.x Internet (kínai). Tsinghua University Press (<http://www.tup.tsinghua.edu.cn/>). ISBN 7-900625-66-6.
- FreeBSD & Windows (kínai). China Railway Publishing House (<http://www.tdpress.com/>). ISBN 7-113-03845-X
- FreeBSD Internet Services HOWTO (kínai). China Railway Publishing House. ISBN 7-113-03423-3
- FreeBSD for PC 98'ers (japán). SHUWA System Co, LTD. ISBN 4-87966-468-5 C3055 P2900E.
- FreeBSD (japán). CUTT. ISBN 4-906391-22-2 C3055 P2400E.
- Complete Introduction to FreeBSD (<http://www.shoeisha.com/book/Detail.asp?bid=650>) (japán). Shoeisha Co., Ltd (<http://www.shoeisha.co.jp/>). ISBN 4-88135-473-6 P3600E.
- Personal UNIX Starter Kit FreeBSD (<http://www.ascii.co.jp/pb/book1/shinkan/detail/1322785.html>) (japán). ASCII (<http://www.ascii.co.jp/>). ISBN 4-7561-1733-3 P3000E.
- FreeBSD Handbook (japán fordítás). ASCII (<http://www.ascii.co.jp/>). ISBN 4-7561-1580-2 P3800E.
- FreeBSD mit Methode (német). Computer und Literatur Verlag (<http://www.cul.de/>) Vertrieb Hanser, 1998. ISBN 3-932311-31-0.
- FreeBSD 4 - Installieren, Konfigurieren, Administrieren (<http://www.cul.de/freebsd.html>) (német). Computer und Literatur Verlag (<http://www.cul.de/>), 2001. ISBN 3-932311-88-4.
- FreeBSD 5 - Installieren, Konfigurieren, Administrieren (<http://www.cul.de/freebsd.html>) (német). Computer und Literatur Verlag (<http://www.cul.de/>), 2003. ISBN 3-936546-06-1.
- FreeBSD de Luxe (<http://www.mitp.de/vmi/mitp/detail/pWert/1343/>) (német). Verlag Moderne Industrie (<http://www.mitp.de/>), 2003. ISBN 3-8266-1343-0.
- FreeBSD Install and Utilization Manual (<http://www.pc.mycom.co.jp/FreeBSD/install-manual.html>) (japán). Mainichi Communications Inc. (<http://www.pc.mycom.co.jp/>), 1998. ISBN 4-8399-0112-0.
- Onno W Purbo, Dodi Maryanto, Syahrial Hubbany, Widjil Widodo *Building Internet Server with FreeBSD* (<http://maxwell.itb.ac.id/>) (indonéz nyelven). Elex Media Komputindo (<http://www.elexmedia.co.id/>).

- Absolute BSD: The Ultimate Guide to FreeBSD (kínai fordítás). GrandTech Press (<http://www.grandtech.com.tw/>), 2003. ISBN 986-7944-92-5.
- The FreeBSD 6.0 Book (<http://www.twbsd.org/cht/book/>) (kínai). Drmaster, 2006. ISBN 9-575-27878-X.

Angol nyelvű könyvek és folyóiratok:

- Absolute BSD, 2nd Edition: The Complete Guide to FreeBSD (<http://www.AbsoluteBSD.com/>). No Starch Press (<http://www.nostarch.com/>), 2007. ISBN: 978-1-59327-151-0
- The Complete FreeBSD (<http://www.freebsdmail.com/cgi-bin/fm/bsdcomp>). O'Reilly (<http://www.oreilly.com/>), 2003. ISBN: 0596005164
- The FreeBSD Corporate Networker's Guide (<http://www.freebsd-corp-net-guide.com/>). Addison-Wesley (<http://www.awl.com/awl/>), 2000. ISBN: 0201704811
- FreeBSD: An Open-Source Operating System for Your Personal Computer (<http://andrsn.stanford.edu/FreeBSD/introbook/>). The Bit Tree Press, 2001. ISBN: 0971204500
- Teach Yourself FreeBSD in 24 Hours. Sams (<http://www.sampublishing.com/>), 2002. ISBN: 0672324245
- FreeBSD 6 Unleashed. Sams (<http://www.sampublishing.com/>), 2006. ISBN: 0672328755
- FreeBSD: The Complete Reference. McGrawHill (<http://books.mcgraw-hill.com>), 2003. ISBN: 0072224096
- BSD Magazine (<http://www.bsdmag.org/>), megjelenik a Software Press Sp., z o.o. SK gondozásában. ISSN 1898-9144

B.2. Felhasználói kézikönyvek

- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-075-9
- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-076-7
- *UNIX in a Nutshell*. O'Reilly & Associates, Inc., 1990. ISBN 093717520X
- Mui, Linda. *What You Need To Know When You Can't Find Your UNIX System Administrator*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-104-6
- Ohio Állami Egyetemnek (<http://www.osu.edu/>) van egy Alapozó UNIX kurzusa (http://8help.osu.edu/wks/unix_course/index.html), amely az Interneten keresztül is elérhető HTML és PostScript formátumokban.

Ennek a dokumentumnak egy olasz fordítása

(http://www.FreeBSD.org/doc/it_IT.ISO8859-15/books/unix-introduction/index.html) is elérhető az Olasz FreeBSD Dokumentációs Projekt keretében.

- Jpman Project, Japanese FreeBSD User's Group (<http://www.jp.FreeBSD.org/>). FreeBSD User's Reference Manual (<http://www.pc.mycom.co.jp/FreeBSD/urm.html>) (japán fordítás). Mainichi Communications Inc. (<http://www.pc.mycom.co.jp/>), 1998. ISBN4-8399-0088-4 P3800E.
- Az Edinburghi Egyetemen (<http://www.ed.ac.uk/>) készítettek az újoncok számára egy Internetes kézikönyvet (<http://unixhelp.ed.ac.uk/>) a UNIX környezetekhez.

B.3. Rendszeradminisztrátori kézikönyvek

- Albitz, Paul and Liu, Cricket. *DNS and BIND* (4. kiadás). O'Reilly & Associates, Inc., 2001. ISBN 1-59600-158-4
- Computer Systems Research Group, UC Berkeley. *4.4BSD System Manager's Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-080-5
- Costales, Brian és mások. *Sendmail* (2. kiadás). O'Reilly & Associates, Inc., 1997. ISBN 1-56592-222-0
- Frisch, Æleen. *Essential System Administration* (2. kiadás). O'Reilly & Associates, Inc., 1995. ISBN 1-56592-127-5
- Hunt, Craig. *TCP/IP Network Administration* (2. kiadás). O'Reilly & Associates, Inc., 1997. ISBN 1-56592-322-7
- Nemeth, Evi. *UNIX System Administration Handbook* (3. kiadás). Prentice Hall, 2000. ISBN 0-13-020601-6
- Stern, Hal. *Managing NFS and NIS*. O'Reilly & Associates, Inc., 1991. ISBN 0-937175-75-7
- Jpman Project, Japan FreeBSD Users Group (<http://www.jp.FreeBSD.org/>). FreeBSD System Administrator's Manual (<http://www.pc.mycom.co.jp/FreeBSD/sam.html>) (japán fordítás). Mainichi Communications Inc. (<http://www.pc.mycom.co.jp/>), 1998. ISBN4-8399-0109-0 P3300E.
- Dreyfus, Emmanuel. Cahiers de l'Admin: BSD (<http://www.eyrolles.com/Informatique/Livre/9782212114638/>) (2. kiadás, franciául). Eyrolles, 2004. ISBN 2-212-11463-X

B.4. Programozói kézikönyvek

- Asente, Paul, Converse, Diana, and Swick, Ralph. *X Window System Toolkit*. Digital Press, 1998. ISBN 1-55558-178-1
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-078-3
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-079-1
- Harbison, Samuel P. and Steele, Guy L. Jr. *C: A Reference Manual* (4. kiadás). Prentice Hall, 1995. ISBN 0-13-326224-3
- Kernighan, Brian and Dennis M. Ritchie. *The C Programming Language* (2. kiadás). PTR Prentice Hall, 1988. ISBN 0-13-110362-8
- Lehey, Greg. *Porting UNIX Software*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-126-7
- Plauger, P. J. *The Standard C Library*. Prentice Hall, 1992. ISBN 0-13-131509-9
- Spinellis, Diomidis. *Code Reading: The Open Source Perspective* (<http://www.spinellis.gr/codereading/>). Addison-Wesley, 2003. ISBN 0-201-79940-5
- Spinellis, Diomidis. *Code Quality: The Open Source Perspective* (<http://www.spinellis.gr/codequality/>). Addison-Wesley, 2006. ISBN 0-321-16607-8
- Stevens, W. Richard and Stephen A. Rago. *Advanced Programming in the UNIX Environment* (2. kiadás). Reading, Mass. : Addison-Wesley, 2005. ISBN 0-201-43307-9
- Stevens, W. Richard. *UNIX Network Programming* (2. kiadás), PTR Prentice Hall, 1998. ISBN 0-13-490012-X

- Wells, Bill. "Writing Serial Drivers for UNIX". *Dr. Dobbs's Journal*. 19(15), 1994. december, 68-71. és 97-99. oldal.

B.5. Az operációs rendszerek belső működéséről

- Andleigh, Prabhat K. *UNIX System Architecture*. Prentice-Hall, Inc., 1990. ISBN 0-13-949843-5
- Jolitz, William. "Porting UNIX to the 386". *Dr. Dobbs's Journal*. 1991. január - 1992. július.
- Leffler, Samuel J., Marshall Kirk McKusick, Michael J Karels és John Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System*. Reading, Mass. : Addison-Wesley, 1989. ISBN 0-201-06196-1
- Leffler, Samuel J., Marshall Kirk McKusick. *The Design and Implementation of the 4.3BSD UNIX Operating System: Answer Book*. Reading, Mass. : Addison-Wesley, 1991. ISBN 0-201-54629-9
- McKusick, Marshall Kirk, Keith Bostic, Michael J Karels és John Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-54979-4
(A könyv 2. fejezete elérhető online (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/design-44bsd/book.html) a FreeBSD Dokumentációs Projekt részeként, valamint itt (http://www.netapp.com/tech_library/nfsbook.html) a 9. fejezet.)
- Marshall Kirk McKusick, George V. Neville-Neil. *The Design and Implementation of the FreeBSD Operating System*. Boston, Mass. : Addison-Wesley, 2004. ISBN 0-201-70245-2
- Stevens, W. Richard. *TCP/IP Illustrated, Vol 1: The Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63346-9
- Schimmel, Curt. *UNIX Systems for Modern Architectures*. Reading, Mass. : Addison-Wesley, 1994. ISBN 0-201-63338-8
- Stevens, W. Richard. *TCP/IP Illustrated, Vol 3: TCP for Transactions, HTTP, NNTP and the UNIX Domain Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63495-3
- Vahalia, Uresh. *UNIX Internals — The New Frontiers*. Prentice Hall, 1996. ISBN 0-13-101908-2
- Wright, Gary R. és W. Richard Stevens. *TCP/IP Illustrated, Vol 2: The Implementation*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63354-X

B.6. Biztonságról szóló írások

- Cheswick, William R. és Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63357-4
- Garfinkel, Simson és Gene Spafford. *Practical UNIX & Internet Security* (2. kiadás). O'Reilly & Associates, Inc., 1996. ISBN 1-56592-148-8
- Garfinkel, Simson. *PGP Pretty Good Privacy*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-098-8

B.7. Hardverrel foglalkozó írások

- Anderson, Don és Tom Shanley. *Pentium Processor System Architecture* (2. kiadás). Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40992-5
- Ferraro, Richard F. *Programmer's Guide to the EGA, VGA, and Super VGA Cards* (3. kiadás). Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-62490-7
- Az Intel által gyártott processzorokról és chipsetekről, valamint az általuk kialakított szabványokról a saját fejlesztői oldalukon (<http://developer.intel.com/>), általában PDF állományok formájában kaphatunk információkat.
- Shanley, Tom. *80486 System Architecture* (3. kiadás). Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40994-1
- Shanley, Tom. *ISA System Architecture* (3. kiadás). Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40996-8
- Shanley, Tom. *PCI System Architecture* (4. kiadás). Reading, Mass. : Addison-Wesley, 1999. ISBN 0-201-30974-2
- Van Gilluwe, Frank. *The Undocumented PC* (2. kiadás). Reading, Mass: Addison-Wesley Pub. Co., 1996. ISBN 0-201-47950-8
- Messmer, Hans-Peter. *The Indispensable PC Hardware Book* (4. kiadás). Reading, Mass: Addison-Wesley Pub. Co., 2002. ISBN 0-201-59616-4

B.8. UNIX történelem

- Lion, John. *Lion's Commentary on UNIX* (6. kiadás, forráskóddal). ITP Media Group, 1996. ISBN 1573980137
- Raymond, Eric S. *The New Hacker's Dictionary* (3. kiadás). MIT Press, 1996. ISBN 0-262-68092-0. Vagy Zsargon fájlként (<http://www.catb.org/~esr/jargon/html/index.html>) is ismert.
- Salus, Peter H. *A quarter century of UNIX*. Addison-Wesley Publishing Company, Inc., 1994. ISBN 0-201-54777-5
- Simon Garfinkel, Daniel Weise, Steven Strassmann. *The UNIX-HATERS Handbook*. IDG Books Worldwide, Inc., 1994. ISBN 1-56884-203-1. Elfogyott, de még elérhető ezen (<http://www.simson.net/ref/ugh.pdf>) a linken.
- Don Libes, Sandy Ressler. *Life with UNIX* — különkiadás. Prentice-Hall, Inc., 1989. ISBN 0-13-536657-7
- *The BSD family tree*. <http://www.FreeBSD.org/cgi/cvsweb.cgi/src/share/misc/bsd-family-tree> vagy egy telepített FreeBSD rendszeren a `/usr/share/misc/bsd-family-tree` állomány.
- *Networked Computer Science Technical Reports Library*. <http://www.ncstrl.org/>
- *Old BSD releases from the Computer Systems Research group (CSRG)*. <http://www.mckusick.com/csrp/Ez> a 4 CD-s készlet tartalmazza az összes BSD verziót a 1BSD-től kezdve a 4.4BSD és 4.4BSD-Lite2-ig (de nem a 2.11BSD-t sajnos nem). Az utolsó lemezen megtalálhatóak a végleges források, illetve az SCCS állományok.

B.9. Magazinok és folyóiratok

- *The C/C++ Users Journal*. R&D Publications Inc. ISSN 1075-2838

- *Sys Admin — The Journal for UNIX System Administrators*. Miller Freeman, Inc. ISSN 1061-2688
- *freeX — Das Magazin für Linux - BSD - UNIX* (német). Computer- und Literaturverlag GmbH. ISSN 1436-7033

Függelék C. Források az interneten

A FreeBSD gyors ütemű fejlődése a nyomtatott médiát alkalmatlanná teszi a legfrissebb fejlesztések nyomonkövetésére. Ezzel szemben az elektronikus erőforrások a biztos, ha gyakran nem is csak az egyetlen, módjai a legújabb előrelépések figyelemmel követésének. Mivel a FreeBSD-t többségében önkéntesek fejlesztik, az őt körülvevő felhasználói közösség önmaga is egyfajta “szakmai segélynyújtó egyletként” funkcionál, amelyet leghatékonyabban elektronikus levélben, webes fórumokon vagy USENET hírcsoportokon keresztül érhetünk el.

A továbbiakban a FreeBSD felhasználók közösségének különböző fajtájú elérhetőségeit vázoljuk fel nagyvonalakban. Ha úgy érezzük, hogy ebből a felsorolásban kimaradt volna valami, akkor ne habozzunk róla értesítést küldeni a FreeBSD Dokumentációs Projekt levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc>) címére (angolul), hogy felvehessük a többi közé.

C.1. Levelezési listák

A FreeBSD köré csoportosulókat levelezési listákon keresztül tudjuk közvetlenül elérni, ezen a módon tehetünk fel kérdéseket, vethetünk fel témákat. Ezek között több különböző területtel foglalkozó listát találhatunk. Ezért célszerű mindig a hozzászólásainkat a témánkhoz legközelebb álló listára küldeni, mert enélkül szinte biztos, hogy nem kapunk pontos vagy gyors választ.

A különböző listák témájának rövid leírása a dokumentum alján olvasható. *Szeretnénk mindenkit megkérni, hogy mielőtt feliratkozik vagy levelet küld valamelyik listára, figyelmesen olvassa el ezeket.* Az egyes listák tagjai már így is naponta többszáz FreeBSD-vel kapcsolatos üzenetet kapnak, miközben a listák tematikájának és szabályainak lefektetésével igyekszünk a jel-zaj arányt minél kedvezőbb szinten tartani. Ezek nélkül a levelezési listák a Projekt számára haszontalan kommunikációs eszközökké válnának.

Megjegyzés: A `freebsd-test` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-test>) címet használjuk, ha ki akarjuk próbálni, hogy tudunk-e levelet küldeni a FreeBSD listáira. A többi listára viszont lehetőleg ne küldjünk teszt jellegű üzeneteket.

Ha nem tudjuk eldönteni, hogy pontosan melyik listát is kellene megcímeznünk kérdésünkkel, olvassuk el a *Hogyan kapjunk értékelhető választ a FreeBSD-questions levelezési listáról* (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/freebsd-questions) című leírást (angolul).

Mielőtt akármelyik listára is levelet küldenénk, olvassuk el a *Levelezési listák Gyakran Ismételt Kérdéseit* (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/mailling-list-faq) (angolul), amivel elkerülhetjük a gyakran feltett kérdések és témák ismételt felhozását.

A levelezési listák tartalma folyamatosan archiválódik, és ezekben az archívumokban a FreeBSD honlapján (<http://www.FreeBSD.org/search/index.html>) tudunk keresni. Az itt elérhető, kulcsszavak alapján történő keresés remek módját nyújtja a gyakran felmerülő kérdések egyszerű és gyors megválaszolásának, ezért ilyen esetekben először mindig ezt javasolt használni. Ez egyben mellesleg azt is jelenti, hogy a FreeBSD levelezési listáira küldött üzenetek fennmaradnak az örökkévalóságig. Ha a beküldendő üzenet bizalmas információkat tartalmaz, érdemes megfontolni egy eldobható anonim e-mail cím használatát és kizárólag csak a publikus részt beküldeni.

C.1.1. A listák összefoglalása

Általános listák: A következő általános célú listákhoz szabadon (és nyugodtan) csatlakozhatunk:

Lista	Tartalom
freebsd-advocacy (http://lists.FreeBSD.org/mailman/listinfo/freebsd-advocacy)	A FreeBSD igéjének terjesztése
freebsd-announce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce)	Fontosabb események és előrelépések a projektek életében
freebsd-arch (http://lists.FreeBSD.org/mailman/listinfo/freebsd-arch)	Architektúrális és tervezési kérdések tárgyalása
freebsd-bugbusters (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugbusters)	A FreeBSD hibabejelentéseit tároló adatbázis és a kapcsolódó eszközök karbantartására vonatkozó megbeszélések
freebsd-bugs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugs)	Hibajelentések
freebsd-chat (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat)	A FreeBSD közösség nem szakmai jellegű dolgai
freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current)	A FreeBSD-CURRENT használatának tárgyalása
freebsd-isp (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp)	A FreeBSD-t alkalmazó internet-szolgáltatók fóruma
freebsd-jobs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-jobs)	FreeBSD-s munkalehetőségek
freebsd-policy (http://lists.FreeBSD.org/mailman/listinfo/freebsd-policy)	A FreeBSD fejlődését irányító csoport (Core Team) döntéseiről tájékoztató lista. A forgalma kicsi, csak olvasható.
freebsd-questions (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions)	A felhasználók kérdései és szakmai segítségnyújtás
freebsd-security-notifications (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications)	Biztonsági figyelmeztetések
freebsd-stable (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable)	A FreeBSD-STABLE használatát illető kérdések
freebsd-test (http://lists.FreeBSD.org/mailman/listinfo/freebsd-test)	Ide lehet küldeni a próbaüzeneteket

Szakmai listák: A következő listák szakmai jellegű témákat képviselnek. Mielőtt bármelyikükre levelet küldenénk vagy feliratkoznánk, figyelmesen olvassuk el a tartalmukat és céljaikat bemutató rövid leírásukat.

Lista	Tartalom
freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi)	Az ACPI és energiagazdálkodás támogatás fejlesztése
freebsd-afs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-afs)	Az AFS portolása FreeBSD-re
freebsd-aic7xxx (http://lists.FreeBSD.org/mailman/listinfo/aic7xxx)	Az Adaptec AIC 7xxx sorozat meghajtóinak fejlesztése
freebsd-alpha (http://lists.FreeBSD.org/mailman/listinfo/freebsd-alpha)	A FreeBSD Alpha portja
freebsd-amd64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-amd64)	A FreeBSD AMD64 portja
freebsd-apache (http://lists.FreeBSD.org/mailman/listinfo/freebsd-apache)	Az Apache és hozzá tartozó portok tárgyalása
freebsd-arm (http://lists.FreeBSD.org/mailman/listinfo/freebsd-arm)	A FreeBSD ARM® portja
freebsd-atm (http://lists.FreeBSD.org/mailman/listinfo/freebsd-atm)	FreeBSD használata ATM hálózatokkal
freebsd-audit (http://lists.FreeBSD.org/mailman/listinfo/freebsd-audit)	A forráskód ellenőrzéséről szóló projekt
freebsd-binup (http://lists.FreeBSD.org/mailman/listinfo/freebsd-binup)	A bináris frissítésekkel foglalkozó rendszer tervezése és fejlesztése
freebsd-bluetooth (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bluetooth)	A Bluetooth technológia használata a FreeBSD-ben
freebsd-cluster (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cluster)	A FreeBSD klaszteres környezetben
freebsd-cvsweb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cvsweb)	A CVSweb karbantartása

Lista

freebsd-database

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-database>)

freebsd-doc

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc>)

freebsd-drivers

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-drivers>)

freebsd-eclipse

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-eclipse>)

freebsd-embedded

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded>)

freebsd-eol

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-eol>)

freebsd-emulation

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-emulation>)

freebsd-firewire

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-firewire>)

freebsd-fs

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-fs>)

freebsd-gecko

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-gecko>)

freebsd-geom

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-geom>)

freebsd-gnome

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-gnome>)

Tartalom

Adatbázisok használata és fejlesztése FreeBSD alatt

FreeBSD-ről szóló leírások készítése

Eszközmeghajtók írása FreeBSD-re

Az Eclipse integrált fejlesztői környezet, eszközeinek, gazdag kliens alkalmazásinak és portjainak FreeBSD alatti használata

A FreeBSD használata beágyazott alkalmazásokban

Olyan FreeBSD-s szoftverek független továbbfejlesztése, amelyeket hivatalosan már nem támogatnak

Linux/MS-DOS/Windows és hasonló rendszerek emulációja

A FreeBSD és a FireWire® (iLink, IEEE 1394) kapcsolatának technikai kérdései

Állományrendszerek

A **Gecko Rendering Engine** alkalmazásával kapcsolatos problémák

A GEOM-hoz tartozó témák és implementációk

A **GNOME** és **GNOME**-alkalmazások portolása

Lista

freebsd-hackers
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>)

freebsd-hardware
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hardware>)

freebsd-i18n
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-i18n>)

freebsd-ia32
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia32>)

freebsd-ia64
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia64>)

freebsd-ipfw
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ipfw>)

freebsd-isdn
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn>)

freebsd-jail
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-jail>)

freebsd-java
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-java>)

freebsd-kde
(<https://mail.kde.org/mailman/listinfo/kde-freebsd>)

freebsd-lfs
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-lfs>)

freebsd-libh
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-libh>)

freebsd-mips
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-mips>)

freebsd-mobile
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-mobile>)

freebsd-mono
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-mono>)

freebsd-mozilla
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-mozilla>)

Tartalom

Általános szakmai témák

A FreeBSD futtatására szolgáló hardverekkel foglalkozó témák

A FreeBSD honosítása

A FreeBSD használata az IA-32 (Intel x86) platformon

A FreeBSD portolása az Intel következő IA64 rendszereire

Az IP tűzfal kódjának újratervezését érintő szakmai megbeszélések

ISDN fejlesztők levelei

A jail(8) segédprogram

Java fejlesztők kérdései és a JDK-k átültetése FreeBSD-re

A **KDE** és **KDE**-alkalmazások portolása

Az LFS portolása FreeBSD-re

A második generációs telepítő- és csomagrendszer

A FreeBSD portolása MIPS®-re

A mobil számítógépekkel kapcsolatos megbeszélések

Mono és C# alkalmazások FreeBSD alatt

A **Mozilla** átültetése FreeBSD-re

Lista

freebsd-multimedia

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-multimedia>)

freebsd-new-bus

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-new-bus>)

freebsd-net

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-net>)

freebsd-openoffice

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-openoffice>)

freebsd-performance

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-performance>)

freebsd-perl

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-perl>)

freebsd-pf

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf>)

freebsd-platforms

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-platforms>)

freebsd-ports

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports>)

freebsd-ports-bugs

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs>)

freebsd-ppc

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ppc>)

freebsd-proliant

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-proliant>)

freebsd-python

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-python>)

freebsd-qa

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-qa>)

Tartalom

Multimédia alkalmazások

A buszarchitektúrával kapcsolatos szakmai megbeszélések

A TCP/IP forráskódjával és hálózatkezeléssel kapcsolatos kérdések

A **OpenOffice.org** és **StarOffice** alkalmazások portolása FreeBSD-re

Nagy terhelésű és teljesítményű rendszerek teljesítményhangolási kérdései

A rengeteg Perl alapú port karbantársa

A csomagszűrő működésével kapcsolatos kérdések és megbeszélések

Portolás nem Intel architektúrájú platformokra

A Portgyűjtemény működése

A portokhoz tartozó hibák és hibajelentések megbeszélése

A FreeBSD portolása PowerPC®-re

HP ProLiant szerverek és a FreeBSD kapcsolata

A Python FreeBSD-n futó változatának problémái

A minőségbiztosítás megbeszélése, különösen a kiadások közeledtével

Lista

freebsd-rc
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-rc>)

freebsd-realtime
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-realtime>)

freebsd-ruby
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ruby>)

freebsd-scsi
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-scsi>)

freebsd-security
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security>)

freebsd-small
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-small>)

freebsd-smp
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-smp>)

freebsd-sparc64
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-sparc64>)

freebsd-standards
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-standards>)

freebsd-sun4v
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-sun4v>)

freebsd-sysinstall
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-sysinstall>)

freebsd-threads
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-threads>)

Tartalom

Az `rc.d` rendszer és annak fejlődése

A FreeBSD valós idejű kiterjesztéseinek fejlesztése

A Ruby használata FreeBSD rendszereken

A SCSI alrendszer

A FreeBSD működését fenyegető biztonsági problémák

A FreeBSD használata beágyazott alkalmazásokban (elavult; helyette a `freebsd-embedded` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded>) címét használjuk)

Az [A]Szimmetrikus többszálú feldolgozáshoz ([A]Symmetric MultiProcessing) tartozó tervezési megbeszélések

A FreeBSD portolása SPARC® alapú rendszerekre

A FreeBSD megfelelése a C99 és POSIX szabványoknak

A FreeBSD portolása UltraSPARC T1 alapú rendszerekre

A `sysinstall(8)` fejlesztése

A FreeBSD szálkezelése

Lista

freebsd-testing
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-testing>)

freebsd-tilera
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-tilera>)

freebsd-tokenring
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-tokenring>)

freebsd-toolchain
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-toolchain>)

freebsd-usb
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-usb>)

freebsd-virtualization
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-virtualization>)

freebsd-vuxml
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-vuxml>)

freebsd-x11
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-x11>)

freebsd-xen
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-xen>)

Korlátozott listák: (Limited lists) A következő listák sokkal jobban specializálódtak (és igényesebb) közösségnek szólnak, nem a nagyközönségnek. Ezért mielőtt egy ilyen listára feliratkoznánk, érdemes némi tapasztalatot gyűjtenünk a szakmai témájú listákon, így megismerjük az itt alkalmazott kommunikációs szabályokat.

Tartalom

A FreeBSD teljesítmény- és megbízhatósági tesztjei

A FreeBSD portolása a Tilera processzorcsalád tagjaira

A Token Ring támogatása a FreeBSD-ben

A FreeBSD alapvető segédprogramjainak karbantartása

USB támogatás a FreeBSD-ben

A FreeBSD részéről támogatott különböző virtualizációs technológiák tárgyalása

A VuXML infrastruktúra tárgyalása

Az X11 karbantartása és támogatása FreeBSD alatt

A Xen™ FreeBSD portjának (implementációk, használat) tárgyalása

Lista

freebsd-hubs
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hubs>)

freebsd-user-groups
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-user-groups>)

freebsd-vendors
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-vendors>)

Tartalom

A tükrözések üzemeltetői számára (infrastrukturális támogatás)

A felhasználói csoportok összefogása

A forgalmazók koordinálása a kiadások előtt

Lista

freebsd-wip-status
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-wip-status>)

freebsd-www
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-www>)

Tartalom

A FreeBSD-vel kapcsolatos folyamatban levő fejlesztések helyzetjelentései

A www.FreeBSD.org (<http://www.FreeBSD.org/index.html>) karbantartói számára

Kivonatolt listák: (Digest lists) Az eddig említett listák elérhetőek kivonatolt formában is. Miután feliratkoztunk egy listára, a hozzáférésünk beállításainál kiválaszthatjuk, hogy kivonatolt formátumban kívánjuk-e kapni a leveleket.

CVS és SVN listák: (CVS & SVN lists) A következő listák a forrásfa különböző részeinek változtatásáról és a hozzájuk tartozó üzenetekről adnak értesítést. Ezek a listák *csak olvasásra* vannak, nem szabad rájuk levelet küldeni.

Lista	Forráskód területe	A terület leírása (minek a forrása)
cvcs-all (http://lists.FreeBSD.org/mailman/listinfo/cvcs-all)	/usr/(CVSROOT doc ports)	A fában végzett akármelyik módosítás (az összes CVS lista együtt)
cvcs-doc (http://lists.FreeBSD.org/mailman/listinfo/cvcs-doc)	/usr/(doc www)	A doc és www ágak változásai
cvcs-ports (http://lists.FreeBSD.org/mailman/listinfo/cvcs-ports)	/usr/ports	A portfa változásai
cvcs-projects (http://lists.FreeBSD.org/mailman/listinfo/cvcs-projects)	/usr/projects	A projektek változásai
cvcs-src (http://lists.FreeBSD.org/mailman/listinfo/cvcs-src)	/usr/src	A rendszer forrásának változásai (az svn és cvs közti importer működése alapján generálódik)
svn-src-all (http://lists.FreeBSD.org/mailman/listinfo/svn-src-all)	/usr/src	A Subversion repositoryk változásai (kivéve a user és a projects)
svn-src-head (http://lists.FreeBSD.org/mailman/listinfo/svn-src-head)	/usr/src	A Subversion repository "főágának" (a FreeBSD-CURRENT forrásainak) változásai

Lista	Forráskód területe	A terület leírása (minek a forrása)
svn-src-projects (http://lists.FreeBSD.org/mailman/listinfo/svn-src-projects)	/usr/projects	A projects változásai a forrásokat tároló Subversion repositoryn belül
svn-src-release (http://lists.FreeBSD.org/mailman/listinfo/svn-src-release)	/usr/src	A releases változásai a forrásokat tároló Subversion repositoryn belül
svn-src-releng (http://lists.FreeBSD.org/mailman/listinfo/svn-src-releng)	/usr/src	A releng ágak (biztonsági frissítések és kiadások) változásai a forrásokat tároló Subversion repositoryn belül
svn-src-stable (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable)	/usr/src	A stabil verziókhoz tartozó ágak változásai a forrásokat tároló Subversion repositoryn belül
svn-src-stable-6 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-6)	/usr/src	A stable/6 ág változásai a forrásokat tároló Subversion repositoryn belül
svn-src-stable-7 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-7)	/usr/src	A stable/7 ág változásai a forrásokat tároló Subversion repositoryn belül
svn-src-stable-8 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-8)	/usr/src	A stable/8 ág változásai a forrásokat tároló Subversion repositoryn belül
svn-src-stable-other (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-other)	/usr/src	A Subversion repositoryban található korábbi stable ágak változásai
svn-src-svnadmin (http://lists.FreeBSD.org/mailman/listinfo/svn-src-svnadmin)	/usr/src	A forrásokat tároló Subversion repositoryhoz tartozó szkriptek és egy konfigurációs állományok változásai
svn-src-user (http://lists.FreeBSD.org/mailman/listinfo/svn-src-user)	/usr/src	A user változásai a forrásokat tároló Subversion repositoryn belül
svn-src-vendor (http://lists.FreeBSD.org/mailman/listinfo/svn-src-vendor)	/usr/src	A vendor változásai a forrásokat tároló Subversion repositoryn belül

C.1.2. Hogyan iratkozzunk fel

Ha fel akarunk iratkozni valamelyik listára, kattintsunk a nevére, vagy menjünk a <http://lists.FreeBSD.org/mailman/listinfo> címre és a válasszuk ki onnan a keresett listát. A lista oldalán megtalálunk minden feliratkozással kapcsolatos utasítást.

Ténylegesen úgy tudunk üzeni egy listára, ha levelet küldünk az `<listanév@FreeBSD.org>` címre, amely ezután a lista tagjai között kézbesítésre kerül a világban.

A listáról úgy tudunk leiratkozni, ha a róla kapott valamelyik levél alján található URL-re kattintunk. Másik megoldás, ha magunk küldünk egy levelet a `<listanév-unsubscribe@FreeBSD.org>` címre.

Még egyszer szeretnénk kérni, hogy a szakmai témájú levelezési listákon folyó társalgásokat igyekezzünk az adott témán belül tartani. Ha csupán a fontosabb bejelentésekre vagyunk kíváncsiak, akkor a kisforgalmú FreeBSD announcements levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce>) használatát válasszuk.

C.1.3. A listák tematikája

Minden FreeBSD-s levelezési lista rendelkezik bizonyos alapszabályokkal, amelyek minden tagnak el kell fogadnia. Az ismeretett irányelvek elleni vétkezés a FreeBSD postamesterének `<postmaster@FreeBSD.org>` két (2, azaz kettő) írásos figyelmeztetését vonja maga után, amelyek figyelmen kívül hagyásával, tehát a harmadik szabálysértés alkalmával, a küldő eltávolításra kerül a FreeBSD összes levelezési listájáról és a továbbiakban szűrni fogják a leveleit. Sajnáljuk, hogy ilyen szabályokat és szankciókat kellett bevezetnünk, de napjaink internetes technológiái igen elvadultak és ahogy az látható is, sokan egyszerűen nem fogják fel, mennyire sérülékenyek egyes részei.

Közlekedési szabályok:

- Minden beküldött levél témájának meg kell felelnie az adott lista tartalmának, tehát például a szakmai kérdésekkel foglalkozó listákon csak szakmai témájú leveleknek szabad megjelenniük. Az oda nem illő cseverészés és értelmetlen vitázás csak a lista értékét csökkenti, ezért ezt senkitől sem tűrjük. A kötetlenebb, konkrét téma nélküli megbeszéléseket inkább a FreeBSD chat levelezési lista (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat>) címén folytassuk.
- 2 listánál többre ne küldjük be ugyanazt a levelet, és 2 listára is csak akkor küldjük, ha az egyértelműen és nyilvánvalóan indokolt. A legtöbb listánál így is rengeteg az átfedés, kivéve a legtökesebb kombinációkat (például “-stable és -scsi”), ezért nem túl sok értelme van egyszerre egynél több listát is értesíteni. Ha olyan üzenetet kapunk, amelynek a Cc (másolat) mezőjében több lista címe is szerepel, akkor továbbküldés vagy válaszadás során töröljük ezeket. *Az általunk küldött levelekért továbbra is mi magunk vagyunk a felelősek, függetlenül attól, hogy ki volt a levél eredeti feladója.*
- Tilos (vita közben) személyeskedni vagy káromkodni, beleértve a felhasználókat és a fejlesztőket is. A netikett megszegését, például a privát levelezés előzetes engedély nélküli továbbküldését vagy egyes részleteinek közzétételét, elítéljük, de nyíltan nem tiltjuk. Nagyon ritka esetekben *azonban* előfordulhat, hogy a sértő tartalom önmagában ellenkezik a lista elveivel és figyelmeztetést (esetleg kitiltást) von maga után.
- A FreeBSD-hez nem kötődő termékek vagy szolgáltatások reklámozása szigorúan tilos, és ha bebizonyosodik, hogy a küldő szándékosan küldte szét, akkor azonnali kitiltásban részesül.

Az egyes listák tematikája:

freebsd-acpi (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>)

Az ACPI és energiagazdálkodás támogatásának fejlesztése

freebsd-afs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-afs>)

Andrew File System

Ez a lista a CMU/Transarc AFS portolásáról szól

freebsd-announce (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce>)

Fontosabb események / nagyobb lépések

Olyan emberek számára ajánlott ez a levelezési lista, akik csak a FreeBSD jelentősebb eseményei bejelentései iránt érdeklődnek. Ide értendők a különböző időközi és egyéb kiadások, a FreeBSD újításainak bejelentései. Időnként önkéntesek toborzására stb. is használják. A forgalma nagyon kicsi, tartalma szigorúan ellenőrzött.

freebsd-arch (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-arch>)

Architektúrális és tervezési kérdések

Ez a lista a FreeBSD architektúráját érintő megbeszélések színtere. Az itt megjelenő üzenetek szigorúan szakmai jellegűek. Néhány idevágó téma:

- Hogyan alakítsuk úgy át a fordítási rendszert, hogy egyszerre több különböző paraméterű fordítás is képes legyen futni.
- Mit kellene javítani a VFS-en a Heidemann-rétegek működéséhez.
- Hogyan tudnánk úgy átalakítani az eszközmeghajtók felületét, hogy ugyanazok a meghajtók minden gond nélkül képesek legyenek több buszon és architektúrán is működni.
- Hogyan írjunk meghajtót hálózati eszközökhöz.

freebsd-audit (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-audit>)

A forráskód vizsgálatát végző projekt

Ez a levelezési lista a FreeBSD forráskódjának vizsgálatával foglalkozik. Habár eredetileg csak a biztonságot érintő változtatások ellenőrzésére jött létre, napjainkra már a forráskód mindenféle változását felülvizsgálja.

Erre a listára rengeteg javítás érkezik, amelyek valószínűleg egy átlag FreeBSD felhasználó számára nem túlzottan érdekesek. A kód változásától független biztonsági kérdések megvitatása a freebsd-security listán történik. Viszont az összes fejlesztőnek javasoljuk, hogy küldjék be felülvizsgálatra a javításaikat, különösen abban az esetben, amikor a forráskód olyan részéhez nyúlnak, ahol az adott hiba javítása a rendszer egészének működésére kihatással lehet.

freebsd-binup (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-binup>)

A FreeBSD bináris frissítésével foglalkozó projekt

Ez a lista ad otthont a **binup** vagy más néven a bináris frissítési rendszer (binary update system) körül felmerülő problémák tárgyalásának. Tervezési kérdések, implementációs részletek, javítások, hiba- és állapotjelentések, funkciók igénylése, a kód változásainak naplózása és minden, ami a **binup**pal kapcsolatos.

freebsd-bluetooth (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-bluetooth>)

Bluetooth a FreeBSD-ben

Ez a Bluetooth-os FreeBSD felhasználók gyülekezőhelye. Tervezési és implementációs kérdések, javítások, hiba- és állapotjelentések, funkciók igénylése, minden, ami Bluetooth.

freebsd-bugbusters (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugbusters>)

A hibajelentések kezelésének összefogása

A lista célja a Bugmeister és az ő Bugbustereinek, valamint a hibajelentések adatbázisai iránti kifejezetten érdeklődő személyek együttműködésének és kapcsolattartásának elősegítése. Ez a lista nem az egyes hibákról, javításokról vagy azok jelentéséről szól.

freebsd-bugs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugs>)

Hibajelentések

Ezen a levelezési listán lehet a FreeBSD hibáit bejelenteni. Ha lehet, akkor a hibákat a send-pr(1) paranccsal vagy a webes felületen (<http://www.FreeBSD.org/send-pr.html>) keresztül küldjük be.

freebsd-chat (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat>)

A FreeBSD közösség nem szakmai jellegű dolgai

Erre a listára kerül minden olyan nem szakmai jellegű, társadalmi érintkezéssel kapcsolatos információ, ami a többi listáról kimaradt: Jordan mennyire hasonlít a rajzfilmekben látható vadászgörényre, kis- vagy nagybetűvel írjuk-e, ki iszik sok kávé, hol főzik a legjobb söröket, ki főz sört az alagsorában és így tovább. Elvértve felbukkannak olyan fontosabb események is (bulik, lakodalmak, gyermekáldás, új munkahely stb), amelyek ugyan szakmai témájúak, de a folyományaik már inkább a -chat listára tartoznak.

freebsd-core

A FreeBSD irányítását végző csapat

Ezt a belső levelezési listát a Core Team tagjai használják. Akkor érdemes ide levelet küldeni, ha FreeBSD-vel kapcsolatos fontos ügyekben lenne szükségünk döntésre vagy véleményre.

freebsd-current (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>)

A FreeBSD-CURRENT használatával kapcsolatos megbeszélések

A FreeBSD-CURRENT felhasználóinak levelezési listája. Itt értesülhetünk a -CURRENT felhasználókat érintő friss újdonságairól, és azokról az utasításokról, amelyek követésével működésképesen tarthatjuk a -CURRENT rendszerünket. Aki a "-CURRENT" verziót használja, mindenképpen iratkozzon fel erre a listára. Ez is egy szakmai jellegű lista, ahová csak szigorúan ilyen témákat várnak.

freebsd-cvsweb (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-cvsweb>)

A FreeBSD CVSweb projekt

A FreeBSD CVSweb szolgáltatásának használatáról, fejlesztéséről és karbantartásáról szóló megbeszélések.

freebsd-doc (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc>)

A dokumentációs projekt

Ez a levelezési lista a FreeBSD-ről szóló különböző dokumentumok készítésével kapcsolatos problémák és projektek tárgyalásait öleli fel. A levelezési lista tagjait együttesen a "FreeBSD Dokumentációs Projekt"-nek nevezik. Ez egy nyílt lista, csatlakozzunk hozzá bátran!

freebsd-drivers (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-drivers>)

Eszközmeghajtók írása FreeBSD-re

A FreeBSD-hez készülő eszközmeghajtókról szóló szakmai fórum. Elsősorban itt tehetik fel a meghajtók készítői a FreeBSD rendszermagjában megtalálható API-kra vonatkozó kérdéseiket.

freebsd-eclipse (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-eclipse>)

Az Eclipse integrált fejlesztői környezetének, segéprogramjainak, kliensalkalmazásainak és portjainak FreeBSD felhasználók számára meghirdetett fóruma.

A lista azzal a szándékkal jött létre, hogy kölcsönös támogatást nyújtson az Eclipse fejlesztői környezet, a hozzá tartozó segédeszközök, kliensalkalmazások FreeBSD változatának megválasztásában, telepítésében és használatában. Emellett az Eclipse környezet és pluginjainak FreeBSD-re történő portolásáról is szó esik.

Valamint igyekszik minél többet profitálni az Eclipse és a FreeBSD köré csoportosuló közösségek kölcsönös információcseréjéből.

Habár a lista elsődlegesen az Eclipse felhasználóinak igényeire koncentrál, azok számára is táptalajt ad, akik az Eclipse keretrendszer segítségével FreeBSD specifikus alkalmazásokat szeretnének kifejleszteni.

freebsd-embedded (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded>)

A FreeBSD használata beágyazott alkalmazásokban

Ez a lista a FreeBSD beágyazott rendszerekben történő használatát igyekszik megvitatni. Ez egy szakmai jellegű lista, ezért ide szigorúan csak ilyen témájú leveleket várunk. A listán tárgyalt beágyazott rendszereknek tekintünk minden olyan számítási eszközt, amely az általános számítási környezetekkel szemben egyetlen feladatot lát el. Nem feltétlenül csak ilyenek, de például a különféle telefonok, illetve hálózati eszközök, mint például útválasztók, switchek, PBX-ek, távoli mérőeszközök, PDA-k, eladási rendszerek és így tovább.

freebsd-emulation (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-emulation>)

A Linux/MS-DOS/Windows rendszerek emulációja

Ezen a listán arról értekezhetünk és olvashatunk, hogy FreeBSD alatt miként futtassunk más operációs rendszerekre írt programokat.

freebsd-eol (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-eol>)

Összefogás a FreeBSD Projekt által tovább már támogatott, FreeBSD-hez tartozó szoftverekért

Ezen a listán kap vagy kaphat helyet a FreeBSD Projekt által hivatalosan tovább már nem fejlesztett szoftverek felhasználói összefogáson alapuló támogatása (például biztonsági figyelmeztetések vagy javítások formájában).

freebsd-firewire (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-firewire>)

FireWire (iLink, IEEE 1394)

Ez a levelezési lista foglalkozik a FreeBSD FireWire (azaz IEEE 1394, avagy iLink) alrendszerének implementációjával. Az itt felmerülő témák többek közt a szabványok, buszos eszközök és a hozzájuk tartozó protokollok, vezérlőkártyák és chipkészletek, valamint a működtetésükre szánt programok felépítése és megvalósítása.

freebsd-fs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-fs>)

Állományrendszerek

A FreeBSD-ben megjelenő állományrendszerek kivesézése. Mivel ez egy szakmai jellegű lista, ide határozottan csak ilyen jellegű leveleket várunk.

freebsd-gecko (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-gecko>)

Gecko Rendering Engine

Ezen a levelezési listán a **Gecko** FreeBSD rendszerekre portolt változatával kapcsolatos fórumot találjuk.

Az itt felmerülő témák többségükben a Gecko alapú alkalmazásokról, telepítésükről, és a FreeBSD alatti fejlesztésükről, támogatásukról szólnak.

freebsd-geom (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-geom>)

GEOM

A GEOM és a vele kapcsolatos implementáció megbeszélései. Szakmai jellegű lista, ezért erre tekintettel csak ilyen témájú leveleket postázzunk ide.

freebsd-gnome (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-gnome>)

GNOME

A **GNOME** asztalkörnyezet FreeBSD rendszereket érintő használatáról szóló lista. Műszaki jellegű, ezért szigorúan csak ilyen témákban társalgodjunk itt.

freebsd-ipfw (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ipfw>)

IP tűzfalak

A FreeBSD-ben levő IP tűzfal újratervezésével foglalkozó elgondolások és szakmai témájú megbeszélések otthona. Ide szigorúan csak ilyen témájú leveleket küldjünk!

freebsd-ia64 (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia64>)

A FreeBSD portolása I64-re

Ez a levelezési lista a FreeBSD az Intel IA-64 platformjára készített portjával foglalkozó egyének kommunikációs eszköze, ahol az ezzel kapcsolatos problémák és azok különböző megoldásai kerülnek terítékre. A téma iránt érdeklődőket is szívesen látjuk.

freebsd-isdn (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn>)

ISDN kommunikáció

Ez a levelezési lista a FreeBSD ISDN támogatásáról szól.

freebsd-java (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-java>)

Java alapú fejlesztések

A levelezési listán a nagyobb Java alkalmazások FreeBSD alapú fejlesztését, valamint a JDK-k portolásáról és karbantartását beszélik meg.

freebsd-jobs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-jobs>)

Munkát keres/kínál

Erre a fórumra tudjuk beküldeni a kifejezetten FreeBSD-hez kapcsolódó munkaajánlatokat és önéletrajzokat, tehát ez a megfelelő hely, ha FreeBSD-s munkát keresünk, vagy éppen FreeBSD szakértőket. Ez azonban *nem* egy általános célú állásbörze, mert arra megvannak a megfelelő helyek.

Szeretnénk hozzátenni, hogy ez a lista, a többi FreeBSD.org levelezési listához hasonlóan, világméretben működik. Ezért ne felejtjük sosem pontosan megjelölni a munkavégzés helyét, illetve hogy milyen kommunikációs és esetlegesen költözési lehetőségeket javasolunk.

A leveleket csak nyílt formátumban küldjük — elsősorban szöveges formátumban, de az egyszerűbb PDF, HTML vagy még néhány más hozzájuk hasonló formátumot is alkalmazhatunk. Az olyan zárt formátumok, mint például a Microsoft Word (.doc) azonban nem fognak továbbítódni.

freebsd-kde (<https://mail.kde.org/mailman/listinfo/kde-freebsd>)

KDE

A **KDE** és FreeBSD kapcsolatáról szóló lista. Szigorúan szakmai jellegű, ezért csak ilyen témájú levelek küldése elfogadott.

freebsd-hackers (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>)

Szakmai kérdések

Ez a FreeBSD szakmai jellegű kérdéseivel foglalkozó fórum. Ez az első számú szakmai levelezési lista. A FreeBSD fejlesztésével aktívan foglalkozó egyének számára ajánljuk, hiszen itt vethetik fel problémáikat, itt kereshetnek rájuk megoldásokat. Az ilyen típusú megbeszéléseket figyelemmel követő egyéneket is szívesen fogadjuk. Mivel ez egy erősen szakmai jellegű lista, ezért csak ilyen témájú leveleket várunk ide.

freebsd-hardware (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hardware>)

A FreeBSD és a hardverek kapcsolatáról általában

Ezen a listán kerül megvitatásra minden olyan hardver, amelyen a FreeBSD működik: milyen gondok adódhatnak, milyen hardvereket érdemes beszerezniük vagy elkerülniük.

freebsd-hubs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hubs>)

Tükrözések

A FreeBSD tükrözéseit karbantartó egyének számára fontos bejelentések és megbeszélések.

freebsd-isp (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp>)

Az internet-szolgáltatók fóruma

Ezen a levelezési listán a FreeBSD-t használó internet-szolgáltatók tehetik fel kérdéseiket. Szigorúan csak szakmai jellegű kérdések engedélyezettek.

freebsd-mono (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-mono>)

Mono és C# alkalmazások FreeBSD alatt

Ezen a levelezési listán a Mono fejlesztői keretrendszer FreeBSD alatt futó változatával kapcsolatos megbeszélések folynak. Ez egy szakmai jellegű lista. Itt a Mono vagy más C# alkalmazások FreeBSD változatának elkészítésén dolgozó egyének tudnak problémákat felvetni vagy megvitatni a különböző megoldásokat. Rajtuk kívül viszont szeretettel várunk minden érdeklődőt a téma iránt.

freebsd-openoffice (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-openoffice>)

OpenOffice.org

Az **OpenOffice.org** és **StarOffice** portolásával és karbantartásával kapcsolatos megbeszélések.

freebsd-performance (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-performance>)

A FreeBSD hangolásának és gyorsításának tárgyalása

Ezen a levelezési listán van lehetőségük a hackereknek, rendszergazdáknak és/vagy az érintett feleknek a FreeBSD teljesítményével kapcsolatos témákban kifejezni a véleményüket. Leginkább nagy terhelés alatt levő, vagy teljesítménybeli problémákkal küszködő, esetleg még többet tudó FreeBSD rendszerek tárgyalása a cél. Lehetőleg az érintett gyártókkal és szállítókkal együttesen próbáljuk kidolgozni a FreeBSD teljesítményének növelésére tett kísérleteinket, ezért őket is szívesen látjuk ezen a listán. Ez a kifejezetten szakmai jellegű lista többségében a tapasztalt FreeBSD felhasználók, hackerek vagy rendszergazdák számára tárja fel a gyors, megbízható és skálázható FreeBSD rendszerek lehetőségeit. Ez alapvetően nem egy kérdeztetős lista, ahol a dokumentációk elolvasását tudjuk megspórolni, hanem egy olyan hely, ahol a teljesítményt érintő megválaszolatlan kérdések és előremutató fejlesztések nyernek teret.

freebsd-pf (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf>)

A csomagszűrő tűzfalrendszerrel kapcsolatos kérdések

A FreeBSD csomagszűrőjéhez (packet filter, pf) tartozó tűzfalrendszer megbeszéléseit összefoglaló lista. Szakmai jellegű fejtegetések és felhasználói kérdések egyaránt jöhetnek. Továbbá ezen a listán foglalkozunk az ALTQ rendszer működésével is.

freebsd-platforms (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-platforms>)

Portolás nem Intel platformokra

A FreeBSD különböző, nem az Intel architektúrára építkező portjainak indítványozása és általános jellegű megvitatása. Ez egy kiemelten szakmai jellegű lista, ezért ide csak ilyen témájú leveleket várunk.

freebsd-policy (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-policy>)

Az Core Team szabályozásai

Alacsony forgalmú, csak olvasható lista, ahol a FreeBSD fejlesztését irányító csoport különböző döntéseiről olvashatunk.

freebsd-ports (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports>)

A "portok" megbeszélése

A FreeBSD "portgyűjteményével" (`/usr/ports`), a portok infrastruktúrájával és a portok fejlesztésének irányításával kapcsolatos megbeszélések. Erősen szakmai jellegű lista, ezért ide csak ilyen témában írjunk.

freebsd-ports-bugs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs>)

A "portok" hibáinak tárgyalása

A FreeBSD "portgyűjteményének" (`/usr/ports`), a bejelentett portok és azok módosításához kötődő hibajelentésekkel foglalkozó lista. Ez egy szakmai jellegű lista, ahol csak ilyen jellegű témákra számítunk.

freebsd-proliant (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-proliant>)

A FreeBSD és a HP ProLiant szerverek kapcsolatát érintő szakmai megbeszélések

Ezen a levelezési listán a FreeBSD HP ProLiant szervereken történő használatát célozzuk meg, beleértve a ProLianthoz tartozó eszközmeghajtókat, karbantartó és konfigurációs szoftvereket és BIOS-frissítéseket. Ennek megfelelően tehát a hpsamd, hpsamcli és hpacucli modulok is elsősorban itt kerülnek felboncolásra.

freebsd-python (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-python>)

A FreeBSD és a Python

A lista a FreeBSD Python támogatásának fejlesztéséről folytatott szakmai megbeszéléseket foglalja össze. Elsősorban a Python portolásával foglalkozó egyének, valamint a külső fejlesztők által készített modulok és a **Zope** FreeBSD-s alkalmazásával foglalkozik. Az említett témák iránti érdeklődőket is szeretettel várjuk.

freebsd-questions (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>)

Felhasználói kérdések

Ez a levelezési lista a FreeBSD-vel kapcsolatos kérdésekről szól. Lehetőleg ne küldjünk “hogyan” témájú kérdéseket erre a szakmai listára, hacsak nem kifejezetten szakmai jellegűnek szánjuk.

freebsd-ruby (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ruby>)

A Ruby használata FreeBSD rendszereken

Ezen a listán a FreeBSD Ruby támogatásával foglalkozunk, témáját tekintve teljesen szakmai jellegű. Elsősorban a Ruby portokon, külső Ruby könyvtárakon és rendszereken dolgozó fejlesztők figyelmébe ajánljuk. Mindenkit szeretettel várunk, aki ezekkel kapcsolatos szakmai tárgyú témákat szeretne megvitatni.

freebsd-scsi (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-scsi>)

A SCSI alrendszer

Ezt a levelezési listát a FreeBSD alatt a SCSI alrendszerrel foglalkozók számára tarjuk fenn. Mivel ez egy erősen szakmai jellegű lista, ezért rajta csak szakmai témák megengedettek.

freebsd-security (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security>)

Biztonsági problémák

A FreeBSD biztonságát illető kérdések (DES, Kerberos, biztonsági rések és javításaik, stb.) Szakmai jellegű lista, ezért ide csak a témához szorosan kapcsolódó leveleket szabad beküldeni. Alapvetően nem kérdezz-felelek típusú a lista működése, habár a GYIK-hoz minden hozzájárulást (kérdést ÉS választ EGYARÁNT) szívesen veszünk.

freebsd-security-notifications (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications>)

Biztonsági figyelmeztetések

A FreeBSD-t érintő biztonsági problémákról és javításaikról szóló értesítések. Megbeszélésekkel, vitákkal nem foglalkozik, mivel azok a FreeBSD-security listán folynak.

freebsd-small (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-small>)

A FreeBSD használata beágyazott alkalmazásokban

A szokatlanul kis méretű vagy beágyazott FreeBSD rendszerekhez kapcsolódó megbeszélések színhelye. Szakmai jellegű lista, ezért szigorúan csak a témához tartozó leveleket fogad.

Megjegyzés: Ezt a listát időközben felváltotta a freebsd-embedded (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded>) lista.

freebsd-stable (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable>)

A FreeBSD-STABLE használatáról szóló lista

Ez a FreeBSD-STABLE használoinak levelezési listája. Ide kerülnek beküldésre a -STABLE ágot futtató felhasználókat érintő friss változások, valamint hozzájuk kötődően a -STABLE használatához szükséges elvégzendő lépések. Aki a "STABLE" jelzésű változatot használja, mindenképpen iratkozzon fel rá. Szigorúan szakmai jellegű lista, ezért csak szakmai témájú leveleket vár.

freebsd-standards (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-standards>)

C99 és POSIX megfelelés

Ez a fórum foglalkozik a FreeBSD és a C99, valamint a POSIX szabványok szerinti megfelelésével.

freebsd-toolchain (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-toolchain>)

A FreeBSD alapvető segédprogramjainak karbantartása

Ezen a listán a FreeBSD egyes kiadásaihoz mellékelt alapvető segédprogramokkal kapcsolatos témákat találjuk meg. Ilyen többek közt rendszerben használt Clang és a GCC fordítók aktuálisan használt változatai, de emellett még szó eshet a rendszerhez kapcsolódó különféle assemblerek, linkerek és debuggerek állapotáról.

freebsd-usb (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-usb>)

A FreeBSD USB támogatása

Ez a levelezési lista fogja összes a FreeBSD USB támogatásával foglalkozó szakmai témákat.

freebsd-user-groups (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-user-groups>)

A felhasználói csoportokat irányító lista

Ez a levelezési lista az egyes területeken működő felhasználói csoportok az irányítást végző központi csoport tagjai általi összehangolásához tartozó problémák megbeszélésére való. Ez a lista leginkább a gyűlések letisztázására és a több csoporton átívelő nagyobb projektek szervezéséhez használatos.

freebsd-vendors (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-vendors>)

Gyártók

A FreeBSD projekt és a hozzá kötődő hardver- és szoftvergyártók együttműködését elősegítő lista.

freebsd-virtualization (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-virtualization>)

A FreeBSD részéről támogatott különböző virtualizációs technológiák

Ezen a levelezési listán elsősorban a FreeBSD által támogatott virtualizációs megoldásokat vitatjuk meg. Ennek keretében egyrészt az ehhez kapcsolódó alapvető funkciók megvalósítása valamint további újítások kerülnek a középpontba, másrészt a felhasználók számára ezzel létrehoztunk egy fórumot a felmerülő problémák megoldására és az alkalmazási lehetőségek megbeszélésére.

freebsd-wip-status (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-wip-status>)

A FreeBSD-vel kapcsolatos folyamatban levő fejlesztések helyzetjelentése

Ezen a levelezési listán kerülnek bejelentésre a FreeBSD továbbfejlesztéséhez fűződő különböző munkák és azok haladásának menete. Az ide befutó üzeneteket moderálják. Javasoljuk, hogy elsődlegesen az adott témához tartozó tematikus FreeBSD listára küldjük a bejelentésünket és csak egy másolatot erre a listára. Ennek köszönhetően a munkánk az adott témaspecifikus listán rögtön meg is vitatható, mivel ezen a listán semmi ilyen nem engedélyezett.

A lista archívumába tekintve tájékozódhatunk arról, hogy pontosan milyen formai követelmények illene megfelelnie a beküldendő üzenetünknek.

A listára beérkező üzenetekből egy szerkesztett válogatás jelenik meg néhány havonta a FreeBSD honlapján a Projekt helyzetjelentésének részeként ¹. A korábban beküldött jelentések mellett itt még találhatunk további példákat.

freebsd-xen (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-xen>)

A Xen FreeBSD portjának (implementáció és használat) megvitatása

A lista elsősorban a Xen FreeBSD-re készült változatával foglalkozik. Előreláthatólag elég kevesen fognak írni erre a listára ahhoz, hogy helyet kapjanak rajta az implementációt és a kialakítást érintő szakmai jellegű megbeszélések és a telepítéssel kapcsolatos kérdések egyaránt.

C.1.4. A levelezési listák szűrése

A kéretlen reklámlevelek, vírusok és egyébek elleni védekezés céljából a FreeBSD levelezési listáinak forgalmát több módon is szűrik. Az ebben a szakaszban bemutatott szűrési megoldások nem fedik le a levelezési listák védelmében alkalmazott összes lehetőséget.

A levelezési listákra csak bizonyos típusú csatolt állományokat küldhetünk be. Az alábbi listában nem található MIME típusú csatolt objektumokat még a listára érkezés előtt törlik.

- application/octet-stream
- application/pdf
- application/pgp-signature
- application/x-pkcs7-signature
- message/rfc822
- multipart/alternative
- multipart/related
- multipart/signed
- text/html
- text/plain
- text/x-diff
- text/x-patch

Megjegyzés: Egyes levelezési listák ugyan megengedhetnek további csatolt MIME objektumokat is, habár a legtöbb lista esetében a fenti lista a mérvadó.

Ha egy levélben a szöveg HTML és nyers szöveg formátumban is szerepel, a HTML változat automatikusan eltávolításra kerül. Ha az e-mail csak HTML formában tartalmazza a szöveget, akkor automatikusan nyers szövegre alakítódik át.

C.2. Usenet hírcsoportok

A két FreeBSD-s hírcsoport mellett még akadnak olyan további csoportok is, ahol FreeBSD témájú kérdéseket vitathatunk meg vagy hasznos lehet számunkra. Az itt felsorolt hírcsoportok kulcsszavakkal kereshető archívuma (http://minnie.tuhs.org/BSD-info/bsdnews_search.html) Warren Toomey tulajdona (<wkt@cs.adfa.edu.au>).

C.2.1. BSD-s hírcsoportok

- comp.unix.bsd.freebsd.announce (news:comp.unix.bsd.freebsd.announce)
- comp.unix.bsd.freebsd.misc (news:comp.unix.bsd.freebsd.misc)
- de.comp.os.unix.bsd (news:de.comp.os.unix.bsd) (német)
- fr.comp.os.bsd (news:fr.comp.os.bsd) (francia)
- it.comp.os.freebsd (news:it.comp.os.freebsd) (olasz)
- tw.bbs.comp.386bsd (news:tw.bbs.comp.386bsd) (hagyományos kínai)

C.2.2. Egyéb érdekes UNIX-os hírcsoportok

- comp.unix (news:comp.unix)
- comp.unix.questions (news:comp.unix.questions)
- comp.unix.admin (news:comp.unix.admin)
- comp.unix.programmer (news:comp.unix.programmer)
- comp.unix.shell (news:comp.unix.shell)
- comp.unix.user-friendly (news:comp.unix.user-friendly)
- comp.security.unix (news:comp.security.unix)
- comp.sources.unix (news:comp.sources.unix)
- comp.unix.advocacy (news:comp.unix.advocacy)
- comp.unix.misc (news:comp.unix.misc)
- comp.bugs.4bsd (news:comp.bugs.4bsd)
- comp.bugs.4bsd.ucb-fixes (news:comp.bugs.4bsd.ucb-fixes)
- comp.unix.bsd (news:comp.unix.bsd)

C.2.3. X Window System

- `comp.windows.x.i386unix` (`news:comp.windows.x.i386unix`)
- `comp.windows.x` (`news:comp.windows.x`)
- `comp.windows.x.apps` (`news:comp.windows.x.apps`)
- `comp.windows.x.announce` (`news:comp.windows.x.announce`)
- `comp.windows.x.intrinsics` (`news:comp.windows.x.intrinsics`)
- `comp.windows.x.motif` (`news:comp.windows.x.motif`)
- `comp.windows.x.pex` (`news:comp.windows.x.pex`)
- `comp.emulators.ms-windows.wine` (`news:comp.emulators.ms-windows.wine`)

C.3. Világhálós szolgáltatások

C.3.1. Fórumok, blogok és ismertségi hálózatok

- A FreeBSD fórumok (<http://forums.freebsd.org/>) a FreeBSD kapcsán felmerülő kérdések és szakmai témák megvitatásához egy webes felületet kínálnak fel.
- A Planet FreeBSD (<http://planet.freebsd.org/>) honlapján fejlesztők által vezetett tucatnyi webes naplót és hozzájuk tartozó RSS feedeket találhatunk. Sok fejlesztő ezen a módon készít rövid feljegyzéseket a jelenlegi munkájáról, az új javításokról és más egyéb terveiről.
- A Youtube-on keresztül elérhető BSDConferences (<http://www.youtube.com/bsdconferences>) csatornán a világ minden táján tartott különböző BSD témájú konferenciák videoanyagait találhatjuk meg. Segítségével megtekinthetjük a fontosabb fejlesztők által a saját munkájukról tartott különböző előadásokat.

C.3.2. Hivatalos tükrözések

Központi szerverek, Amerikai Egyesült Államok, Ausztrália, Ausztria, Cseh Köztársaság, Dánia, Dél-Afrika, Egyesült Királyság, Finnország, Franciaország, Hollandia, Hongkong, Írország, Izland, Japán, Kanada, Lettország, Litvánia, Németország, Norvégia, Örményország, Oroszország, Spanyolország, Svájc, Svédország, Szlovák Köztársaság, Szlovénia, Tajvan.

(Dátum: UTC)

•

Központi szerverek

- <http://www.FreeBSD.org/>

•

Amerikai Egyesült Államok

- <http://www5.us.FreeBSD.org/> (IPv6)

•

Ausztrália

- <http://www.au.FreeBSD.org/>
- <http://www2.au.FreeBSD.org/>

•

Ausztria

- <http://www.at.FreeBSD.org/> (IPv6)

•

Cseh Köztársaság

- <http://www.cz.FreeBSD.org/> (IPv6)

•

Dánia

- <http://www.dk.FreeBSD.org/> (IPv6)

•

Dél-Afrika

- <http://www.za.FreeBSD.org/>
- <http://www2.za.FreeBSD.org/>

•

Egyesült Királyság

- <http://www1.uk.FreeBSD.org/>
- <http://www3.uk.FreeBSD.org/>

•

Finnország

- <http://www.fi.FreeBSD.org/>

•

Franciaország

- <http://www1.fr.FreeBSD.org/>
- Hollandia
 - <http://www.nl.FreeBSD.org/>
- Hongkong
 - <http://www.hk.FreeBSD.org/>
- Írország
 - <http://www.ie.FreeBSD.org/>
- Izland
 - <http://www.is.FreeBSD.org/>
- Japán
 - <http://www.jp.FreeBSD.org/www.FreeBSD.org/> (IPv6)
- Kanada
 - <http://www.ca.FreeBSD.org/>
 - <http://www2.ca.FreeBSD.org/>
- Lettország
 - <http://www.lv.FreeBSD.org/>
- Litvánia
 - <http://www.lt.FreeBSD.org/>

- Németország
 - <http://www.de.FreeBSD.org/>
- Norvégia
 - <http://www.no.FreeBSD.org/>
- Örményország
 - <http://www1.am.FreeBSD.org/> (IPv6)
- Oroszország
 - <http://www.ru.FreeBSD.org/>
 - <http://www2.ru.FreeBSD.org/>
- Spanyolország
 - <http://www.es.FreeBSD.org/>
 - <http://www2.es.FreeBSD.org/>
- Svájc
 - <http://www.ch.FreeBSD.org/> (IPv6)
 - <http://www2.ch.FreeBSD.org/> (IPv6)
- Svédország
 - <http://www.se.FreeBSD.org/>
 - <http://www2.se.FreeBSD.org/>
- Szlovák Köztársaság
 - <http://www.sk.FreeBSD.org/>

•

Szlovénia

- <http://www.si.FreeBSD.org/>

•

Tajvan

- <http://www.tw.FreeBSD.org/> (IPv6)
- <http://www2.tw.FreeBSD.org/>
- <http://www4.tw.FreeBSD.org/>
- <http://www5.tw.FreeBSD.org/> (IPv6)

C.4. E-mail címek

A következő felhasználói csoportok nyújtanak FreeBSD-s e-mail címeket tagjaiknak. A rendszergazdák bármilyen visszaélés esetén fenntartják a visszavonás jogát.

Címtartomány	Lehetőségek	Felhasználói csoport	Rendszergazda
ukug.uk.FreeBSD.org	Csak továbbítás	<ukfreebsd@uk.FreeBSD.org>	Lee Johnston <lee@uk.FreeBSD.org>

Megjegyzések

1. <http://www.freebsd.org/news/status/>

Függelék D. PGP-kulcsok

Arra az esetre, ha ellenőrizniünk kellene a tisztségviselők vagy fejlesztők aláírását, esetleg titkosított e-mailt szeretnénk küldeni nekik, ebben a fejezetben mellékeljük az ehhez szükséges kulcsokat. A FreeBSD.org összes felhasználójának kulcsa letölthető a <http://www.FreeBSD.org/doc/pgpkeyring.txt> címről.

D.1. Tisztségviselők

D.1.1. Security Officer Team <security-officer@FreeBSD.org>

```
pub 1024D/CA6CDFB2 2002-08-27 FreeBSD Security Officer <security-officer@FreeBSD.org>
    Key fingerprint = C374 0FC5 69A6 FBB1 4AED B131 15D6 8804 CA6C DFB2
sub 2048g/A3071809 2002-08-27
```

D.1.2. Core Team Titkár <core-secretary@FreeBSD.org>

```
pub 2048R/2CA49776 2012-07-23
    Key fingerprint = 89F6 C031 B4E3 D472 E4CE 8372 4D58 FDCD 2CA4 9776
uid FreeBSD Core Team Secretary <core-secretary@freebsd.org>
sub 2048R/BBAD1C98 2012-07-23
```

D.1.3. Ports Management Team Titkár <portmgr-secretary@FreeBSD.org>

```
pub 2048R/BBC4D7D5 2012-07-24
    Key fingerprint = FB37 45C8 6F15 E8ED AC81 32FC D829 4EC3 BBC4 D7D5
uid FreeBSD Ports Management Team Secretary <portmgr-secretary@FreeBSD.org>
sub 2048R/5F65CFE7 2012-07-24
```

D.2. A Core Team tagjai

D.2.1. Thomas Abthorpe <tabthorpe@FreeBSD.org>

```
pub 2048R/A473C990 2010-05-28
    Key fingerprint = D883 2D7C EB78 944A 69FC 36A6 D937 1097 A473 C990
uid Thomas Abthorpe (FreeBSD Committer) <tabthorpe@FreeBSD.org>
uid Thomas Abthorpe <tabthorpe@abthorpe.org>
uid Thomas Abthorpe <tabthorpe@goodking.ca>
uid Thomas Abthorpe <tabthorpe@goodking.org>
uid Thomas Abthorpe <thomas@goodking.ca>
sub 2048R/8CA60EE0 2010-05-28
```

D.2.2. Gavin Atkinson <gavin@FreeBSD.org>

```

pub 1024D/A093262B 2005-02-18
    Key fingerprint = 313A A79F 697D 3A5C 216A EDF5 935D EF44 A093 262B
uid          Gavin Atkinson (FreeBSD key) <gavin@FreeBSD.org>
uid          Gavin Atkinson (Work e-mail) <ga9@york.ac.uk>
uid          Gavin Atkinson <gavin@16squared.co.uk>
uid          Gavin Atkinson <gavin.atkinson@ury.york.ac.uk>
uid          Gavin Atkinson (Work e-mail) <gavin.atkinson@york.ac.uk>
sub 2048g/58F40B3D 2005-02-18

```

D.2.3. John Baldwin <jhb@FreeBSD.org>

```

pub 1024R/C10A874D 1999-01-13 John Baldwin <jbaldwin@weather.com>
    Key fingerprint = 43 33 1D 37 72 B1 EF 5B 9B 5F 39 F8 BD C1 7C B5
uid          John Baldwin <john@baldwin.cx>
uid          John Baldwin <jhb@FreeBSD.org>
uid          John Baldwin <jobaldwi@vt.edu>

```

D.2.4. Konstantin Belousov <kib@FreeBSD.org>

```

pub 4096R/C1BCAD41 2012-11-17
    Key fingerprint = 7DE0 3388 64AC 53C3 7B88 3A79 90C2 B92B C1BC AD41
uid          Konstantin Belousov <kib@FreeBSD.org>
uid          Konstantin Belousov <kostikbel@gmail.com>
uid          Konstantin Belousov <kib@kib.kiev.ua>
sub 4096R/3BBC8F64 2012-11-17

```

D.2.5. David Chisnall <theraven@FreeBSD.org>

```

pub 4096R/65C4F55D 2012-11-28
    Key fingerprint = 3E8F 5E9F 7586 F090 AC2C 58C2 BA06 FF14 65C4 F55D
uid          David Chisnall <theraven@FreeBSD.org>
sub 4096R/04B2A21D 2012-11-28

```

D.2.6. Hiroki Sato <hrs@FreeBSD.org>

```

pub 1024D/2793CF2D 2001-06-12
    Key fingerprint = BDB3 443F A5DD B3D0 A530 FFD7 4F2C D3D8 2793 CF2D
uid          Hiroki Sato <hrs@allbsd.org>
uid          Hiroki Sato <hrs@eos.ocn.ne.jp>
uid          Hiroki Sato <hrs@ring.gr.jp>
uid          Hiroki Sato <hrs@FreeBSD.org>
uid          Hiroki Sato <hrs@jp.FreeBSD.org>
uid          Hiroki Sato <hrs@vlsi.ee.noda.tus.ac.jp>
uid          Hiroki Sato <hrs@jp.NetBSD.org>

```

```
uid      Hiroki Sato <hrs@NetBSD.org>
uid      Hiroki Sato <hrs@ec.ss.titech.ac.jp>
uid      Hiroki Sato <hrs@ieee.org>
uid      Hiroki Sato <hrs@acm.org>
uid      Hiroki Sato <hrs@bsdconsulting.co.jp>
uid      Hiroki Sato <hrs@bsdresearch.org>
uid      Hiroki Sato <hrs@ec.ce.titech.ac.jp>
sub      1024g/8CD251FF 2001-06-12
```

D.2.7. Peter Wemm <peter@FreeBSD.org>

```
pub      1024D/7277717F 2003-12-14 Peter Wemm <peter@wemm.org>
        Key fingerprint = 622B 2282 E92B 3BAB 57D1 A417 1512 AE52 7277 717F
uid      Peter Wemm <peter@FreeBSD.ORG>
sub      1024g/8B40D9D1 2003-12-14
pub      1024R/D89CE319 1995-04-02 Peter Wemm <peter@netplex.com.au>
        Key fingerprint = 47 05 04 CA 4C EE F8 93 F6 DB 02 92 6D F5 58 8A
uid      Peter Wemm <peter@perth.dialix.oz.au>
uid      Peter Wemm <peter@haywire.dialix.com>
```

D.2.8. Martin Wilke <miwi@FreeBSD.org>

```
pub      1024D/B1E6FCE9 2009-01-31
        Key fingerprint = C022 7D60 F598 8188 2635 0F6E 74B2 4884 B1E6 FCE9
uid      Martin Wilke <miwi@FreeBSD.org>
sub      4096g/096DA69D 2009-01-31
```

D.3. Fejlesztők

D.3.1. Ariff Abdullah <ariff@FreeBSD.org>

```
pub      1024D/C5304CDA 2005-10-01
        Key fingerprint = 5C7C 6BF4 8293 DE76 27D9 FD57 96BF 9D78 C530 4CDA
uid      Ariff Abdullah <skywizard@MyBSD.org.my>
uid      Ariff Abdullah <ariff@MyBSD.org.my>
uid      Ariff Abdullah <ariff@FreeBSD.org>
sub      2048g/8958C1D3 2005-10-01
```

D.3.2. Thomas Abthorpe <tabthorpe@FreeBSD.org>

```
pub      2048R/A473C990 2010-05-28
        Key fingerprint = D883 2D7C EB78 944A 69FC 36A6 D937 1097 A473 C990
uid      Thomas Abthorpe (FreeBSD Committer) <tabthorpe@FreeBSD.org>
uid      Thomas Abthorpe <tabthorpe@abthorpe.org>
```

```
uid          Thomas Abthorpe <tabthorpe@goodking.ca>
uid          Thomas Abthorpe <tabthorpe@goodking.org>
uid          Thomas Abthorpe <thomas@goodking.ca>
sub 2048R/8CA60EE0 2010-05-28
```

D.3.3. Eitan Adler <eadler@FreeBSD.org>

```
pub 4096R/8FC8196C 2011-02-11
   Key fingerprint = 49C7 29DF E09C 0FC7 A1C4 6ECB A338 A6FC 8FC8 196C
uid          Eitan Adler <lists@eitanadler.com>
sub 4096R/18763D51 2011-02-11
sub 4096R/DAB9CF9B 2011-02-11
```

D.3.4. Shaun Amott <shaun@FreeBSD.org>

```
pub 1024D/6B387A9A 2001-03-19
   Key fingerprint = B506 E6C7 74A1 CC11 9A23 5C13 9268 5D08 6B38 7A9A
uid          Shaun Amott <shaun@inerd.com>
uid          Shaun Amott <shaun@FreeBSD.org>
sub 2048g/26FA8703 2001-03-19
sub 2048R/7FFF5151 2005-11-06
sub 2048R/27C54137 2005-11-06
```

D.3.5. Henrik Brix Andersen <brix@FreeBSD.org>

```
pub 1024D/54E278F8 2003-04-09
   Key fingerprint = 7B63 EF32 7831 A704 220D 7E61 BFE4 387E 54E2 78F8
uid          Henrik Brix Andersen <henrik@brixandersen.dk>
uid          Henrik Brix Andersen <brix@FreeBSD.org>
uid          Henrik Brix Andersen <hbn@terma.com>
uid          Henrik Brix Andersen <brix@osaa.dk>
sub 1024g/3B13C209 2003-04-09
```

D.3.6. Matthias Andree <mandree@FreeBSD.org>

```
pub 1024D/052E7D95 2003-08-28
   Key fingerprint = FDD0 0C43 6E33 07E1 0758 C6A8 BE61 8339 052E 7D95
uid          Matthias Andree <mandree@freebsd.org>
uid          Matthias Andree <matthias.andree@gmx.de>
sub 1536g/E65A83DA 2003-08-28
```

D.3.7. Will Andrews <will@FreeBSD.org>

```

pub 1024D/F81672C5 2000-05-22 Will Andrews (Key for official matters) <will@FreeBSD.org>
    Key fingerprint = 661F BBF7 9F5D 3D02 C862 5F6C 178E E274 F816 72C5
uid                               Will Andrews <will@physics.purdue.edu>
uid                               Will Andrews <will@puck.firepipe.net>
uid                               Will Andrews <will@c-60.org>
uid                               Will Andrews <will@csociety.org>
uid                               Will Andrews <will@csociety.ecn.purdue.edu>
uid                               Will Andrews <will@telperion.openpackages.org>
sub 1024g/55472804 2000-05-22

```

D.3.8. Dmitry Andric <dim@FreeBSD.org>

```

pub 1024D/2E2096A3 1997-11-17
    Key fingerprint = 7AB4 62D2 CE35 FC6D 4239 4FCD B05E A30A 2E20 96A3
uid                               Dmitry Andric <dimitry@andric.com>
uid                               Dmitry Andric <dim@xs4all.nl>
uid                               Dmitry Andric <dimitry.andric@tomtom.com>
uid                               [jpeg image of size 5132]
uid                               Dmitry Andric <dim@nah6.com>
uid                               Dmitry Andric <dim@FreeBSD.org>
sub 4096g/6852A5C5 1997-11-17

```

D.3.9. Eric Anholt <anholt@FreeBSD.org>

```

pub 1024D/6CF0EAF7 2003-09-08
    Key fingerprint = 76FE 2475 820B B75F DCA4 0F3E 1D47 6F60 6CF0 EAF7
uid                               Eric Anholt <eta@lclark.edu>
uid                               Eric Anholt <anholt@FreeBSD.org>
sub 1024g/80B404C1 2003-09-08

```

D.3.10. Marcus von Appen <mva@FreeBSD.org>

```

pub 1024D/B267A647 2009-02-14
    Key fingerprint = C7CC 1853 D8C5 E580 7795 B654 8BAF 3F12 B267 A647
uid                               Marcus von Appen <freebsd@sysfault.org>
uid                               Marcus von Appen <mva@freebsd.org>
sub 2048g/D34A3BAF 2009-02-14

```

D.3.11. Marcelo Araujo <araujo@FreeBSD.org>

```

pub 1024D/53E4CFA8 2007-04-27
    Key fingerprint = 9D6A 2339 925C 4F61 ED88 ED8B A2FC 4977 53E4 CFA8
uid                               Marcelo Araujo (Ports Committer) <araujo@FreeBSD.org>
sub 2048g/63CC012D 2007-04-27

```

D.3.12. Mathieu Arnold <mat@FreeBSD.org>

```

pub 1024D/FE6D850F 2005-04-25
    Key fingerprint = 2771 11F4 0A7E 73F9 ADDD A542 26A4 7C6A FE6D 850F
uid      Mathieu Arnold <mat@FreeBSD.org>
uid      Mathieu Arnold <mat@mat.cc>
uid      Mathieu Arnold <mat@cpan.org>
uid      Mathieu Arnold <m@absolight.fr>
uid      Mathieu Arnold <m@absolight.net>
uid      Mathieu Arnold <mat@club-internet.fr>
uid      Mathieu Arnold <marnold@april.org>
uid      Mathieu Arnold <paypal@mat.cc>
sub 2048g/EAD18BD9 2005-04-25

```

D.3.13. Takuya ASADA <syuu@FreeBSD.org>

```

pub 2048R/43788F78 2012-11-21
    Key fingerprint = 31CE 242E 6F4F F24F EE4 D9BB 0890 2C5F 4378 8F78
uid      Takuya ASADA <syuu@freebsd.org>
sub 2048R/A87B0906 2012-11-21

```

D.3.14. Satoshi Asami <asami@FreeBSD.org>

```

pub 1024R/1E08D889 1997-07-23 Satoshi Asami <asami@cs.berkeley.edu>
    Key fingerprint = EB 3C 68 9E FB 6C EB 3F DB 2E 0F 10 8F CE 79 CA
uid      Satoshi Asami <asami@FreeBSD.ORG>

```

D.3.15. Gavin Atkinson <gavin@FreeBSD.org>

```

pub 1024D/A093262B 2005-02-18
    Key fingerprint = 313A A79F 697D 3A5C 216A EDF5 935D EF44 A093 262B
uid      Gavin Atkinson (FreeBSD key) <gavin@FreeBSD.org>
uid      Gavin Atkinson (Work e-mail) <ga9@york.ac.uk>
uid      Gavin Atkinson <gavin@16squared.co.uk>
uid      Gavin Atkinson <gavin.atkinson@ury.york.ac.uk>
uid      Gavin Atkinson (Work e-mail) <gavin.atkinson@york.ac.uk>
sub 2048g/58F40B3D 2005-02-18

```

D.3.16. Joseph S. Atkinson <jsa@FreeBSD.org>

```

pub 2048R/21AA7B06 2010-07-14
    Key fingerprint = 5B38 63B0 9CCA 12BE 3919 9412 CC9D FC84 21AA 7B06
uid      Joseph S. Atkinson <jsa@FreeBSD.org>
uid      Joseph S. Atkinson <jsa.bsd@gmail.com>
uid      Joseph S. Atkinson <jsa@wickedmachine.net>
sub 2048R/5601C3E3 2010-07-14

```

D.3.17. Philippe Audeoud <jadawin@FreeBSD.org>

```
pub 1024D/C835D40E 2005-04-13
    Key fingerprint = D090 8C96 3612 15C9 4E3E 7A4A E498 FC2B C835 D40E
uid      Philippe Audeoud <jadawin@tuxaco.net>
uid      Philippe Audeoud <philippe@tuxaco.net>
uid      Philippe Audeoud <philippe.audeoud@sitadelle.com>
uid      Philippe Audeoud <jadawin@freebsd.org>
sub 2048g/EF8EA329 2005-04-13
```

D.3.18. Timur I. Bakeyev <timur@FreeBSD.org>

```
pub 1024D/60BA1F47 2002-04-27
    Key fingerprint = 84BF EAD1 607D 362F 210E 69B3 0BF0 6412 60BA 1F47
uid      Timur I. Bakeyev (BaT) <timur@bat.ru>
uid      Timur I. Bakeyev <timur@gnu.org>
uid      Timur I. Bakeyev (BaT) <bat@cpan.org>
uid      Timur I. Bakeyev (BaT) <timur@FreeBSD.org>
uid      Timur I. Bakeyev (BaT) <timur@gnome.org>
uid      Timur I. Bakeyev <timur@gnome.org>
sub 2048g/8A5B0042 2002-04-27
```

D.3.19. Glen Barber <gjb@FreeBSD.org>

```
pub 2048R/A0B946A3 2010-08-03 [expires: 2017-04-25]
    Key fingerprint = 78B3 42BA 26C7 B2AC 681E A7BE 524F 0C37 A0B9 46A3
uid      Glen Barber <gjb@FreeBSD.org>
uid      Glen Barber <glen.j.barber@gmail.com>
uid      Glen Barber <gjb@glenbarber.us>
sub 2048R/6C0527E5 2010-08-03
```

D.3.20. Nick Barkas <snb@FreeBSD.org>

```
pub 2048R/DDADB9DC 2010-07-27
    Key fingerprint = B678 6ECB 303D F580 A050 098F BDFF 4F3D DDAD B9DC
uid      S. Nicholas Barkas <snb@freebsd.org>
sub 2048R/36E181FB 2010-07-27
sub 2048R/BDA4BED3 2010-07-29
sub 2048R/782A8737 2010-07-29
```

D.3.21. Simon Barner <barner@FreeBSD.org>

```
pub 1024D/EBADA82A 2000-11-10
    Key fingerprint = 67D1 3562 9A2F 3177 E46A 35ED 0A49 FEFD EBAD A82A
uid      Simon Barner <barner@FreeBSD.org>
uid      Simon Barner <barner@in.tum.de>
```

```
uid          Simon Barner <barner@informatik.tu-muenchen.de>
uid          Simon Barner <barner@gmx.de>
sub 2048g/F63052DE 2000-11-10
```

D.3.22. Artem Belevich <art@FreeBSD.org>

```
pub 2048R/9ED4C836 2011-03-28
   Key fingerprint = 7400 D541 07ED 3DF3 3E97 F2D5 8BDF 101C 9ED4 C836
uid          Artem Belevich <artemb@gmail.com>
uid          Artem Belevich <art@freebsd.org>
sub 2048R/55B0E4EB 2011-03-28
```

D.3.23. Anton Berezin <tobez@FreeBSD.org>

```
pub 1024D/7A7BA3C0 2000-05-25 Anton Berezin <tobez@catpipe.net>
   Key fingerprint = CDD8 560C 174B D8E5 0323 83CE 22CA 584C 7A7B A3C0
uid          Anton Berezin <tobez@tobez.org>
uid          Anton Berezin <tobez@FreeBSD.org>
sub 1024g/ADC71E87 2000-05-25
```

D.3.24. Damien Bergamini <damien@FreeBSD.org>

```
pub 2048R/D129F093 2005-03-02
   Key fingerprint = D3AB 28C3 1A4A E219 3145 54FE 220A 7486 D129 F093
uid          Damien Bergamini <damien.bergamini@free.fr>
uid          Damien Bergamini <damien@FreeBSD.org>
sub 2048R/9FBA73A4 2005-03-02
```

D.3.25. Tim Bishop <tdb@FreeBSD.org>

```
pub 1024D/5AE7D984 2000-10-07
   Key fingerprint = 1453 086E 9376 1A50 ECF6 AE05 7DCE D659 5AE7 D984
uid          Tim Bishop <tim@bishnet.net>
uid          Tim Bishop <T.D.Bishop@kent.ac.uk>
uid          Tim Bishop <tdb@i-scream.org>
uid          Tim Bishop <tdb@FreeBSD.org>
sub 4096g/7F886031 2000-10-07
```

D.3.26. Grzegorz Blach <gblach@FreeBSD.org>

```
pub 2048R/D25B0682 2012-11-03 [expires: 2014-11-03]
   Key fingerprint = 225B 941C A886 05C6 1C87 9C03 DE72 593D D25B 0682
uid          Grzegorz Blach <gblach@FreeBSD.org>
sub 2048R/5DE28719 2012-11-03 [expires: 2014-11-03]
```

D.3.27. Martin Blapp <mbr@FreeBSD.org>

```
pub 1024D/D300551E 2001-12-20 Martin Blapp <mb@imp.ch>
    Key fingerprint = B434 53FC C87C FE7B 0A18 B84C 8686 EF22 D300 551E
sub 1024g/998281C8 2001-12-20
```

D.3.28. Warren Block <wblock@FreeBSD.org>

```
pub 2048R/A1F360A3 2011-09-14
    Key fingerprint = 3A44 4DEC B304 5191 8A41 C317 5117 4BB6 A1F3 60A3
uid Warren Block <wblock@FreeBSD.org>
uid Warren Block <wblock@wonkity.com>
sub 2048R/51F483F3 2011-09-14
```

D.3.29. Vitaly Bogdanov <bvs@FreeBSD.org>

```
pub 1024D/B32017F7 2005-10-02 Vitaly Bogdanov <gad@gad.glazov.net>
    Key fingerprint = 402E B8E4 53CB 22FF BE62 AE35 A0BF B077 B320 17F7
uid Vitaly Bogdanov <bvs@freebsd.org>
sub 1024g/0E88C62E 2005-10-02
```

D.3.30. Roman Bogorodskiy <novel@FreeBSD.org>

```
pub 2048R/08C2226A 2010-12-03
    Key fingerprint = 8BA4 DF2A D14F 99B6 37E0 0070 C96D 5FFE 08C2 226A
uid Roman Bogorodskiy <bogorodskiy@gmail.com>
uid Roman Bogorodskiy <novel@FreeBSD.org>
uid Roman Bogorodskiy <rbogorodskiy@apache.org>
uid Roman Bogorodskiy <rbogorodskiy@gridynamics.com>
sub 2048R/EC4ED237 2010-12-03
```

D.3.31. Renato Botelho <garga@FreeBSD.org>

```
pub 4096R/9F625790 2012-11-28 [expires: 2017-11-27]
    Key fingerprint = E3DA 9B2A 6160 99CB 4B31 7641 F1F0 E7A1 9F62 5790
uid Renato Botelho (FreeBSD) <garga@FreeBSD.org>
uid Renato Botelho (Personal) <rbgarga@gmail.com>
uid Renato Botelho (FreeBSD) <garga.bsd@gmail.com>
sub 4096R/473CC82A 2012-11-28 [expires: 2017-11-27]
```

D.3.32. Alexander Botero-Lowry <alexbl@FreeBSD.org>

```
pub 1024D/12A95A7B 2006-09-13
    Key fingerprint = D0C3 47F8 AE87 C829 0613 3586 24DF F52B 12A9 5A7B
uid Alexander Botero-Lowry <alexbl@FreeBSD.org>
sub 2048g/CA287923 2006-09-13
```

D.3.33. Sofian Brabez <sbz@FreeBSD.org>

```
pub 1024D/2487E57E 2011-03-15 [expires: 2016-03-14]
    Key fingerprint = 05BA DC7E F628 DE3F B241 BFBB 7363 51F4 2487 E57E
uid Sofian Brabez <sbrabez@gmail.com>
uid Sofian Brabez <sbz@FreeBSD.org>
uid Sofian Brabez <sbz@6dev.net>
```

D.3.34. Edson Brandi <ebrandi@FreeBSD.org>

```
pub 3072R/FFD3035B 2012-11-26 [expires: 2017-11-25]
    Key fingerprint = 443B 5363 564F 06C3 EA54 9482 209E 9B54 FFD3 035B
uid Edson Brandi <ebrandi@FreeBSD.org>
uid Edson Brandi <ebrandi@fugspbr.org>
uid Edson Brandi <ebrandi@ebrandi.eti.br>
uid Edson Brandi <edson.brandi@gmail.com>
uid Edson Brandi <ebrandi@primeirospassos.org>
uid Edson Brandi <ebrandi@gmail.com>
uid Edson Brandi <ebrandi@fug.com.br>
uid Edson Brandi <contato@edsonbrandi.com>
uid Edson Brandi (Born 1977-08-14 in S. S. DA GRAMA, SP - Brazil)
sub 3072R/A34B8175 2012-11-26 [expires: 2013-11-26]
sub 3072R/4EB0E0EA 2012-11-26 [expires: 2013-11-26]
sub 3072R/89917E73 2012-11-26 [expires: 2013-11-26]
```

D.3.35. Hartmut Brandt <harti@FreeBSD.org>

```
pub 1024D/5920099F 2003-01-29 Hartmut Brandt <brandt@fokus.fraunhofer.de>
    Key fingerprint = F60D 09A0 76B7 31EE 794B BB91 082F 291D 5920 099F
uid Hartmut Brandt <harti@freebsd.org>
sub 1024g/21D30205 2003-01-29
```

D.3.36. Oliver Braun <obraun@FreeBSD.org>

```
pub 1024D/EF25B1BA 2001-05-06 Oliver Braun <obraun@unsane.org>
    Key fingerprint = 6A3B 042A 732E 17E4 B6E7 3EAF C0B1 6B7D EF25 B1BA
uid Oliver Braun <obraun@obraun.net>
uid Oliver Braun <obraun@freebsd.org>
uid Oliver Braun <obraun@haskell.org>
```

```
sub 1024g/09D28582 2001-05-06
```

D.3.37. Max Brazhnikov <makc@FreeBSD.org>

```
pub 1024D/ACB3CD12 2008-08-18
   Key fingerprint = 4BAA 200E 720A 0BD1 7BB0 9DFD FBD9 08C2 ACB3 CD12
uid          Max Brazhnikov <makc@FreeBSD.org>
uid          Max Brazhnikov <makc@issp.ac.ru>
sub 1024g/5FAA4088 2008-08-18
```

D.3.38. Jonathan M. Bresler <jmb@FreeBSD.org>

```
pub 1024R/97E638DD 1996-06-05 Jonathan M. Bresler <jmb@Bresler.org>
   Key fingerprint = 31 57 41 56 06 C1 40 13 C5 1C E3 E5 DC 62 0E FB
uid          Jonathan M. Bresler <jmb@FreeBSD.ORG>
uid          Jonathan M. Bresler
uid          Jonathan M. Bresler <Jonathan.Bresler@USi.net>
uid          Jonathan M. Bresler <jmb@Frb.GOV>
```

D.3.39. Antoine Brodin <antoine@FreeBSD.org>

```
pub 1024D/50CC2671 2008-02-03
   Key fingerprint = F3F7 72F0 9C4C 9E56 4BE9 44EA 1B80 31F3 50CC 2671
uid          Antoine Brodin <antoine@FreeBSD.org>
sub 2048g/6F4AFBE5 2008-02-03
```

D.3.40. Diane Bruce <db@FreeBSD.org>

```
pub 2048R/8E9CAA7B 2012-05-16
   Key fingerprint = 8B08 E022 705D 0083 64C4 5E60 5148 0C74 8E9C AA7B
uid          Diane Bruce <db@db.net>
uid          Diane Bruce <db@FreeBSD.org>
sub 2048R/932E5985 2012-05-16
```

D.3.41. Christian Brueffer <brueffer@FreeBSD.org>

```
pub 1024D/A0ED982D 2002-10-14 Christian Brueffer <chris@unixpages.org>
   Key fingerprint = A5C8 2099 19FF AACA F41B B29B 6C76 178C A0ED 982D
uid          Christian Brueffer <brueffer@hitnet.rwth-aachen.de>
uid          Christian Brueffer <brueffer@FreeBSD.org>
sub 4096g/1DCC100F 2002-10-14
```

D.3.42. Markus Brueffer <markus@FreeBSD.org>

```

pub 1024D/78F8A8D4 2002-10-21
    Key fingerprint = 3F9B EBE8 F290 E5CC 1447 8760 D48D 1072 78F8 A8D4
uid          Markus Brueffer <markus@brueffer.de>
uid          Markus Brueffer <buff@hitnet.rwth-aachen.de>
uid          Markus Brueffer <mbrueffer@mi.rwth-aachen.de>
uid          Markus Brueffer <markus@FreeBSD.org>
sub 4096g/B7E5C7B6 2002-10-21

```

D.3.43. Sean Bruno <sbruno@FreeBSD.org>

```

pub 2048R/08E81687 2012-10-15
    Key fingerprint = B9F9 138F 349C D3B2 2AA4 1398 1909 45DC 08E8 1687
uid          Sean Bruno (clusteradm and developer key) <sbruno@freebsd.org>
sub 2048R/BCC23981 2012-10-15

```

D.3.44. Oleg Bulyzhin <oleg@FreeBSD.org>

```

pub 1024D/78CE105F 2004-02-06
    Key fingerprint = 98CC 3E66 26DE 50A8 DBC4 EB27 AF22 DCEF 78CE 105F
uid          Oleg Bulyzhin <oleg@FreeBSD.org>
uid          Oleg Bulyzhin <oleg@rinet.ru>
sub 1024g/F747C159 2004-02-06

```

D.3.45. Michael Bushkov <bushman@FreeBSD.org>

```

pub 1024D/F694C6E4 2007-03-11 [expires: 2008-03-10]
    Key fingerprint = 4278 4392 BF6B 2864 C48E 0FA9 7216 C73C F694 C6E4
uid          Michael Bushkov <bushman@rsu.ru>
uid          Michael Bushkov <bushman@freebsd.org>
sub 2048g/5A783997 2007-03-11 [expires: 2008-03-10]

```

D.3.46. Jayachandran C. <jchandra@FreeBSD.org>

```

pub 1024D/3316E465 2010-05-19
    Key fingerprint = 320B DB08 4FE3 BCFD 60AF E4DB F486 015F 3316 E465
uid          Jayachandran C. <jchandra@freebsd.org>
sub 2048g/1F7755F9 2010-05-19

```

D.3.47. Jesus R. Camou <jcamou@FreeBSD.org>

```
pub 1024D/C2161947 2005-03-01
   Key fingerprint = 274C B265 48EC 42AE A2CA 47D9 7D98 588A C216 1947
uid      Jesus R. Camou <jcamou@FreeBSD.org>
sub 2048g/F8D2A8DF 2005-03-01
```

D.3.48. José Alonso Cárdenas Márquez <acm@FreeBSD.org>

```
pub 1024D/9B21BC19 2006-07-18
   Key fingerprint = 4156 2EAC A11C 9651 713B 3FC1 195F D4A8 9B21 BC19
uid      Jose Alonso Cardenas Marquez <acm@FreeBSD.org>
sub 2048g/ADA16C52 2006-07-18
```

D.3.49. Pietro Cerutti <gahr@FreeBSD.org>

```
pub 1024D/9571F78E 2006-05-17
   Key fingerprint = 1203 92B5 3919 AF84 9B97 28D6 C0C2 6A98 9571 F78E
uid      Pietro Cerutti <gahr@gahr.ch>
uid      Pietro Cerutti (The FreeBSD Project) <gahr@FreeBSD.org>
sub 2048g/F24227D5 2006-05-17 [expires: 2011-05-16]
```

D.3.50. Dmitry Chagin <dchagin@FreeBSD.org>

```
pub 1024D/738EFCED 2009-02-27
   Key fingerprint = 3F3F 8B87 CE09 9E10 3606 6ACA D2DD 936F 738E FCED
uid      Dmitry Chagin <dchagin@freebsd.org>
uid      Dmitry Chagin (dchagin key) <chagin.dmitry@gmail.com>
sub 2048g/6A3FDFF9 2009-02-27
```

D.3.51. Hye-Shik Chang <perky@FreeBSD.org>

```
pub 1024D/CFDB4BA4 1999-04-23 Hye-Shik Chang <perky@FreeBSD.org>
   Key fingerprint = 09D9 57D6 58BA 44DD CAEC 71CD 0D65 2C59 CFDB 4BA4
uid      Hye-Shik Chang <hyeshik@gmail.com>
sub 1024g/A94A8ED1 1999-04-23
```

D.3.52. Jonathan Chen <jon@FreeBSD.org>

```
pub 1024D/2539468B 1999-10-11 Jonathan Chen <jon@spock.org>
   Key fingerprint = EE31 CDA1 A105 C8C9 5365 3DB5 C2FC 86AA 2539 468B
uid      Jonathan Chen <jon@freebsd.org>
uid      Jonathan Chen <chenj@rpi.edu>
uid      Jonathan Chen <spock@acm.rpi.edu>
```

```
uid                               Jonathan Chen <jon@cs.rpi.edu>
sub 3072g/B81EF1DB 1999-10-11
```

D.3.53. Jonathan Anderson <jonathan@FreeBSD.org>

```
pub 1024D/E3BBCA48 2006-06-17
   Key fingerprint = D7C6 9096 874F 707E 48F8 FAB7 22A6 6E53 E3BB CA48
uid                               Jonathan Anderson <jonathan@FreeBSD.org>
uid                               Jonathan Anderson <jonathan.anderson@ieee.org>
uid                               Jonathan Anderson <anderson@engr.mun.ca>
uid                               Jonathan Anderson <jonathan.anderson@mun.ca>
sub 2048g/A703650D 2006-06-17
```

D.3.54. Fukang Chen <loader@FreeBSD.org>

```
pub 4096R/6BD4DDE6 2012-10-26
   Key fingerprint = A33E 88AB D358 DA49 59A6 B263 A9A2 599C 6BD4 DDE6
uid                               loader <loader@FreeBSD.org>
uid                               loader <loader@FreeBSDMall.com>
sub 4096R/1036D26C 2012-10-26
```

D.3.55. Luoqi Chen <luoqi@FreeBSD.org>

```
pub 1024D/2926F3BE 2002-02-22 Luoqi Chen <luoqi@FreeBSD.org>
   Key fingerprint = B470 A815 5917 D9F4 37F3 CE2A 4D75 3BD1 2926 F3BE
uid                               Luoqi Chen <luoqi@bricore.com>
uid                               Luoqi Chen <lchen@onetta.com>
sub 1024g/5446EB72 2002-02-22
```

D.3.56. Andrey A. Chernov <ache@FreeBSD.org>

```
pub 1024D/964474DD 2006-12-26
   Key fingerprint = 0F63 1B61 D76D AA23 1591 EA09 560E 582B 9644 74DD
uid                               Andrey Chernov <ache@freebsd.org>
uid                               [jpeg image of size 4092]
sub 2048g/08331894 2006-12-26
```

D.3.57. Alexander V. Chernikov <melifaro@FreeBSD.org>

```
pub 1024D/2675AB69 2008-02-17
   Key fingerprint = 00D2 E063 2FB0 2990 C602 50FD C1C2 7889 2675 AB69
uid                               Alexander V. Chernikov <melifaro@yandex-team.ru>
uid                               Alexander V. Chernikov <melifaro@ipfw.ru>
uid                               Alexander V. Chernikov <melifaro@freebsd.org>
```

sub 4096g/BC64F40C 2008-02-17

D.3.58. Sean Chittenden <seanc@FreeBSD.org>

pub 1024D/EE278A28 2004-02-08 Sean Chittenden <sean@chittenden.org>
 Key fingerprint = E41F F441 7E91 6CBA 1844 65CF B939 3C78 EE27 8A28
 sub 2048g/55321853 2004-02-08

D.3.59. Junho CHOI <cjh@FreeBSD.org>

pub 1024D/E60260F5 2002-10-14 CHOI Junho (Work) <cjh@wdb.co.kr>
 Key fingerprint = 1369 7374 A45F F41A F3C0 07E3 4A01 C020 E602 60F5
 uid CHOI Junho (Personal) <cjh@kr.FreeBSD.org>
 uid CHOI Junho (FreeBSD) <cjh@FreeBSD.org>
 sub 1024g/04A4FDD8 2002-10-14

D.3.60. Crist J. Clark <cjc@FreeBSD.org>

pub 1024D/FE886AD3 2002-01-25 Crist J. Clark <cjclark@jhu.edu>
 Key fingerprint = F04E CCD7 3834 72C2 707F 0A8F 259F 8F4B FE88 6AD3
 uid Crist J. Clark <cjclark@alum.mit.edu>
 uid Crist J. Clark <cjc@freebsd.org>
 sub 1024g/9B6BAB99 2002-01-25

D.3.61. Joe Marcus Clarke <marcus@FreeBSD.org>

pub 1024D/FE14CF87 2002-03-04 Joe Marcus Clarke (FreeBSD committer address) <marcus@FreeBSD.org>
 Key fingerprint = CC89 6407 73CC 0286 28E4 AFB9 6F68 8F8A FE14 CF87
 uid Joe Marcus Clarke <marcus@marcuscom.com>
 sub 1024g/B9ACE4D2 2002-03-04

D.3.62. Nik Clayton <nik@FreeBSD.org>

pub 1024D/2C37E375 2000-11-09 Nik Clayton <nik@freebsd.org>
 Key fingerprint = 15B8 3FFC DDB4 34B0 AA5F 94B7 93A8 0764 2C37 E375
 uid Nik Clayton <nik@slashdot.org>
 uid Nik Clayton <nik@crf-consulting.co.uk>
 uid Nik Clayton <nik@ngo.org.uk>
 uid Nik Clayton <nik@bsdi.com>
 sub 1024g/769E298A 2000-11-09

D.3.63. Benjamin Close <benjsc@FreeBSD.org>

```
pub 1024D/4842B5B4 2002-04-10
    Key fingerprint = F00D C83D 5F7E 5561 DF91 B74D E602 CAA3 4842 B5B4
uid Benjamin Simon Close <Benjamin.Close@clearchain.com>
uid Benjamin Simon Close <benjsc@FreeBSD.org>
uid Benjamin Simon Close <benjsc@clearchain.com>
sub 2048g/3FA8A57E 2002-04-10
```

D.3.64. Tijl Coosemans <tijl@FreeBSD.org>

```
pub 2048D/20A0B62B 2010-07-13
    Key fingerprint = 39AA F580 6B44 5161 9F86 ED49 7E80 92D8 20A0 B62B
uid Tijl Coosemans <tijl@coosemans.org>
uid Tijl Coosemans <tijl@freebsd.org>
sub 2048g/7D71BA74 2010-07-13
```

D.3.65. Raphael Kubo da Costa <rakuco@FreeBSD.org>

```
pub 4096R/18DCEED6 2011-10-03
    Key fingerprint = 6911 54FE BA6E 6106 5789 7099 8DD0 7D21 18DC EED6
uid Raphael Kubo da Costa (Personal key) <rakuco@FreeBSD.org>
```

D.3.66. Alan L. Cox <alc@FreeBSD.org>

```
pub 2048R/33E2893B 2013-06-15
    Key fingerprint = FC7C 93FD 2C2C ABA5 C1D1 3E74 8513 043C 33E2 893B
uid Alan Cox <alc@FreeBSD.org>
uid Alan Cox <alc@cs.rice.edu>
uid Alan Cox <alc@rice.edu>
sub 2048R/693757AA 2013-06-15
```

D.3.67. Bruce Cran <brucec@FreeBSD.org>

```
pub 2048R/6AF6F99E 2010-01-29
    Key fingerprint = 9A3C AE57 2706 B0E3 4B8A 8374 5787 A72B 6AF6 F99E
uid Bruce Cran <brucec@FreeBSD.org>
uid Bruce Cran <bruce@cran.org.uk>
sub 2048R/1D665CEE 2010-01-29
```

D.3.68. Frederic Culot <culot@FreeBSD.org>

```
pub 1024D/34876C5B 2006-08-26
    Key fingerprint = 50EE CE94 E43E BA85 CB67 262B B739 1A26 3487 6C5B
uid Frederic Culot <culot@FreeBSD.org>
uid Frederic Culot <frederic@culot.org>
sub 2048g/F1EF901F 2006-08-26
```

D.3.69. Aaron Dalton <aaron@FreeBSD.org>

```
pub 1024D/8811D2A4 2006-06-21 [expires: 2011-06-20]
    Key fingerprint = 8DE0 3CBB 3692 992F 53EF ACC7 BE56 0A4D 8811 D2A4
uid Aaron Dalton <aaron@freebsd.org>
sub 2048g/304EE8E5 2006-06-21 [expires: 2011-06-20]
```

D.3.70. Baptiste Daroussin <bapt@FreeBSD.org>

```
pub 1024D/49A4E84C 2008-11-19
    Key fingerprint = A14B A5FC B860 86DE 73E2 B24C F244 ED31 49A4 E84C
uid Baptiste Daroussin <bapt@etoilebsd.net>
uid Baptiste Daroussin <baptiste.daroussin@gmail.com>
uid Baptiste Daroussin <bapt@FreeBSD.org>
sub 2048g/54AB46B4 2008-11-19
```

D.3.71. Ceri Davies <ceri@FreeBSD.org>

```
pub 1024D/34B7245F 2002-03-08
    Key fingerprint = 9C88 EB05 A908 1058 A4AE 9959 A1C7 DCC1 34B7 245F
uid Ceri Davies <ceri@submonkey.net>
uid Ceri Davies <ceri@FreeBSD.org>
uid Ceri Davies <ceri@opensolaris.org>
sub 1024g/0C482CBC 2002-03-08
```

D.3.72. Brad Davis <brd@FreeBSD.org>

```
pub 1024D/ED0A754D 2005-05-14 [expires: 2014-02-21]
    Key fingerprint = 5DFD D1A6 BEEE A6D4 B3F5 4236 D362 3291 ED0A 754D
uid Brad Davis <sol4k@sol4k.com>
uid Brad Davis <brd@FreeBSD.org>
sub 2048g/1F29D404 2005-05-14 [expires: 2014-02-21]
```

D.3.73. Pawel Jakub Dawidek <pjd@FreeBSD.org>

```
pub 1024D/B1293F34 2004-02-02 Pawel Jakub Dawidek <Pawel@Dawidek.net>
    Key fingerprint = A3A3 5B4D 9CF9 2312 0783 1B1D 168A EF5D B129 3F34
uid                                     Pawel Jakub Dawidek <pjd@FreeBSD.org>
uid                                     Pawel Jakub Dawidek <pjd@FreeBSD.pl>
sub 2048g/3EEC50A7 2004-02-02 [expires: 2006-02-01]
```

D.3.74. Brian S. Dean <bsd@FreeBSD.org>

```
pub 1024D/723BDEE9 2002-01-23 Brian S. Dean <bsd@FreeBSD.org>
    Key fingerprint = EF49 7ABE 47ED 91B3 FC3D 7EA5 4D90 2FF7 723B DEE9
sub 1024g/4B02F876 2002-01-23
```

D.3.75. Carl Delsey <carl@FreeBSD.org>

```
pub 4096R/FB3B5D38 2013-01-15
    Key fingerprint = F0E5 3849 C6C3 668B 68A3 BCC7 6031 E963 FB3B 5D38
uid                                     Carl Delsey <carl@FreeBSD.org>
sub 4096R/256F29D3 2013-01-15
```

D.3.76. Vasil Dimov <vd@FreeBSD.org>

```
pub 1024D/F6C1A420 2004-12-08
    Key fingerprint = B1D5 04C6 26CC 0D20 9525 14B8 170E 923F F6C1 A420
uid                                     Vasil Dimov <vd@FreeBSD.org>
uid                                     Vasil Dimov <vd@datamax.bg>
sub 4096g/A0148C94 2004-12-08
```

D.3.77. Roman Divacky <rdivacky@FreeBSD.org>

```
pub 1024D/3DC2044C 2006-11-15
    Key fingerprint = 6B61 25CA 49BC AAC5 21A9 FA7A 2D51 23E8 3DC2 044C
uid                                     Roman Divacky <rdivacky@freebsd.org>
sub 2048g/39BDCE16 2006-11-15
```

D.3.78. Alexey Dokuchaev <danfe@FreeBSD.org>

```
pub 1024D/3C060B44 2004-08-23 Alexey Dokuchaev <danfe@FreeBSD.org>
    Key fingerprint = D970 08A4 922C 8D63 0C19 8D27 F421 76EE 3C06 0B44
sub 1024g/70BAE967 2004-08-23
```

D.3.79. Dima Dorfman <dd@FreeBSD.org>

```

pub 1024D/69FAE582 2001-09-04
    Key fingerprint = B340 8338 7DA3 4D61 7632 098E 0730 055B 69FA E582
uid          Dima Dorfman <dima@trit.org>
uid          Dima Dorfman <dima@unixfreak.org>
uid          Dima Dorfman <dd@freebsd.org>
sub 2048g/65AF3B89 2003-08-19 [expires: 2005-08-18]
sub 2048g/8DB0CF2C 2005-05-29 [expires: 2007-05-29]

```

D.3.80. Bryan Drewery <bdrewery@FreeBSD.org>

```

pub 4096R/3C9B0CF9 2012-04-06 [expires: 2017-04-05]
    Key fingerprint = 36FE BE99 2F52 80DF 4811 362A 6E78 2AC0 3C9B 0CF9
uid          Bryan Drewery <bryan@shatow.net>
uid          Bryan Drewery <bdrewery@gmail.com>
uid          Bryan Drewery <bryan@xzibition.com>
uid          Bryan Drewery <bdrewery@FreeBSD.org>
sub 4096R/9E2CE2D3 2012-04-06 [expires: 2017-04-05]

```

D.3.81. Olivier Duchateau <olivierd@FreeBSD.org>

```

pub 2048R/22431859 2012-05-28 [expires: 2017-05-27]
    Key fingerprint = C057 112A 4A27 B5F2 CD8F 6C9A FC5A 0167 2243 1859
uid          Olivier Duchateau <duchateau.olivier@gmail.com>
sub 2048R/63A85BDF 2012-05-28 [expires: 2017-05-27]

```

D.3.82. Bruno Ducrot <bruno@FreeBSD.org>

```

pub 1024D/7F463187 2000-12-29
    Key fingerprint = 7B79 E1D6 F5A1 6614 792F D906 899B 4D28 7F46 3187
uid          Ducrot Bruno (Poup Master) <ducrot@poupinou.org>
sub 1024g/40282874 2000-12-29

```

D.3.83. Alex Dupre <ale@FreeBSD.org>

```

pub 1024D/CE5F554D 1999-06-27 Alex Dupre <sysadmin@alexdupre.com>
    Key fingerprint = DE23 02EA 5927 D5A9 D793 2BA2 8115 E9D8 CE5F 554D
uid          Alex Dupre <ale@FreeBSD.org>
uid          [jpeg image of size 5544]
uid          Alex Dupre <ICQ:5431856>
sub 2048g/FD5E2D21 1999-06-27

```

D.3.84. Peter Edwards <peadar@FreeBSD.org>

```
pub 1024D/D80B4B3F 2004-03-01 Peter Edwards <peadar@FreeBSD.org>
   Key fingerprint = 7A8A 9756 903E BEF2 4D9E 3C94 EE52 52F7 D80B 4B3F
uid                                     Peter Edwards <pmedwards@eircom.net>
```

D.3.85. Daniel Eischen <deischen@FreeBSD.org>

```
pub 4096R/7D15560B 2012-11-17
   Key fingerprint = 0039 2133 69CA 14D3 236A E331 361A 68B2 7D15 560B
uid                                     Daniel Eischen <deischen@FreeBSD.org>
sub 4096R/A51F81F7 2012-11-17
```

D.3.86. Josef El-Rayes <josef@FreeBSD.org>

```
pub 2048R/A79DB53C 2004-01-04 Josef El-Rayes <josef@FreeBSD.org>
   Key fingerprint = 58EB F5B7 2AB9 37FE 33C8 716B 59C5 22D9 A79D B53C
uid                                     Josef El-Rayes <josef@daemon.li>
```

D.3.87. Lars Engels <lme@FreeBSD.org>

```
pub 1024D/C0F769F8 2004-08-27
   Key fingerprint = 17FC 08E1 5E09 BD21 489E 2050 29CE 75DA C0F7 69F8
uid                                     Lars Engels <lars.engels@0x20.net>
sub 1024g/8AD5BF9D 2004-08-27
```

D.3.88. Udo Erdelhoff <ue@FreeBSD.org>

```
pub 1024R/E74FA871 1994-07-19 Udo Erdelhoff <uer@de.uu.net>
   Key fingerprint = 8C B1 80 CA 2C 52 73 81 FB A7 B4 03 C5 32 C8 67
uid                                     Udo Erdelhoff <ue@nathan.ruhr.de>
uid                                     Udo Erdelhoff <ue@freebsd.org>
uid                                     Udo Erdelhoff <uerdelho@eu.uu.net>
uid                                     Udo Erdelhoff <uerdelho@uu.net>
```

D.3.89. Ruslan Ermilov <ru@FreeBSD.org>

```
pub 1024D/996E145E 2004-06-02 Ruslan Ermilov (FreeBSD) <ru@FreeBSD.org>
   Key fingerprint = 274E D201 71ED 11F6 9CCB 0194 A917 E9CC 996E 145E
uid                                     Ruslan Ermilov (FreeBSD Ukraine) <ru@FreeBSD.org.ua>
uid                                     Ruslan Ermilov (IPNet) <ru@ip.net.ua>
sub 1024g/557E3390 2004-06-02 [expires: 2007-06-02]
```

D.3.90. Lukas Ertl <le@FreeBSD.org>

```
pub 1024D/F10D06CB 2000-11-23 Lukas Ertl <le@FreeBSD.org>
   Key fingerprint = 20CD C5B3 3A1D 974E 065A B524 5588 79A9 F10D 06CB
uid                                     Lukas Ertl <a9404849@unet.univie.ac.at>
uid                                     Lukas Ertl <l.ertl@univie.ac.at>
uid                                     Lukas Ertl <le@univie.ac.at>
sub 1024g/5960CE8E 2000-11-23
```

D.3.91. Brendan Fabeny <bf@FreeBSD.org>

```
pub 2048R/9806EBC1 2010-06-08 [expires: 2012-06-07]
   Key fingerprint = 2075 ADD3 7634 A4F9 5357 D934 08E7 06D9 9806 EBC1
uid                                     b. f. <bf@freebsd.org>
sub 2048R/1CD0AD79 2010-06-08 [expires: 2012-06-07]
```

D.3.92. Guido Falsi <madpilot@FreeBSD.org>

```
pub 2048R/56CBD293 2012-04-12
   Key fingerprint = F317 2057 E17E 4E3A 3DA5 9E1D 1AE6 860E 56CB D293
uid                                     Guido Falsi <madpilot@FreeBSD.org>
uid                                     Guido Falsi <mad@madpilot.net>
sub 2048R/1F9772C5 2012-04-12
```

D.3.93. Rong-En Fan <rafan@FreeBSD.org>

```
pub 1024D/86FD8C68 2004-06-04
   Key fingerprint = DC9E 5B4D 2DDA D5C7 B6F8 6E69 D78E 1091 86FD 8C68
uid                                     Rong-En Fan <rafan@infor.org>
uid                                     Rong-En Fan <rafan@csie.org>
uid                                     Rong-En Fan <rafan@FreeBSD.org>
sub 2048g/42A8637E 2009-01-25 [expires: 2012-07-08]
```

D.3.94. Stefan Farfeleder <stefanf@FreeBSD.org>

```
pub 1024D/8BEFD15F 2004-03-14 Stefan Farfeleder <stefan@fafoe.narf.at>
   Key fingerprint = 4220 FE60 A4A1 A490 5213 27A6 319F 8B28 8BEF D15F
uid                                     Stefan Farfeleder <stefanf@complang.tuwien.ac.at>
uid                                     Stefan Farfeleder <stefanf@FreeBSD.org>
uid                                     Stefan Farfeleder <stefanf@ten15.org>
sub 2048g/418753E9 2004-03-14 [expires: 2007-03-14]
```

D.3.95. Babak Farrokhi <farrokhi@FreeBSD.org>

```
pub 1024D/7C810476 2005-12-22
    Key fingerprint = AABD 388F A207 58B4 2EE3 5DFD 4FC1 32C3 7C81 0476
uid Babak Farrokhi <farrokhi@FreeBSD.org>
uid Babak Farrokhi <babak@farrokhi.net>
sub 2048g/2A5F93C7 2005-12-22
```

D.3.96. Chris D. Faulhaber <jedgar@FreeBSD.org>

```
pub 1024D/FE817A50 2000-12-20 Chris D. Faulhaber <jedgar@FreeBSD.org>
    Key fingerprint = A47D A838 9216 F921 A456 54FF 39B6 86E0 FE81 7A50
uid Chris D. Faulhaber <jedgar@fxp.org>
sub 2048g/93452698 2000-12-20
```

D.3.97. Mark Felder <feld@FreeBSD.org>

```
pub 2048R/E64C94FE 2013-06-25
    Key fingerprint = 71ED 6A7F F4D7 430A BDF3 A180 BF01 619F E64C 94FE
uid Mark Felder <feld@freebsd.org>
sub 2048R/FDC20CA9 2013-06-25
```

D.3.98. Brian F. Feldman <green@FreeBSD.org>

```
pub 1024D/41C13DE3 2000-01-11 Brian Fundakowski Feldman <green@FreeBSD.org>
    Key fingerprint = 6A32 733A 1BF6 E07B 5B8D AE14 CC9D DCA2 41C1 3DE3
sub 1024g/A98B9FCC 2000-01-11 [expires: 2001-01-10]

pub 1024D/773905D6 2000-09-02 Brian Fundakowski Feldman <green@FreeBSD.org>
    Key fingerprint = FE23 7481 91EA 5E58 45EA 6A01 B552 B043 7739 05D6
sub 2048g/D2009B98 2000-09-02
```

D.3.99. Mário Sérgio Fujikawa Ferreira <lioux@FreeBSD.org>

```
pub 1024D/75A63712 2006-02-23 [expires: 2007-02-23]
    Key fingerprint = 42F2 2F74 8EF9 5296 898F C981 E9CF 463B 75A6 3712
uid Mario Sergio Fujikawa Ferreira (lioux) <lioux@FreeBSD.org>
uid Mario Sergio Fujikawa Ferreira <lioux@uol.com.br>
sub 4096g/BB7D80F2 2006-02-23 [expires: 2007-02-23]
```

D.3.100. Matthew Fleming <mdf@FreeBSD.org>

```
pub 2048R/A783DAA2 2012-11-22 [expires: 2016-11-22]
    Key fingerprint = 773F E069 BE98 CE96 4AC6 B8AB 1A1B 255E A783 DAA2
uid      Matthew D Fleming <mdf356@gmail.com>
uid      Matthew D Fleming <mdf@FreeBSD.org>
sub 2048R/4015B7AA 2012-11-22 [expires: 2016-11-22]
```

D.3.101. Tony Finch <fanf@FreeBSD.org>

```
pub 1024D/84C71B6E 2002-05-03 Tony Finch <dot@dotat.at>
    Key fingerprint = 199C F25B 2679 6D04 63C5 2159 FFC0 F14C 84C7 1B6E
uid      Tony Finch <fanf@FreeBSD.org>
uid      Tony Finch <fanf@apache.org>
uid      Tony Finch <fanf2@cam.ac.uk>
sub 2048g/FD101E8B 2002-05-03
```

D.3.102. Marc Fonvieille <blackend@FreeBSD.org>

```
pub 1024D/4F8E74E8 2004-12-25 Marc Fonvieille <blackend@FreeBSD.org>
    Key fingerprint = 55D3 4883 4A04 828A A139 A5CF CD0F 51C0 4F8E 74E8
uid      Marc Fonvieille <marc@blackend.org>
uid      Marc Fonvieille <marc@freebsd-fr.org>
sub 1024g/37AD4E7D 2004-12-25
```

D.3.103. Pete Fritchman <petef@FreeBSD.org>

```
pub 1024D/74B91CFD 2001-01-30 Pete Fritchman <petef@FreeBSD.org>
    Key fingerprint = 9A9F 8A13 DB0D 7777 8D8E 1CB2 C5C9 A08F 74B9 1CFD
uid      Pete Fritchman <petef@databits.net>
uid      Pete Fritchman <petef@csh.rit.edu>
sub 1024g/0C02AF0C 2001-01-30
```

D.3.104. Bernhard Fröhlich <decke@FreeBSD.org>

```
pub 1024D/CF5840D4 2008-01-07 [expires: 2015-05-05]
    Key fingerprint = 47F6 BDF1 DF9E 81E2 2C54 8A06 E796 7A5A CF58 40D4
uid      Bernhard Fröhlich <decke@FreeBSD.org>
uid      Bernhard Fröhlich <decke@bluelife.at>
sub 2048g/4E51CE79 2008-01-07
```

D.3.105. Bill Fumerola <billf@FreeBSD.org>

```
pub 1024D/7F868268 2000-12-07 Bill Fumerola (FreeBSD Developer) <billf@FreeBSD.org>
    Key fingerprint = 5B2D 908E 4C2B F253 DAEB FC01 8436 B70B 7F86 8268
uid                                     Bill Fumerola (Security Yahoo) <fumerola@yahoo-inc.com>
sub 1024g/43980DA9 2000-12-07
```

D.3.106. Andriy Gapon <avg@FreeBSD.org>

```
pub 2048R/A651FE2F 2009-02-16
    Key fingerprint = F234 4D58 DEFF 5E3A 4E0F 13BC 74A5 2D27 A651 FE2F
uid                                     Andriy Gapon (FreeBSD) <avg@FreeBSD.org>
uid                                     Andriy Gapon (FreeBSD) <avg@freebsd.org>
uid                                     Andriy Gapon (FreeBSD) <avg@icyb.net.ua>
sub 4096R/F9A4D312 2009-02-16
```

D.3.107. Beat Gätzi <beat@FreeBSD.org>

```
pub 1024D/774249DB 2009-01-28 [expires: 2014-01-27]
    Key fingerprint = C410 3187 5B29 DD02 745F 0890 40C5 BCF7 7742 49DB
uid                                     Beat Gaetzi <beat@FreeBSD.org>
sub 2048g/173CFFCA 2009-01-28 [expires: 2014-01-27]
```

D.3.108. Daniel Geržo <danger@FreeBSD.org>

```
pub 1024D/DA913352 2007-08-30 [expires: 2008-08-29]
    Key fingerprint = 7372 3F15 F839 AFF5 4052 CAC7 1ADA C204 DA91 3352
uid                                     Daniel Gerzo <gerzo@rulez.sk>
uid                                     Daniel Gerzo <danger@rulez.sk>
uid                                     Daniel Gerzo (The FreeBSD Project) <danger@FreeBSD.org>
uid                                     Daniel Gerzo (Micronet, a.s.) <gerzo@micronet.sk>
sub 2048g/C5D57BDC 2007-08-30 [expires: 2008-08-29]
```

D.3.109. Simon J. Gerraty <sjg@FreeBSD.org>

```
pub 1024D/B6CC76BF 2002-06-12
    Key fingerprint = F3BA D6CB E1F8 02EA 705F BCAD 6125 F840 B6CC 76BF
uid                                     Simon J. Gerraty <sjg@cruffy.net>
uid                                     Simon J. Gerraty <sjg@juniper.net>
uid                                     Simon J. Gerraty <sjg@NetBSD.org>
uid                                     Simon J. Gerraty <sjg@FreeBSD.org>
sub 1024g/D94B72B9 2002-06-12
```

D.3.110. Justin T. Gibbs <gibbs@FreeBSD.org>

```
pub 2048R/45A4FC2F 2012-02-10
    Key fingerprint = B98A C3AB 412B 094B D6FE E713 FA5A 1E30 45A4 FC2F
uid Justin T. Gibbs <gibbs@FreeBSD.org>
uid Justin T. Gibbs <gibbs@FreeBSDFoundation.org>
uid Justin T. Gibbs <gibbs@scsiguy.com>
sub 2048R/AF6927F8 2012-02-10
```

D.3.111. Pedro Giffuni <pfg@FreeBSD.org>

```
pub 2048D/422BDFE4 2011-12-06
    Key fingerprint = A12B 7C6B 54C0 921B C64F 7B35 58DF 6813 422B DFE4
uid Pedro Giffuni (FreeBSD key signature) <pfg@FreeBSD.org>
sub 2048g/43A91DE0 2011-12-06
```

D.3.112. Palle Girgensohn <girgen@FreeBSD.org>

```
pub 2048R/4A6BAAAD 2012-02-23 [expires: 2016-02-23]
    Key fingerprint = BD8C 332C E630 31D6 2FDB 80BD 5FF2 A161 4A6B AAAD
uid Palle Girgensohn <girgen@pingpong.net>
uid [jpeg image of size 8260]
uid Palle Girgensohn <girgen@FreeBSD.org>
sub 2048R/6BC41243 2012-02-23 [expires: 2016-02-23]
```

D.3.113. Philip M. Gollucci <pgollucci@FreeBSD.org>

```
pub 1024D/DB9B8C1C 2008-04-15
    Key fingerprint = B90B FBC3 A3A1 C71A 8E70 3F8C 75B8 8FFB DB9B 8C1C
uid Philip M. Gollucci (FreeBSD Foundation) <pgollucci@freebsd.org>
uid Philip M. Gollucci (Riderway Inc.) <pgollucci@riderway.com>
uid Philip M. Gollucci <pgollucci@p6m7g8.com>
uid Philip M. Gollucci (ASF) <pgollucci@apache.org>
sub 2048g/73943732 2008-04-15
```

D.3.114. Daichi GOTO <daichi@FreeBSD.org>

```
pub 1024D/09EBADD6 2002-09-25 Daichi GOTO <daichi@freebsd.org>
    Key fingerprint = 620A 9A34 57FB 5E93 0828 28C7 C360 C6ED 09EB ADD6
sub 1024g/F0B1F1CA 2002-09-25
```

D.3.115. Marcus Alves Grando <mnag@FreeBSD.org>

```
pub 1024D/CDCC273F 2005-09-15 [expires: 2010-09-14]
    Key fingerprint = 57F9 DEC1 5BBF 06DE 44A5 9A4A 8BEE 5F3A CDCC 273F
uid          Marcus Alves Grando <marcus@sbh.eng.br>
uid          Marcus Alves Grando <marcus@corp.grupos.com.br>
uid          Marcus Alves Grando <mnag@FreeBSD.org>
sub 2048g/698AC00C 2005-09-15 [expires: 2010-09-14]
```

D.3.116. Peter Grehan <grehan@FreeBSD.org>

```
pub 1024D/EA45EA7D 2004-07-13 Peter Grehan <grehan@freebsd.org>
    Key fingerprint = 84AD 73DC 370E 15CA 7556 43C8 F5C8 4450 EA45 EA7D
sub 2048g/0E122D70 2004-07-13
```

D.3.117. Jamie Gritton <jamie@FreeBSD.org>

```
pub 1024D/8832CB7F 2009-01-29
    Key fingerprint = 34F8 1E62 C7A5 7CB9 A91F 7864 8C5A F85E 8832 CB7F
uid          James Gritton <jamie@FreeBSD.org>
sub 2048g/94E3594D 2009-01-29
```

D.3.118. William Grzybowski <wg@FreeBSD.org>

```
pub 2048R/CFC460C5 2012-09-28
    Key fingerprint = FC40 5CD8 0879 7F50 0036 D924 D9F7 8B27 CFC4 60C5
uid          William Grzybowski (FreeBSD) <wg@freebsd.org>
uid          William Grzybowski <william88@gmail.com>
sub 2048R/05577997 2012-09-28
```

D.3.119. Barbara Guida <bar@FreeBSD.org>

```
pub 2048R/3DF5F750 2012-11-13
    Key fingerprint = D367 F6C8 2A5F 2921 70D2 B446 27DD 6FD6 3DF5 F750
uid          Barbara Guida <bar@FreeBSD.org>
uid          Barbara Guida <barbara.freebsd@gmail.com>
sub 2048R/1DF7506C 2012-11-13
```

D.3.120. John-Mark Gurney <jmg@FreeBSD.org>

```
pub 1024D/6D3FA396 2011-03-03 [expires: 2016-03-01]
    Key fingerprint = 54BA 873B 6515 3F10 9E88 9322 9CB1 8F74 6D3F A396
uid          John-Mark Gurney <jmg@FreeBSD.org>
uid          John-Mark Gurney <jmg@funkthat.com>
```

sub 4096g/0A4C095E 2011-03-03 [expires: 2016-03-01]

D.3.121. Mateusz Guzik <mjg@FreeBSD.org>

pub 2048R/21489259 2012-06-03
Key fingerprint = 3A9F 25FF ABF6 BB23 5C70 C61B 96D3 5178 2148 9259
uid Mateusz Guzik <mjg@freebsd.org>
sub 2048R/EA19FE8D 2012-06-03

D.3.122. Jason E. Hale <jhale@FreeBSD.org>

pub 3072D/8F2E5907 2012-09-07
Key fingerprint = 009C 54BF 32D0 F373 8126 C8A1 D8DD 2CA4 8F2E 5907
uid Jason E. Hale <jhale@FreeBSD.org>
uid Jason E. Hale <bsdkafee@gmail.com>
sub 4096g/7081A001 2012-09-07

D.3.123. Daniel Harris <dannyboy@FreeBSD.org>

pub 1024D/84D0D7E7 2001-01-15 Daniel Harris <dannyboy@worksforfood.com>
Key fingerprint = 3C61 B8A1 3F09 D194 3259 7173 6C63 DA04 84D0 D7E7
uid Daniel Harris <dannyboy@freebsd.org>
uid Daniel Harris <dh@askdh.com>
uid Daniel Harris <dh@wordassault.com>
sub 1024g/9DF0231A 2001-01-15

D.3.124. Daniel Hartmeier <dhartmei@FreeBSD.org>

pub 1024R/6A3A7409 1994-08-15 Daniel Hartmeier <dhartmei@freebsd.org>
Key fingerprint = 13 7E 9A F3 36 82 09 FE FD 57 B8 5C 2B 81 7E 1F

D.3.125. Oliver Hauer <ohauer@FreeBSD.org>

pub 2048R/5D008F1A 2010-07-26
Key fingerprint = E9EE C9A5 EB4C BD29 74D7 9178 E56E 06B3 5D00 8F1A
uid olli hauer <ohauer@FreeBSD.org>
uid olli hauer <ohauer@gmx.de>
sub 2048R/5E25776E 2010-07-26

D.3.126. Emanuel Haupt <ehaupt@FreeBSD.org>

```
pub 3072D/329A273C 2012-11-17 [expires: 2013-11-17]
    Key fingerprint = 920C A49A 5A23 F9E3 4EB0 4387 AB90 5C56 329A 273C
uid Emanuel Haupt <ehaupt@FreeBSD.org>
sub 3072g/70183B96 2012-11-17 [expires: 2013-11-17]
```

D.3.127. John Hay <jhay@FreeBSD.org>

```
pub 2048R/A9275B93 2000-05-10 John Hay <jhay@icomtek.csir.co.za>
    Key fingerprint = E7 95 F4 B9 D4 A7 49 6A 83 B9 77 49 28 9E 37 70
uid John Hay <jhay@mikom.csir.co.za>
uid Thawte Freemail Member <jhay@mikom.csir.co.za>
uid John Hay <jhay@csir.co.za>
uid John Hay <jhay@FreeBSD.ORG>
```

D.3.128. Sheldon Hearn <sheldonh@FreeBSD.org>

```
pub 1024D/74A06ACD 2002-06-20 Sheldon Hearn <sheldonh@starjuice.net>
    Key fingerprint = 01A3 EF91 9C5A 3633 4E01 8085 A462 57F1 74A0 6ACD
sub 1536g/C42F8AC8 2002-06-20
```

D.3.129. Mike Heffner <mikeh@FreeBSD.org>

```
pub 1024D/CDECBF99 2001-02-02 Michael Heffner <mheffner@novacoxmail.com>
    Key fingerprint = AFAB CCEB 68C7 573F 5110 9285 1689 1942 CDEC BF99
uid Michael Heffner <mheffner@vt.edu>
uid Michael Heffner <mikeh@FreeBSD.org>
uid Michael Heffner <spock@techfour.net>
uid Michael Heffner (ACM sysadmin) <mheffner@acm.vt.edu>
sub 1024g/3FE83FB5 2001-02-02
```

D.3.130. Martin Heinen <mheinen@FreeBSD.org>

```
pub 1024D/116C5C85 2002-06-17 Martin Heinen <mheinen@freebsd.org>
    Key fingerprint = C898 3FCD EEA0 17ED BEA9 564D E5A6 AFF2 116C 5C85
uid Martin Heinen <martin@sumuk.de>
sub 1024g/EA67506B 2002-06-17
```

D.3.131. Niels Heinen <niels@FreeBSD.org>

```
pub 1024D/5FE39B80 2004-12-06 Niels Heinen <niels.heinen@ubizen.com>
    Key fingerprint = 75D8 4100 CF5B 3280 543F 930C 613E 71AA 5FE3 9B80
uid Niels Heinen <niels@defaced.be>
```

```
uid          Niels Heinen <niels@heinen.ws>
uid          Niels Heinen <niels@FreeBSD.org>
sub 2048g/057F4DA7 2004-12-06
```

D.3.132. Jaakko Heinonen <jh@FreeBSD.org>

```
pub 1024D/53CCB781 2009-10-01 [expires: 2014-09-30]
    Key fingerprint = 3AED A2B6 B63D D771 1AFD 25FA DFDF 5B89 53CC B781
uid          Jaakko Heinonen (FreeBSD) <jh@FreeBSD.org>
sub 4096g/BB97397E 2009-10-01 [expires: 2014-09-30]
```

D.3.133. Jason Helfman <jgh@FreeBSD.org>

```
pub 2048R/4150D3DC 2011-12-18 [expires: 2021-12-15]
    Key fingerprint = 8E0D C457 9A0F C91C 23F3 0454 2059 9A63 4150 D3DC
uid          Jason Helfman <jgh@FreeBSD.org>
sub 2048R/695B1B92 2011-12-18 [expires: 2021-12-15]
```

D.3.134. Guy Helmer <ghelmer@FreeBSD.org>

```
pub 2048R/8F1CEBC4 2012-05-22
    Key fingerprint = 483E 9E6C C644 2520 C9FE 4E87 9989 CCAF 8F1C EBC4
uid          Guy Helmer <guy.helmer@palisadesystems.com>
uid          Guy Helmer <guy.helmer@gmail.com>
uid          Guy Helmer <ghelmer@freebsd.org>
sub 2048R/2073E3F8 2012-05-22

pub 1024R/35F4ED2D 1997-01-26 Guy G. Helmer <ghelmer@freebsd.org>
    Key fingerprint = A2 59 4B 92 02 5B 9E B1 B9 4E 2E 03 29 D5 DC 3A
uid          Guy G. Helmer <ghelmer@cs.iastate.edu>
uid          Guy G. Helmer <ghelmer@palisadesys.com>
```

D.3.135. Maxime Henrion <mux@FreeBSD.org>

```
pub 1024D/881D4806 2003-01-09 Maxime Henrion <mux@FreeBSD.org>
    Key fingerprint = 81F1 BE2D 12F1 184A 77E4 ACD0 5563 7614 881D 4806
sub 2048g/D0B510C0 2003-01-09
```

D.3.136. Wen Heping <wen@FreeBSD.org>

```
pub 2048R/A03F07DA 2012-12-10
    Key fingerprint = 0258 F2C7 C123 E627 9E14 B4BA 270F 30AA A03F 07DA
uid          Wen Heping (wen) <wen@FreeBSD.org>
sub 2048R/CFC8D6A9 2012-12-10
```

D.3.137. Dennis Herrmann <dh@FreeBSD.org>

```
pub 4096R/F7CDCAA1 2012-08-26
    Key fingerprint = 0587 E730 68A6 2646 A991 505D CD9B 3A87 F7CD CAA1
uid Dennis 'dh' Herrmann (Everybody wants to go to heaven, but nobody wants to o
sub 4096R/0A6D554F 2012-08-26
```

D.3.138. Justin Hibbits <jhibbits@FreeBSD.org>

```
pub 2048R/37BE2DB9 2011-12-01
    Key fingerprint = 8A12 7064 4F3D 339A 191D AD52 30C7 858E 37BE 2DB9
uid Justin Hibbits <chmreedalf@gmail.com>
uid Justin Hibbits <jhibbits@freebsd.org>
uid Justin Hibbits <jrh29@alumni.cwru.edu>
sub 2048R/A8DA156F 2011-12-01
```

D.3.139. Peter Holm <pho@FreeBSD.org>

```
pub 1024D/CF244E81 2008-11-17
    Key fingerprint = BE9B 32D8 89F1 F285 00E4 E4C5 EF3F B4B5 CF24 4E81
uid Peter Holm <pho@FreeBSD.org>
sub 2048g/E20A409F 2008-11-17
```

D.3.140. Michael L. Hostbaek <mich@FreeBSD.org>

```
pub 1024D/0F55F6BE 2001-08-07 Michael L. Hostbaek <mich@freebsdcluster.org>
    Key fingerprint = 4D62 9396 B19F 38D3 5C99 1663 7B0A 5212 0F55 F6BE
uid Michael L. Hostbaek <mich@freebsdcluster.dk>
uid Michael L. Hostbaek <mich@icommerce-france.com>
uid Micahel L. Hostbaek <mich@freebsd.dk>
uid Michael L. Hostbaek <mich@the-lab.org>
uid Michael L. Hostbaek <mich@freebsd.org>
sub 1024g/8BE4E30F 2001-08-07
```

D.3.141. Po-Chuan Hsieh <sunpoet@FreeBSD.org>

```
pub 4096R/CC57E36B 2010-09-21
    Key fingerprint = 8AD8 68F2 7D2B 0A10 7E9B 8CC0 DC44 247E CC57 E36B
uid Po-Chuan Hsieh (FreeBSD) <sunpoet@FreeBSD.org>
uid Po-Chuan Hsieh (sunpoet) <sunpoet@sunpoet.net>
sub 4096R/ADE9E203 2010-09-21
```

D.3.142. Li-Wen Hsu <lwhsu@FreeBSD.org>

```

pub 1024D/2897B228 2005-01-16
    Key fingerprint = B6F7 170A 6DC6 5D1A BD4B D86A 416B 0E39 2897 B228
uid      Li-wen Hsu <lwhsu@lwhsu.org>
uid      Li-wen Hsu <lwhsu@lwhsu.ckefgisc.org>
uid      Li-wen Hsu <lwhsu@lwhsu.csie.net>
uid      Li-wen Hsu <lwhsu@ckefgisc.org>
uid      Li-wen Hsu <lwhsu@csie.nctu.edu.tw>
uid      Li-wen Hsu <lwhsu@ccca.nctu.edu.tw>
uid      Li-wen Hsu <lwhsu@iis.sinica.edu.tw>
uid      Li-wen Hsu <lwhsu@cs.nctu.edu.tw>
uid      Li-Wen Hsu <lwhsu@FreeBSD.org>
sub 2048g/16F82238 2005-01-16

```

D.3.143. Howard F. Hu <foxfair@FreeBSD.org>

```

pub 1024D/4E9BCA59 2003-09-01 Foxfair Hu <foxfair@FreeBSD.org>
    Key fingerprint = 280C A846 CA1B CAC9 DDCF F4CB D553 4BD5 4E9B CA59
uid      Foxfair Hu <foxfair@drago.fomokka.net>
uid      Howard Hu <howardhu@yahoo-inc.com>
sub 1024g/3356D8C1 2003-09-01

```

D.3.144. Chin-San Huang <chinsan@FreeBSD.org>

```

pub 1024D/350EECFA 2006-10-04
    Key fingerprint = 1C4D 0C9E 0E68 DB74 0688 CE43 D2A5 3F82 350E ECFA
uid      Chin-San Huang (lab) <chinsan@chinsan2.twbbs.org>
uid      Chin-San Huang (FreeBSD committer) <chinsan@FreeBSD.org>
uid      Chin-San Huang (Gmail) <chinsan.tw@gmail.com>
sub 2048g/35F75A30 2006-10-04

```

D.3.145. Davide Italiano <davide@FreeBSD.org>

```

pub 2048R/4CB47484 2012-01-17
    Key fingerprint = B5C9 77F5 1E67 D110 8D19 7587 EB95 EA82 4CB4 7484
uid      Davide Italiano <davide@FreeBSD.org>
sub 2048R/91F7443D 2012-01-17

```

D.3.146. Jordan K. Hubbard <jkh@FreeBSD.org>

```

pub 1024R/8E542D5D 1996-04-04 Jordan K. Hubbard <jkh@FreeBSD.org>
    Key fingerprint = 3C F2 27 7E 4A 6C 09 0A 4B C9 47 CD 4F 4D 0B 20

```

D.3.147. Konrad Jankowski <versus@FreeBSD.org>

```
pub 1024D/A01C218A 2008-10-28
    Key fingerprint = A805 21DC 859F E941 D2EA 9986 2264 8E5D A01C 218A
uid      Konrad Jankowski <versus@freebsd.org>
sub 2048g/56AE1959 2008-10-28
```

D.3.148. Weongyo Jeong <weongyo@FreeBSD.org>

```
pub 1024D/22354D7A 2007-12-28
    Key fingerprint = 138E 7115 A86F AA40 B509 5883 B387 DCE9 2235 4D7A
uid      Weongyo Jeong <weongyo.jeong@gmail.com>
uid      Weongyo Jeong <weongyo@freebsd.org>
sub 2048g/9AE6DAEE 2007-12-28
```

D.3.149. Peter Jeremy <peterj@FreeBSD.org>

```
pub 1024D/F00FB887 2005-10-20
    Key fingerprint = 0BF7 7A72 5894 EBE6 4F4D 7EEE FE8A 47BF F00F B887
uid      Peter Jeremy <peterjeremy@acm.org>
uid      [jpeg image of size 4413]
uid      Peter Jeremy <peter.jeremy@auug.org.au>
uid      Peter Jeremy <peterjeremy@optusnet.com.au>
uid      Peter Jeremy (preferred) <peter@rulingia.com>
uid      Peter Jeremy <peterj@freebsd.org>
sub 2048g/7E0B423B 2005-10-20
```

D.3.150. Tatuya JINMEI <jinmei@FreeBSD.org>

```
pub 1024D/ABA82228 2002-08-15
    Key fingerprint = BB70 3050 EE39 BE00 48BB A5F3 5892 F203 ABA8 2228
uid      JINMEI Tatuya <jinmei@FreeBSD.org>
uid      JINMEI Tatuya <jinmei@jinmei.org>
uid      JINMEI Tatuya (the KAME project) <jinmei@isl.rdc.toshiba.co.jp>
sub 1024g/8B43CF66 2002-08-15
```

D.3.151. Michael Johnson <ahze@FreeBSD.org>

```
pub 1024D/3C046FD6 2004-10-29 Michael Johnson (FreeBSD key) <ahze@FreeBSD.org>
    Key fingerprint = 363C 6ABA ED24 C23B 5F0C 3AB4 9F8B AA7D 3C04 6FD6
uid      Michael Johnson (pgp key) <ahze@ahze.net>
sub 2048g/FA334AE3 2004-10-29
```

D.3.152. Mark Johnston <markj@FreeBSD.org>

```
pub 2048R/80A62628 2012-12-19
    Key fingerprint = AFEF AD33 1C4E FFE5 141E 0157 05A4 DA8B 80A6 2628
uid                               Mark Johnston <markj@freebsd.org>
sub 2048R/47C7D3C2 2012-12-19
```

D.3.153. Trevor Johnson <trevor@FreeBSD.org>

```
pub 1024D/3A3EA137 2000-04-20 Trevor Johnson <trevor@jpj.net>
    Key fingerprint = 7ED1 5A92 76C1 FFCB E5E3 A998 F037 5A0B 3A3E A137
sub 1024g/46C24F1E 2000-04-20
```

D.3.154. Tom Judge <tj@FreeBSD.org>

```
pub 2048R/81E22216 2012-05-27 [expires: 2017-05-26]
    Key fingerprint = 8EF8 36C8 44A6 9576 6ADB EB0E 4252 33DC 81E2 2216
uid                               Tom Judge <tom@tomjudge.com>
uid                               Tom Judge <tjudge@sourcefire.com>
uid                               Tom Judge <tj@freebsd.org>
sub 2048R/2CA4AA0D 2012-05-27 [expires: 2017-05-26]
```

D.3.155. Alexander Kabaev <kan@FreeBSD.org>

```
pub 1024D/C9BE5D96 2002-07-01
    Key fingerprint = 7474 A847 DBF5 50A5 FC3E F223 43AC F58C C9BE 5D96
uid                               Alexander Kabaev <kabaev@gmail.com>
uid                               Alexander Kabaev (FreeBSD committer account ID) <kan@FreeBSD.ORG>
sub 1024g/534D9E06 2002-07-01
```

D.3.156. Benjamin Kaduk <bjk@FreeBSD.org>

```
pub 4096R/8302FE9F 2011-08-20 [expires: 2013-07-21]
    Key fingerprint = 9FD9 F966 D914 5101 BE59 FE13 2D29 EEED 8302 FE9F
uid                               Benjamin Kaduk <bjk@FreeBSD.org>
sub 4096R/28698ABE 2011-08-20 [expires: 2013-08-19]
```

D.3.157. Poul-Henning Kamp <phk@FreeBSD.org>

```
pub 1024R/0358FCBD 1995-08-01 Poul-Henning Kamp <phk@FreeBSD.org>
    Key fingerprint = A3 F3 88 28 2F 9B 99 A2 49 F4 E2 FA 5A 78 8B 3E
```

D.3.158. Sergey Kandaurov <pluknet@FreeBSD.org>

```
pub 2048R/10607419 2010-10-04
    Key fingerprint = 020B EC25 7E1F 8BC5 C42C 513B 3F4E 97BA 1060 7419
uid          Sergey Kandaurov (freebsd) <pluknet@freebsd.org>
uid          Sergey Kandaurov <pluknet@gmail.com>
sub 2048R/5711F73B 2010-10-04
```

D.3.159. Coleman Kane <cokane@FreeBSD.org>

```
pub 1024D/C5DAB797 2007-07-22
    Key fingerprint = FC09 F326 4318 E714 DE45 6CB0 70C4 B141 C5DA B797
uid          Coleman Kane (Personal PGP Key) <cokane@cokane.org>
uid          Coleman Kane (Personal PGP Key) <cokane@FreeBSD.org>
sub 2048g/5C680129 2007-07-22
```

D.3.160. Takenori KATO <kato@FreeBSD.org>

```
pub 4096R/3CF9ACE7 2012-10-02
    Key fingerprint = 5B72 AEF9 B2F9 069D 54FE CF60 444F 91C8 3CF9 ACE7
uid          KATO Takenori <kato@FreeBSD.org>
uid          KATO Takenori <kato@nendai.nagoya-u.ac.jp>
sub 4096R/1C593356 2012-10-02
```

D.3.161. Josef Karthauser <joe@FreeBSD.org>

```
pub 1024D/E6B15016 2000-10-19 Josef Karthauser <joe@FreeBSD.org>
    Key fingerprint = 7266 8EAF 82C2 D439 5642 AC26 5D52 1C8C E6B1 5016
uid          Josef Karthauser <joe@tao.org.uk>
uid          Josef Karthauser <joe@uk.FreeBSD.org>
uid          [revoked] Josef Karthauser <josef@bsd.i.com>
uid          [revoked] Josef Karthauser <joe@pavilion.net>
sub 2048g/1178B692 2000-10-19
```

D.3.162. Vinod Kashyap <vkashyap@FreeBSD.org>

```
pub 1024R/04FCCDD3 2004-02-19 Vinod Kashyap (gnupg key) <vkashyap@freebsd.org>
    Key fingerprint = 9B83 0B55 604F E491 B7D2 759D DF92 DAA0 04FC CDD3
```

D.3.163. Kris Kennaway <kris@FreeBSD.org>

```
pub 1024D/68E840A5 2000-01-14 Kris Kennaway <kris@citusc.usc.edu>
    Key fingerprint = E65D 0E7D 7E16 B212 1BD6 39EE 5ABC B405 68E8 40A5
uid          Kris Kennaway <kris@FreeBSD.org>
```

```
uid          Kris Kennaway <kris@obsecurity.org>
sub 2048g/03A41C45 2000-01-14 [expires: 2006-01-14]
```

D.3.164. Giorgos Keramidas <keramida@FreeBSD.org>

```
pub 1024D/318603B6 2001-09-21
   Key fingerprint = C1EB 0653 DB8B A557 3829 00F9 D60F 941A 3186 03B6
uid          Giorgos Keramidas <keramida@FreeBSD.org>
uid          Giorgos Keramidas <keramida@ceid.upatras.gr>
uid          Giorgos Keramidas <keramida@hellug.gr>
uid          Giorgos Keramidas <keramida@linux.gr>
uid          Giorgos Keramidas <gkeramidas@gmail.com>
sub 1024g/50FDBAD1 2001-09-21
```

D.3.165. Max Khon <fjoe@FreeBSD.org>

```
pub 1024D/6B87E212 2009-02-17
   Key fingerprint = 124D EC6C 6365 D41A 497A 9C3E FCF3 8708 6B87 E212
uid          Max Khon <fjoe@FreeBSD.org>
uid          Max Khon <fjoe@samodelkin.net>
sub 2048g/CB71491D 2009-02-17
```

D.3.166. Manolis Kiagias <manolis@FreeBSD.org>

```
pub 1024D/6E0FB494 2006-08-22
   Key fingerprint = F820 5AAF 7112 2CDD 23D8 3BDF 67F3 311A 6E0F B494
uid          Manolis Kiagias <manolis@FreeBSD.org>
uid          Manolis Kiagias <sonicy@otenet.gr>
uid          Manolis Kiagias (A.K.A. sonic, sonicy, sonic2000gr) <sonic@diktia.dyndns.org>
sub 2048g/EB94B411 2006-08-22
```

D.3.167. Jung-uk Kim <jkim@FreeBSD.org>

```
pub 2048R/D932A1CE 2012-11-19
   Key fingerprint = 2202 B5FB 78B7 A303 4919 B7C7 25E9 69B1 D932 A1CE
uid          Jung-uk Kim <jkim@FreeBSD.org>
sub 2048R/41858FC6 2012-11-19
```

D.3.168. Zack Kirsch <zack@FreeBSD.org>

```
pub 1024D/1A725562 2010-11-05 Zack Kirsch <zack@freebsd.org>
   Key fingerprint = A8CC AA5E FB47 A386 E757 A2B8 BDD2 0684 1A72 5562
sub 1024g/6BFE2C06 2010-11-05
```

D.3.169. Jakub Klama <jceel@FreeBSD.org>

```
pub 2048R/2AAEA67D 2011-09-27
    Key fingerprint = 40D6 097A 174F 511B 80EB F3A3 0946 4193 2AAE A67D
uid Jakub Klama <jceel@FreeBSD.org>
sub 2048R/5291BC4D 2011-09-27
```

D.3.170. Andreas Klemm <andreas@FreeBSD.org>

```
pub 1024D/6C6F6CBA 2001-01-06 Andreas Klemm <andreas.klemm@eu.didata.com>
    Key fingerprint = F028 D51A 0D42 DD67 4109 19A3 777A 3E94 6C6F 6CBA
uid Andreas Klemm <andreas@klemm.gtn.com>
uid Andreas Klemm <andreas@FreeBSD.org>
uid Andreas Klemm <andreas@apsfilter.org>
sub 2048g/FE23F866 2001-01-06
```

D.3.171. Johann Kois <jkois@FreeBSD.org>

```
pub 1024D/DD61C2D8 2004-06-27 Johann Kois <J.Kois@web.de>
    Key fingerprint = 8B70 03DB 3C45 E71D 0ED4 4825 FEB0 EBEF DD61 C2D8
uid Johann Kois <jkois@freebsd.org>
sub 1024g/568307CB 2004-06-27
```

D.3.172. Sergei Kolobov <sergei@FreeBSD.org>

```
pub 1024D/3BA53401 2003-10-10 Sergei Kolobov <sergei@FreeBSD.org>
    Key fingerprint = A2F4 5F34 0586 CC9C 493A 347C 14EC 6E69 3BA5 3401
uid Sergei Kolobov <sergei@kolobov.com>
sub 2048g/F8243671 2003-10-10
```

D.3.173. Maxim Konovalov <maxim@FreeBSD.org>

```
pub 1024D/2C172083 2002-05-21 Maxim Konovalov <maxim@FreeBSD.org>
    Key fingerprint = 6550 6C02 EFC2 50F1 B7A3 D694 ECF0 E90B 2C17 2083
uid Maxim Konovalov <maxim@macomnet.ru>
sub 1024g/F305DDCA 2002-05-21
```

D.3.174. Taras Korenko <taras@FreeBSD.org>

```
pub 1024D/8ACCC68B 2010-03-30
    Key fingerprint = 5128 2A8B 9BC1 A664 21E0 1E61 D838 54D3 8ACC C68B
uid Taras Korenko <taras@freebsd.org>
uid Taras Korenko <ds@ukrhub.net>
uid Taras Korenko <tarasishche@gmail.com>
```

sub 2048g/8D7CC0FA 2010-03-30 [expires: 2015-03-29]

D.3.175. Joseph Koshy <jkoshy@FreeBSD.org>

pub 1024D/D93798B6 2001-12-21 Joseph Koshy (FreeBSD) <jkoshy@freebsd.org>
 Key fingerprint = 0DE3 62F3 EF24 939F 62AA 2E3D ABB8 6ED3 D937 98B6
 sub 1024g/43FD68E9 2001-12-21

D.3.176. Wojciech A. Koszek <wkoszek@FreeBSD.org>

pub 1024D/C9F25145 2006-02-15
 Key fingerprint = 6E56 C571 9D33 D23E 9A61 8E50 623C AD62 C9F2 5145
 uid Wojciech A. Koszek <dunstan@FreeBSD.czyst.pl>
 uid Wojciech A. Koszek <wkoszek@FreeBSD.org>
 sub 4096g/3BBD20A5 2006-02-15

D.3.177. Alex Kozlov <ak@FreeBSD.org>

pub 2048R/0D1D29A0 2012-03-01 [expires: 2024-02-27]
 Key fingerprint = 7774 4FCF 6AC9 126B BD0E DBF3 5EBF 4968 0D1D 29A0
 uid Alex Kozlov <ak@freebsd.org>
 sub 2048R/2DD82C65 2012-03-01 [expires: 2024-02-27]

D.3.178. Steven Kreuzer <skreuzer@FreeBSD.org>

pub 1024D/E0D6F907 2009-03-16 [expires: 2013-04-25]
 Key fingerprint = 8D8F 14D6 ED9F 6BD0 7756 7A46 66BA B4B6 E0D6 F907
 uid Steven Kreuzer <skreuzer@exit2shell.com>
 uid Steven Kreuzer <skreuzer@freebsd.org>

D.3.179. Gábor Kövesdán <gabor@FreeBSD.org>

pub 1024D/2373A6B1 2006-12-05
 Key fingerprint = A42A 10D6 834B BEC0 26F0 29B1 902D D04F 2373 A6B1
 uid Gabor Kovesdan <gabor@FreeBSD.org>
 sub 2048g/92B0A104 2006-12-05

D.3.180. Ana Kukec <anchie@FreeBSD.org>

pub 2048R/510D23BB 2010-04-18
 Key fingerprint = 0A9B 0ABB 0E1C B5A4 3408 398F 778A C3B4 510D 23BB
 uid Ana Kukec <anchie@FreeBSD.org>

sub 2048R/699E4DDA 2010-04-18

D.3.181. Roman Kurakin <rik@FreeBSD.org>

pub 1024D/C8550F4C 2005-12-16 [expires: 2008-12-15]
 Key fingerprint = 25BB 789A 6E07 E654 8E59 0FA9 42B1 937C C855 0F4C
 uid Roman Kurakin <rik@FreeBSD.org>
 sub 2048g/D15F2AB6 2005-12-16 [expires: 2008-12-15]

D.3.182. Hideyuki KURASHINA <rushani@FreeBSD.org>

pub 1024D/439ADC57 2002-03-22 Hideyuki KURASHINA <rushani@bl.mmtr.or.jp>
 Key fingerprint = A052 6F98 6146 6FE3 91E2 DA6B F2FA 2088 439A DC57
 uid Hideyuki KURASHINA <rushani@FreeBSD.org>
 uid Hideyuki KURASHINA <rushani@jp.FreeBSD.org>
 sub 1024g/64764D16 2002-03-22

D.3.183. Jun Kuriyama <kuriyama@FreeBSD.org>

pub 1024D/FE3B59CD 1998-11-23 Jun Kuriyama <kuriyama@imgsrc.co.jp>
 Key fingerprint = 5219 55CE AC84 C296 3A3B B076 EE3C 4DBB FE3B 59CD
 uid Jun Kuriyama <kuriyama@FreeBSD.org>
 uid Jun Kuriyama <kuriyama@jp.FreeBSD.org>
 sub 2048g/1CF20D27 1998-11-23

D.3.184. René Ladan <rene@FreeBSD.org>

pub 4096R/0A3789B7 2012-11-18
 Key fingerprint = 101A 716B 162B 00E5 5BED EA05 ADBB F861 0A37 89B7
 uid René Ladan <rene@freebsd.org>
 sub 4096R/B67184C6 2012-11-18

D.3.185. Julien Laffaye <jlaffaye@FreeBSD.org>

pub 2048R/6AEBE420 2011-06-06
 Key fingerprint = 031A B449 B383 5C3B B618 E2F4 BAD0 0F0E 6AEB E420
 uid Julien Laffaye <jlaffaye@FreeBSD.org>
 sub 2048R/538B8D5B 2011-06-06

D.3.186. Clement Laforet <clement@FreeBSD.org>

```
pub 1024D/0723BA1D 2003-12-13 Clement Laforet (FreeBSD committer address) <clement@FreeBSD.org>
    Key fingerprint = 3638 4B14 8463 A67B DC7E 641C B118 5F8F 0723 BA1D
uid                                Clement Laforet <sheepkiller@cultdeadsheep.org>
uid                                Clement Laforet <clement.laforet@cotds.org>
sub 2048g/23D57658 2003-12-13
```

D.3.187. Max Laier <mlaier@FreeBSD.org>

```
pub 1024D/3EB6046D 2004-02-09
    Key fingerprint = 917E 7F25 E90F 77A4 F746 2E8D 5F2C 84A1 3EB6 046D
uid                                Max Laier <max@love2party.net>
uid                                Max Laier <max.laier@ira.uka.de>
uid                                Max Laier <mlaier@freebsd.org>
uid                                Max Laier <max.laier@tm.uka.de>
sub 4096g/EDD08B9B 2005-06-28
```

D.3.188. Erwin Lansing <erwin@FreeBSD.org>

```
pub 1024D/15256990 1998-07-03
    Key fingerprint = FB58 9797 299A F18E 2D3E 73D6 AB2F 5A5B 1525 6990
uid                                Erwin Lansing <erwin@lansing.dk>
uid                                Erwin Lansing <erwin@FreeBSD.org>
uid                                Erwin Lansing <erwin@droso.dk>
uid                                Erwin Lansing <erwin@droso.org>
uid                                Erwin Lansing <erwin@aauug.dk>
sub 2048g/7C64013D 1998-07-03
```

D.3.189. Ganael Laplanche <martymac@FreeBSD.org>

```
pub 1024D/10B87391 2006-01-13
    Key fingerprint = D59D 984D 8988 7BB9 DA37 BA77 757E D5F0 10B8 7391
uid                                Ganael LAPLANCHE <ganael.laplanche@martymac.org>
uid                                Ganael LAPLANCHE <martymac@martymac.com>
uid                                Ganael LAPLANCHE <ganael.laplanche@martymac.com>
uid                                Ganael LAPLANCHE <martymac@martymac.org>
uid                                Ganael LAPLANCHE <martymac@pasteur.fr>
uid                                Ganael LAPLANCHE <ganael.laplanche@pasteur.fr>
uid                                Ganael LAPLANCHE <martymac@FreeBSD.org>
sub 2048g/D65069D5 2006-01-13
```

D.3.190. Greg Larkin <glarkin@FreeBSD.org>

```
pub 1024D/1C940290 2003-10-09
    Key fingerprint = 8A4A 80AA F26C 8C2C D01B 94C6 D2C4 68B8 1C94 0290
uid      Greg Larkin (The FreeBSD Project) <glarkin@FreeBSD.org>
uid      Gregory C. Larkin (SourceHosting.Net, LLC) <glarkin@sourcehosting.net>
uid      [jpeg image of size 6695]
sub 2048g/47674316 2003-10-09
```

D.3.191. Frank J. Laszlo <laszlof@FreeBSD.org>

```
pub 4096R/012360EC 2006-11-06 [expires: 2011-11-05]
    Key fingerprint = 3D93 21DB B5CC 1339 E4B4 1BC4 AD50 C17C 0123 60EC
uid      Frank J. Laszlo <laszlof@FreeBSD.org>
```

D.3.192. Dru Lavigne <dru@FreeBSD.org>

```
pub 1024D/C6AA2E94 2013-01-22
    Key fingerprint = 6CC4 2180 F27C 29B6 5A9C EC0D A454 DC05 C6AA 2E94
uid      Dru Lavigne <dru@freebsd.org>
sub 1024g/7FAC82EA 2013-01-22
```

D.3.193. Sam Lawrance <lawrance@FreeBSD.org>

```
pub 1024D/32708C59 2003-08-14
    Key fingerprint = 1056 2A02 5247 64D4 538D 6975 8851 7134 3270 8C59
uid      Sam Lawrance <lawrance@FreeBSD.org>
uid      Sam Lawrance <boris@brooknet.com.au>
sub 2048g/0F9CCF92 2003-08-14
```

D.3.194. Nate Lawson <njl@FreeBSD.org>

```
pub 1024D/60E5AC11 2007-02-07
    Key fingerprint = 18E2 7E5A FD6A 199B B08B E9FB 73C8 DB67 60E5 AC11
uid      Nate Lawson <nate@root.org>
sub 2048g/CDBC7E1B 2007-02-07
```

D.3.195. Jeremie Le Hen <jlh@FreeBSD.org>

```
pub 2048D/8BF6CF92 2012-04-18
    Key fingerprint = 66C9 B361 16CA BFF6 5C07 DA0A 28DE 3702 8BF6 CF92
uid      Jeremie Le Hen <jeremie@le-hen.org>
uid      Jeremie Le Hen <jeremie@lehen.org>
uid      Jeremie Le Hen <ttz@chchile.org>
```

```
uid          Jeremie Le Hen <jlh@FreeBSD.org>
sub 2048g/045479A3 2012-04-18
```

D.3.196. Yen-Ming Lee <leeym@FreeBSD.org>

```
pub 1024D/93FA8BD6 2007-05-21
   Key fingerprint = DEC4 6E7F 69C0 4AC3 21ED EE65 6C0E 9257 93FA 8BD6
uid          Yen-Ming Lee <leeym@leeym.com>
sub 2048g/899A3931 2007-05-21
```

D.3.197. Sam Leffler <sam@FreeBSD.org>

```
pub 1024D/BD147743 2005-03-28
   Key fingerprint = F618 F2FC 176B D201 D91C 67C6 2E33 A957 BD14 7743
uid          Samuel J. Leffler <sam@freebsd.org>
sub 2048g/8BA91D05 2005-03-28
```

D.3.198. Jean-Yves Lefort <jylefort@FreeBSD.org>

```
pub 1024D/A3B8006A 2002-09-07
   Key fingerprint = CC99 D1B0 8E44 293D 32F7 D92E CB30 FB51 A3B8 006A
uid          Jean-Yves Lefort <jylefort@FreeBSD.org>
uid          Jean-Yves Lefort <jylefort@brutele.be>
sub 4096g/C9271AFC 2002-09-07
```

D.3.199. Alexander Leidinger <netchild@FreeBSD.org>

```
pub 1024D/72077137 2002-01-31
   Key fingerprint = AA3A 8F69 B214 6BBD 5E73 C9A0 C604 3C56 7207 7137
uid          Alexander Leidinger <netchild@FreeBSD.org>
uid          [jpeg image of size 19667]
sub 2048g/8C9828D3 2002-01-31
```

D.3.200. Andrey V. Elsukov <ae@FreeBSD.org>

```
pub 2048R/10C8A17A 2010-05-29
   Key fingerprint = E659 1E1B 41DA 1516 F0C9 BC00 01C5 EA04 10C8 A17A
uid          Andrey V. Elsukov <ae@freebsd.org>
uid          Andrey V. Elsukov <bu7cher@yandex.ru>
sub 2048R/0F6D64C5 2010-05-29
```

D.3.201. Dejan Lesjak <lesi@FreeBSD.org>

```
pub 1024D/96C5221F 2004-08-18 Dejan Lesjak <lesi@FreeBSD.org>
    Key fingerprint = 2C5C 02EA 1060 1D6D 9982 38C0 1DA7 DBC4 96C5 221F
uid                               Dejan Lesjak <dejan.lesjak@ijs.si>
sub 1024g/E0A69278 2004-08-18
```

D.3.202. Achim Leubner <achim@FreeBSD.org>

```
pub 2048R/2E15B3C1 2013-01-22
    Key fingerprint = 2A48 0317 D477 2A07 2AD9 CF1C 7C1D 832E 2E15 B3C1
uid                               Achim Leubner <achim@freebsd.org>
sub 2048R/E275EF01 2013-01-22
```

D.3.203. Chuck Lever <cel@FreeBSD.org>

```
pub 1024D/8FFC2B87 2006-02-13
    Key fingerprint = 6872 923F 5012 F88B 394C 2F69 37B4 8171 8FFC 2B87
uid                               Charles E. Lever <cel@freebsd.org>
sub 2048g/9BCE0459 2006-02-13
```

D.3.204. Greg Lewis <glewis@FreeBSD.org>

```
pub 1024D/1BB6D9E0 2002-03-05 Greg Lewis (FreeBSD) <glewis@FreeBSD.org>
    Key fingerprint = 2410 DA6D 5A3C D801 65FE C8DB DEEA 9923 1BB6 D9E0
uid                               Greg Lewis <glewis@eyesbeyond.com>
sub 2048g/45E67D60 2002-03-05
```

D.3.205. Qing Li <qingli@FreeBSD.org>

```
pub 2048R/A3CA4C13 2013-06-12 [expires: 2017-06-12]
    Key fingerprint = E37B CB18 35D1 F01B 7D7B 1000 0EAF 4BEA A3CA 4C13
uid                               Qing Li <qingli@freebsd.org>
sub 2048R/EF3A9370 2013-06-12 [expires: 2017-06-12]
```

D.3.206. Xin Li <delphij@FreeBSD.org>

```
pub 1024D/CAEEB8C0 2004-01-28
    Key fingerprint = 43B8 B703 B8DD 0231 B333 DC28 39FB 93A0 CAEE B8C0
uid                               Xin LI <delphij@FreeBSD.org>
uid                               Xin LI <delphij@frontfree.net>
uid                               Xin LI <delphij@delphij.net>
uid                               Xin LI <delphij@geekcn.org>
```

```

pub 1024D/42EA8A4B 2006-01-27 [expired: 2008-01-01]
    Key fingerprint = F19C 2616 FA97 9C13 2581 C6F3 85C5 1CCE 42EA 8A4B
uid      Xin LI <delphij@geekcn.org>
uid      Xin LI <delphij@FreeBSD.org>
uid      Xin LI <delphij@delphij.net>

pub 1024D/18EDEBA0 2008-01-02 [expired: 2010-01-02]
    Key fingerprint = 79A6 CF42 F917 DDCA F1C2 C926 8BEB DB04 18ED EBA0
uid      Xin LI <delphij@geekcn.org>
uid      Xin LI <delphij@FreeBSD.org>
uid      Xin LI <delphij@delphij.net>

pub 2048R/3FCA37C1 2010-01-10 [expired: 2012-01-10]
    Key fingerprint = 27EA 5D6C 9398 BA7F B205 8F70 04CE F812 3FCA 37C1
uid      Xin LI <delphij@delphij.net>
uid      Xin LI <delphij@gmail.com>
uid      Xin LI <delphij@geekcn.org>
uid      Xin LI <delphij@FreeBSD.org>

pub 4096R/2E54AB2C 2011-12-05
    Key fingerprint = D95C D3C3 8FA8 25C2 C62B 9FEA 0887 6D93 2E54 AB2C
uid      Xin Li <delphij@geekcn.org>
uid      Xin Li <delphij@delphij.net>
uid      Xin Li <delphij@FreeBSD.org>
sub 4096R/7832B740 2011-12-05
sub 2048R/BC50FBB3 2011-12-05 [expires: 2013-12-05]
sub 2048R/C894647D 2011-12-05 [expires: 2013-12-05]

```

D.3.207. Tai-hwa Liang <avatar@FreeBSD.org>

```

pub 1024R/F4013AB1 1998-05-13 Tai-hwa Liang <avatar@FreeBSD.org>
    Key fingerprint = 5B 05 1D 37 7F 35 31 4E 5D 38 BD 07 10 32 B9 D0
uid      Tai-hwa Liang <avatar@mmlab.cse.yzu.edu.tw>

```

D.3.208. Ying-Chieh Liao <ijliao@FreeBSD.org>

```

pub 1024D/11C02382 2001-01-09 Ying-Chieh Liao <ijliao@CCCA.NCTU.edu.tw>
    Key fingerprint = 4E98 55CC 2866 7A90 EFD7 9DA5 ACC6 0165 11C0 2382
uid      Ying-Chieh Liao <ijliao@FreeBSD.org>
uid      Ying-Chieh Liao <ijliao@csie.nctu.edu.tw>
uid      Ying-Chieh Liao <ijliao@dragon2.net>
uid      Ying-Chieh Liao <ijliao@tw.FreeBSD.org>
sub 4096g/C1E16E89 2001-01-09

```

D.3.209. Ulf Lilleengen <lulf@FreeBSD.org>

```

pub 1024D/ADE1B837 2009-08-19 [expires: 2014-08-18]
    Key fingerprint = 3822 B4E6 6D1C 6F71 4AA8 7A27 ADDF C400 ADE1 B837
uid          Ulf Lilleengen <lulf.lilleengen@gmail.com>
uid          Ulf Lilleengen <lulf@pvv.ntnu.no>
uid          Ulf Lilleengen <lulf@stud.ntnu.no>
uid          Ulf Lilleengen <lulf@FreeBSD.org>
uid          Ulf Lilleengen <lulf@idi.ntnu.no>
sub 2048g/B5409122 2009-08-19 [expires: 2014-08-18]

```

D.3.210. Clive Lin <clive@FreeBSD.org>

```

pub 1024D/A008C03E 2001-07-30 Clive Lin <clive@tongi.org>
    Key fingerprint = FA3F 20B6 A77A 6CEC 1856 09B0 7455 2805 A008 C03E
uid          Clive Lin <clive@CirX.ORG>
uid          Clive Lin <clive@FreeBSD.org>
sub 1024g/03C2DC87 2001-07-30 [expires: 2005-08-25]

```

D.3.211. Po-Chien Lin <pclin@FreeBSD.org>

```

pub 4096R/865C427F 2013-02-05
    Key fingerprint = CF3B AB13 4C94 6388 B047 B599 8B28 1692 865C 427F
uid          Po-Chien Lin <pclin@FreeBSD.org>
uid          Po-Chien Lin <linpc@cs.nctu.edu.tw>
sub 4096R/F31280BA 2013-02-05

```

D.3.212. Yi-Jheng Lin <yzlin@FreeBSD.org>

```

pub 2048R/A34C6A8A 2009-07-20
    Key fingerprint = 7E3A E981 BB7C 5D73 9534 ED39 0222 04D3 A34C 6A8A
uid          Yi-Jheng Lin (FreeBSD) <yzlin@FreeBSD.org>
sub 2048R/B4D776FE 2009-07-20

```

D.3.213. Mark Linimon <linimon@FreeBSD.org>

```

pub 1024D/84C83473 2003-10-09
    Key fingerprint = 8D43 1B55 D127 0BFC 842E 1C96 803C 5A34 84C8 3473
uid          Mark Linimon <linimon@FreeBSD.org>
uid          Mark Linimon <linimon@lonesome.com>
sub 1024g/24BFF840 2003-10-09

```

D.3.214. Tilman Keskinöz <arved@FreeBSD.org>

```
pub 1024D/807AC53A 2002-06-03 [expires: 2013-09-07]
    Key fingerprint = A92F 344F 31A8 B8DE DDFA 7FB4 7C22 C39F 807A C53A
uid      Tilman Keskinöz <arved@arved.at>
uid      Tilman Keskinöz <arved@FreeBSD.org>
sub 1024g/FA351986 2002-06-03 [expires: 2013-09-07]
```

D.3.215. Dryice Liu <dryice@FreeBSD.org>

```
pub 1024D/77B67874 2005-01-28
    Key fingerprint = 8D7C F82D D28D 07E5 EF7F CD25 6B5B 78A8 77B6 7874
uid      Dryice Dong Liu (Dryice) <dryice@FreeBSD.org>
uid      Dryice Dong Liu (Dryice) <dryice@liu.com.cn>
uid      Dryice Dong Liu (Dryice) <dryice@hotpop.com>
uid      Dryice Dong Liu (Dryice) <dryiceliu@gmail.com>
uid      Dryice Dong Liu (Dryice) <dryice@dryice.name>
sub 2048g/ECFA49E4 2005-01-28
```

D.3.216. Tong Liu <nemoliu@FreeBSD.org>

```
pub 1024D/ECC7C907 2007-07-10
    Key fingerprint = B62E 3109 896B B283 E2FA 60FE A1BA F92E ECC7 C907
uid      Tong LIU <nemoliu@FreeBSD.org>
sub 4096g/B6D7B15D 2007-07-10
```

D.3.217. Zachary Loafman <zml@FreeBSD.org>

```
pub 1024D/4D65492D 2009-05-26
    Key fingerprint = E513 4AE9 5D6D 8BF9 1CD3 4389 4860 D79B 4D65 492D
uid      Zachary Loafman <zml@FreeBSD.org>
sub 2048g/1AD659F0 2009-05-26
```

D.3.218. Juergen Lock <nox@FreeBSD.org>

```
pub 1024D/1B6BFBFD 2006-12-22
    Key fingerprint = 33A7 7FAE 51AF 00BC F0D3 ECCE FAFD 34C1 1B6B FBFD
uid      Juergen Lock <nox@FreeBSD.org>
sub 2048g/251229D1 2006-12-22
```

D.3.219. Remko Lodder <remko@FreeBSD.org>

```
pub 4096R/3F774079 2012-11-11 [expires: 2016-11-11]
    Key fingerprint = 7EE4 C4AF DCA3 E0B4 479B A344 7135 8ED6 3F77 4079
uid                                Remko Lodder <remko@FreeBSD.org>
sub 4096R/59F38CB0 2012-11-11 [expires: 2016-11-11]
```

D.3.220. Alexander Logvinov <avl@FreeBSD.org>

```
pub 1024D/1C47D5C0 2009-05-28
    Key fingerprint = 8B5F 880A 382B 075E E707 9DB2 E135 4176 1C47 D5C0
uid                                Alexander Logvinov <alexander@logvinov.com>
uid                                Alexander Logvinov (FreeBSD Ports Committer) <avl@FreeBSD.org>
uid                                Alexander Logvinov <ports@logvinov.com>
uid                                Alexander Logvinov <logvinov@gmail.com>
uid                                Alexander Logvinov <logvinov@yandex.ru>
sub 2048g/60BDD4BB 2009-05-28
```

D.3.221. Isabell Long <issyl0@FreeBSD.org>

```
pub 4096R/EB83C2BD 2009-09-26
    Key fingerprint = D55A 42E7 0974 EFD9 3939 56B9 6E6B E425 EB83 C2BD
uid                                Isabell Long <isabell@issyl0.co.uk>
uid                                Isabell Long <me@issyl0.co.uk>
uid                                Isabell Long <isabell1121@gmail.com>
uid                                Isabell Long (BitFolk Ltd.) <isabell@bitfolk.com>
uid                                Isabell Long (College) <IL18685@woking.ac.uk>
uid                                Isabell Long (The Open University) <il948@my.open.ac.uk>
uid                                Isabell Long (Mailing lists address.) <lists@issyl0.co.uk>
uid                                Isabell Long (YRS) <isabell@youngwiredstate.org>
uid                                Isabell Long (FreeBSD) <issyl0@FreeBSD.org>
```

D.3.222. Scott Long <scottl@FreeBSD.org>

```
pub 1024D/017C5EBF 2003-01-18 Scott A. Long (This is my official FreeBSD key) <scottl@freebsd.org>
    Key fingerprint = 34EA BD06 44F7 F8C3 22BC B52C 1D3A F6D1 017C 5EBF
sub 1024g/F61C8F91 2003-01-18
```

D.3.223. Rick Macklem <rmacklem@FreeBSD.org>

```
pub 1024D/7FB9C5F1 2009-04-05
    Key fingerprint = B9EA 767A F6F3 3786 E0C7 434A 05C6 70D6 7FB9 C5F1
uid                                Rick Macklem <rmacklem@freebsd.org>
sub 1024g/D0B20E8A 2009-04-05
```

D.3.224. Bruce A. Mah <bmah@FreeBSD.org>

```
pub 1024D/5BA052C3 1997-12-08
    Key fingerprint = F829 B805 207D 14C7 7197 7832 D8CA 3171 5BA0 52C3
uid Bruce A. Mah <bmah@acm.org>
uid Bruce A. Mah <bmah@ca.sandia.gov>
uid Bruce A. Mah <bmah@ieee.org>
uid Bruce A. Mah <bmah@cisco.com>
uid Bruce A. Mah <bmah@employees.org>
uid Bruce A. Mah <bmah@freebsd.org>
uid Bruce A. Mah <bmah@packetdesign.com>
uid Bruce A. Mah <bmah@kitchenlab.org>
sub 2048g/B4E60EA1 1997-12-08
```

D.3.225. Ruslan Makhmatkhanov <rm@FreeBSD.org>

```
pub 2048R/F60D756F 2011-11-10
    Key fingerprint = 9D18 8A88 304C B78B 8003 0379 4574 0BAF F60D 756F
uid Ruslan Makhmatkhanov <rm@FreeBSD.org>
sub 2048R/B658C269 2011-11-10
```

D.3.226. Mike Makonnen <mtm@FreeBSD.org>

```
pub 1024D/7CD41F55 2004-02-06 Michael Telahun Makonnen <mtm@FreeBSD.Org>
    Key fingerprint = AC7B 5672 2D11 F4D0 EBF8 5279 5359 2B82 7CD4 1F55
uid Michael Telahun Makonnen <mtm@tmsa-inc.com>
uid Mike Makonnen <mtm@identd.net>
uid Michael Telahun Makonnen <mtm@acs-et.com>
sub 2048g/E7DC936B 2004-02-06
```

D.3.227. David Malone <dwmalone@FreeBSD.org>

```
pub 512/40378991 1994/04/21 David Malone <dwmalone@maths.tcd.ie>
    Key fingerprint = 86 A7 F4 86 39 2C 47 2C C1 C2 35 78 8E 2F B8 F5
```

D.3.228. Dmitry Marakasov <amdmi3@FreeBSD.org>

```
pub 1024D/F9D2F77D 2008-06-15 [expires: 2010-06-15]
    Key fingerprint = 55B5 0596 FF1E 8D84 5F56 9510 D35A 80DD F9D2 F77D
uid Dmitry Marakasov <amdmi3@amdmi3.ru>
uid Dmitry Marakasov <amdmi3@FreeBSD.org>
sub 2048g/2042CDD8 2008-06-15
```

D.3.229. John Marino <marino@FreeBSD.org>

```
pub 2048R/A0AE6229 2011-07-19
    Key fingerprint = EE48 4F90 C861 3A5F E39E AB9E 33CF 4190 A0AE 6229
uid      John Marino (DragonFly) <draco@marino.st>
uid      John R. Marino <john.secure@marino.st>
uid      John Marino (NetBSD) <marino@netbsd.org>
sub 2048R/71D9FB68 2011-07-19
```

D.3.230. Koop Mast <kwm@FreeBSD.org>

```
pub 1024D/F95426DA 2004-09-10 Koop Mast <kwm@rainbow-runner.nl>
    Key fingerprint = C66F 1835 0548 3440 8576 0FFE 6879 B7CD F954 26DA
uid      Koop Mast <kwm@FreeBSD.org>
sub 1024g/A782EEDD 2004-09-10
```

D.3.231. Ed Maste <emaste@FreeBSD.org>

```
pub 2048R/50A17BF4 2012-12-18
    Key fingerprint = 0C08 ECC9 3A0A 8500 AB95 B553 49C4 7851 50A1 7BF4
uid      Ed Maste <emaste@freebsd.org>
sub 2048R/08FA5F72 2012-12-18
```

D.3.232. Cherry G. Mathew <cherry@FreeBSD.org>

```
pub 2048R/2D066FE1 2007-05-22
    Key fingerprint = FBF1 89FF 81BB E1C7 6C1B 378D 3438 20E9 2D06 6FE1
uid      Cherry G. Mathew (FreeBSD email) <cherry@FreeBSD.org>
uid      "Cherry G. Mathew" (NetBSD email) <cherry@NetBSD.org>
sub 2048R/7B2C4166 2007-05-22
```

D.3.233. Makoto Matsushita <matusita@FreeBSD.org>

```
pub 1024D/20544576 1999-04-18
    Key fingerprint = 71B6 13BF B262 2DD8 2B7C 6CD0 EB2D 4147 2054 4576
uid      Makoto Matsushita <matusita@matatabi.or.jp>
uid      Makoto Matsushita <matusita@FreeBSD.org>
uid      Makoto Matsushita <matusita@jp.FreeBSD.ORG>
uid      Makoto Matsushita <matusita@ist.osaka-u.ac.jp>
sub 1024g/F1F3C94D 1999-04-18
```

D.3.234. Martin Matuska <mm@FreeBSD.org>

```
pub 1024D/4261B0D1 2007-02-05
    Key fingerprint = 17C4 3F32 B3DE 3ED7 E84E 5592 A76B 8B03 4261 B0D1
uid      Martin Matuska <martin@matuska.org>
uid      Martin Matuska <mm@FreeBSD.org>
uid      Martin Matuska <martin.matuska@wu-wien.ac.at>
sub 2048g/3AC9A5A6 2007-02-05
```

D.3.235. Sergey Matveychuk <sem@FreeBSD.org>

```
pub 1024D/B71F605D 1999-10-13
    Key fingerprint = 4704 F374 DB28 BEC6 51C8 1322 4DC9 4BD8 B71F 605D
uid      Sergey Matveychuk <sem@FreeBSD.org>
uid      Sergey Matveychuk <sem@ciam.ru>
uid      Sergey Matveychuk <sem@core.inec.ru>
sub 2048g/DEAF9D91 1999-10-13
```

D.3.236. Tom McLaughlin <tmclaugh@FreeBSD.org>

```
pub 1024D/E2F7B3D8 2005-05-24
    Key fingerprint = 7692 B222 8D23 CF94 1993 0138 E339 E225 E2F7 B3D8
uid      Tom McLaughlin (Personal email address) <tmclaugh@sdf.lonestar.org>
uid      Tom McLaughlin (Work email address) <tmclaughlin@meditech.com>
uid      Tom McLaughlin (FreeBSD email address) <tmclaugh@FreeBSD.org>
sub 2048g/16838F62 2005-05-24
```

D.3.237. Jean Milanez Melo <jmelo@FreeBSD.org>

```
pub 1024D/AA5114BF 2006-03-03
    Key fingerprint = 826D C2AA 6CF2 E29A EBE7 4776 D38A AB83 AA51 14BF
uid      Jean Milanez Melo <jmelo@FreeBSD.org>
uid      Jean Milanez Melo <jmelo@freebsdbrasil.com.br>
sub 4096g/E9E1CBD9 2006-03-03
```

D.3.238. Kenneth D. Merry <ken@FreeBSD.org>

```
pub 1024D/54C745B5 2000-05-15 Kenneth D. Merry <ken@FreeBSD.org>
    Key fingerprint = D25E EBC5 F17A 9E52 84B4 BF14 9248 F0DA 54C7 45B5
uid      Kenneth D. Merry <ken@kdm.org>
sub 2048g/89D0F797 2000-05-15

pub 1024R/2FA0A505 1995-10-30 Kenneth D. Merry <ken@plutotech.com>
    Key fingerprint = FD FA 85 85 95 C4 8E E8 98 1A CA 18 56 F0 00 1F
```

D.3.239. Dirk Meyer <dinoex@FreeBSD.org>

```
pub 1024R/331CDA5D 1995-06-04 Dirk Meyer <dinoex@FreeBSD.org>
    Key fingerprint = 44 16 EC 0A D3 3A 4F 28 8A 8A 47 93 F1 CF 2F 12
uid                                Dirk Meyer <dirk.meyer@dinoex.sub.org>
uid                                Dirk Meyer <dirk.meyer@guug.de>
```

D.3.240. Yoshiro Sanpei MIHIRA <sanpei@FreeBSD.org>

```
pub 1024R/391C5D69 1996-11-21 sanpei@SEAPLE.ICC.NE.JP
    Key fingerprint = EC 04 30 24 B0 6C 1E 63 5F 5D 25 59 3E 83 64 51
uid                                MIHIRA Yoshiro <sanpei@sanpei.org>
uid                                Yoshiro MIHIRA <sanpei@FreeBSD.org>
uid                                MIHIRA Yoshiro <sanpei@yy.cs.keio.ac.jp>
uid                                MIHIRA Yoshiro <sanpei@cc.keio.ac.jp>
uid                                MIHIRA Yoshiro <sanpei@educ.cc.keio.ac.jp>
uid                                MIHIRA Yoshiro <sanpei@st.keio.ac.jp>
```

D.3.241. Robert Millan <rmh@FreeBSD.org>

```
pub 4096R/DEA2C38E 2009-08-14
    Key fingerprint = A537 F029 AAAE 0E9C 39A7 C22C BB9D 98D9 DEA2 C38E
uid                                Robert Millan <rmh@debian.org>
uid                                Robert Millan <rmh@freebsd.org>
uid                                Robert Millan <rmh@gnu.org>
sub 4096R/65A0A9CE 2009-08-14
sub 4096R/41F37946 2009-08-14
```

D.3.242. Stephen Montgomery-Smith <stephen@FreeBSD.org>

```
pub 2048R/9A92D807 2011-06-14
    Key fingerprint = 2B61 D82E 168E F08B 6E08 712E 2DF1 2BD1 9A92 D807
uid                                Stephen Montgomery-Smith <stephen@freebsd.org>
sub 2048R/A4BA6560 2011-06-14
```

D.3.243. Marcel Moolenaar <marcel@FreeBSD.org>

```
pub 1024D/61EE89F6 2002-02-09 Marcel Moolenaar <marcel@xcllnt.net>
    Key fingerprint = 68BB E2B7 49AA FF69 CA3A DF71 A605 A52D 61EE 89F6
sub 1024g/6EAAB456 2002-02-09
```

D.3.244. Kris Moore <kmoore@FreeBSD.org>

```
pub 1024D/6294612C 2009-05-26
    Key fingerprint = 8B70 9876 346F 1F97 5687 6950 4C92 D789 6294 612C
uid          Kris Moore <kmoore@freebsd.org>
sub 2048g/A7FFE8FB 2009-05-26
```

D.3.245. Dmitry Morozovsky <marck@FreeBSD.org>

```
pub 1024D/6B691B03 2001-07-20
    Key fingerprint = 39AC E336 F03D C0F8 5305 B725 85D4 5045 6B69 1B03
uid          Dmitry Morozovsky <marck@rinet.ru>
uid          Dmitry Morozovsky <marck@FreeBSD.org>
sub 2048g/44D656F8 2001-07-20
```

D.3.246. Alexander Motin <mav@FreeBSD.org>

```
pub 1024D/0577BACA 2007-04-20 [expires: 2012-04-18]
    Key fingerprint = 0E84 B263 E97D 3E48 161B 98A2 D240 A09E 0577 BACA
uid          Alexander Motin <mav@freebsd.org>
uid          Alexander Motin <mav@mavhome.dp.ua>
uid          Alexander Motin <mav@alkar.net>
sub 2048g/4D59D1C2 2007-04-20 [expires: 2012-04-18]
```

D.3.247. Felipe de Meirelles Motta <lippe@FreeBSD.org>

```
pub 1024D/F2CF7DAE 2008-09-02 [expires: 2010-09-02]
    Key fingerprint = 0532 A900 286D DAFD 099D 394D 231B AF20 F2CF 7DAE
uid          Felipe de Meirelles Motta (FreeBSD Ports Committer) <lippe@FreeBSD.org>
sub 2048g/38E8EEF3 2008-09-02 [expires: 2010-09-02]
```

D.3.248. Rich Murphey <rich@FreeBSD.org>

```
pub 1024R/583443A9 1995-03-31 Rich Murphey <rich@lamprey.utmb.edu>
    Key fingerprint = AF A0 60 C4 84 D6 0C 73 D1 EF C0 E9 9D 21 DB E4
```

D.3.249. Akinori MUSHASHA <knu@FreeBSD.org>

```
pub 1024D/9FD9E1EE 2000-03-21 Akinori MUSHASHA <knu@and.or.jp>
    Key fingerprint = 081D 099C 1705 861D 4B70 B04A 920B EFC7 9FD9 E1EE
uid          Akinori MUSHASHA <knu@FreeBSD.org>
uid          Akinori MUSHASHA <knu@idaemons.org>
uid          Akinori MUSHASHA <knu@ruby-lang.org>
sub 1024g/71BA9D45 2000-03-21
```

D.3.250. Thomas Möstl <tmml@FreeBSD.org>

```
pub 1024D/419C776C 2000-11-28 Thomas Moestl <tmml@FreeBSD.org>
    Key fingerprint = 1C97 A604 2BD0 E492 51D0 9C0F 1FE6 4F1D 419C 776C
uid                                Thomas Moestl <tmoestl@gmx.net>
uid                                Thomas Moestl <t.moestl@tu-bs.de>
sub 2048g/ECE63CE6 2000-11-28
```

D.3.251. Masafumi NAKANE <max@FreeBSD.org>

```
pub 1024D/CE356B59 2000-02-19 Masafumi NAKANE <max@wide.ad.jp>
    Key fingerprint = EB40 BCAB 4CE5 0764 9942 378C 9596 159E CE35 6B59
uid                                Masafumi NAKANE <max@FreeBSD.org>
uid                                Masafumi NAKANE <max@accessibility.org>
uid                                Masafumi NAKANE <kd5pdi@qsl.net>
sub 1024g/FA9BD48B 2000-02-19
```

D.3.252. Maho Nakata <maho@FreeBSD.org>

```
pub 1024D/F28B4069 2009-02-09
    Key fingerprint = 3FE4 99A9 6F41 8161 4F5F 240C 8615 A60C F28B 4069
uid                                Maho NAKATA (NAKATA's FreeBSD.org alias) <maho@FreeBSD.org>
sub 2048g/6B49098E 2009-02-09
```

D.3.253. Yoichi NAKAYAMA <yoichi@FreeBSD.org>

```
pub 1024D/E0788E46 2000-12-28 Yoichi NAKAYAMA <yoichi@assist.media.nagoya-u.ac.jp>
    Key fingerprint = 1550 2662 46B3 096C 0460 BC03 800D 0C8A E078 8E46
uid                                Yoichi NAKAYAMA <yoichi@eken.phys.nagoya-u.ac.jp>
uid                                Yoichi NAKAYAMA <yoichi@FreeBSD.org>
sub 1024g/B987A394 2000-12-28
```

D.3.254. Edward Tomasz Napierala <trasz@FreeBSD.org>

```
pub 1024D/8E53F00E 2007-04-13
    Key fingerprint = DD8F 91B0 12D9 6237 42D9 DBE1 AFC8 CDE9 8E53 F00E
uid                                Edward Tomasz Napierala <trasz@FreeBSD.org>
sub 2048g/7C1F5D67 2007-04-13
```

D.3.255. David Naylor <dbn@FreeBSD.org>

```
pub 1024D/FF6916B2 2008-04-09
    Key fingerprint = 6540 B47C 54AA 3EBA B23B 58AC 51A6 8580 FF69 16B2
uid                                David Naylor <dbn@freebsd.org>
```

```
uid          David Naylor <naylor.b.david@gmail.com>
sub 4096g/77FA885C 2008-04-09
```

D.3.256. Alexander Nedotsukov <bland@FreeBSD.org>

```
pub 1024D/D004116C 2003-08-14 Alexander Nedotsukov <bland@FreeBSD.org>
   Key fingerprint = 35E2 5020 55FC 2071 4ADD 1A4A 86B6 8A5D D004 116C
sub 1024g/1CCA8D46 2003-08-14
```

D.3.257. George V. Neville-Neil <gnn@FreeBSD.org>

```
pub 1024D/440A33D2 2002-09-17
   Key fingerprint = AF66 410F CC8D 1FC9 17DB 6225 61D8 76C1 440A 33D2
uid          George V. Neville-Neil <gnn@freebsd.org>
uid          George V. Neville-Neil <gnn@neville-neil.com>
sub 2048g/95A74F6E 2002-09-17
```

D.3.258. Simon L. B. Nielsen <simon@FreeBSD.org>

```
pub 1024D/FF7490AB 2007-01-14
   Key fingerprint = 4E92 BA8D E45E 85E2 0380 B264 049C 7480 FF74 90AB
uid          Simon L. Nielsen <simon@FreeBSD.org>
uid          Simon L. Nielsen <simon@nitro.dk>
sub 2048g/E3F5A76E 2007-01-14
```

D.3.259. Robert Noland <rnoland@FreeBSD.org>

```
pub 1024D/8A9F44E3 2007-07-24
   Key fingerprint = 107A 0C87 E9D0 E581 677B 2A28 3384 EB43 8A9F 44E3
uid          Robert C. Noland III <rnoland@FreeBSD.org>
uid          Robert C. Noland III (Personal Key) <rnoland@2hip.net>
sub 2048g/76C3CF00 2007-07-24
```

D.3.260. Anders Nordby <anders@FreeBSD.org>

```
pub 1024D/00835956 2000-08-13 Anders Nordby <anders@fix.no>
   Key fingerprint = 1E0F C53C D8DF 6A8F EAAD 19C5 D12A BC9F 0083 5956
uid          Anders Nordby <anders@FreeBSD.org>
sub 2048g/4B160901 2000-08-13
```

D.3.261. Michael Nottebrock <lofi@FreeBSD.org>

```

pub 1024D/6B2974B0 2002-06-06 Michael Nottebrock <michaelnottebrock@gmx.net>
    Key fingerprint = 1079 3C72 0726 F300 B8EC 60F9 5E17 3AF1 6B29 74B0
uid      Michael Nottebrock <lofi@freebsd.org>
uid      Michael Nottebrock <lofi@tigress.com>
uid      Michael Nottebrock <lofi@lofi.dyndns.org>
uid      Michael Nottebrock <michaelnottebrock@web.de>
uid      Michael Nottebrock <michaelnottebrock@meitner.wh.uni-dortmund.de>
sub 1024g/EF652E04 2002-06-06 [expires: 2004-06-15]

```

D.3.262. David O'Brien <obrien@FreeBSD.org>

```

pub 1024R/34F9F9D5 1995-04-23 David E. O'Brien <defunct - obrien@Sea.Legent.com>
    Key fingerprint = B7 4D 3E E9 11 39 5F A3 90 76 5D 69 58 D9 98 7A
uid      David E. O'Brien <obrien@Nuxi.com>
uid      deobrien@ucdavis.edu
uid      David E. O'Brien <whois Do38>
uid      David E. O'Brien <obrien@FreeBSD.org>
uid      David E. O'Brien <dobrien@seas.gwu.edu>
uid      David E. O'Brien <obrien@cs.ucdavis.edu>
uid      David E. O'Brien <defunct - obrien@media.sra.com>
uid      David E. O'Brien <obrien@elsewhere.roanoke.va.us>
uid      David E. O'Brien <obrien@Nuxi.com>

pub 1024D/7F9A9BA2 1998-06-10 "David E. O'Brien" <obrien@cs.ucdavis.edu>
    Key fingerprint = 02FD 495F D03C 9AF2 5DB7 F496 6FC8 DABD 7F9A 9BA2
uid      "David E. O'Brien" <obrien@Nuxi.com>
uid      "David E. O'Brien" <obrien@FreeBSD.org>
sub 3072g/BA32C20D 1998-06-10

```

D.3.263. Jimmy Olgeni <olgeni@FreeBSD.org>

```

pub 2048R/6450AE47 2012-11-01
    Key fingerprint = 7133 AB4D DFC8 0A0D F891 B0D2 90B7 A98E 6450 AE47
uid      Giacomo Olgeni <olgeni@olgeni.com>
uid      Jimmy Olgeni <olgeni@FreeBSD.org>
uid      Giacomo Olgeni <olgeni@moviereading.com>
uid      Giacomo Olgeni <olgeni@unimaccess.com>
uid      Giacomo Olgeni <olgeni@colby.it>
uid      Giacomo Olgeni <olgeni@colby.eu>
uid      Giacomo Olgeni <olgeni@colby.tv>
sub 2048R/1988BB4B 2012-11-01

```

D.3.264. Philip Paeps <philip@FreeBSD.org>

```

pub 4096R/C5D34D05 2006-10-22
    Key fingerprint = 356B AE02 4763 F739 2FA2 E438 2649 E628 C5D3 4D05
uid Philip Paeps <philip@paeps.cx>
uid Philip Paeps <philip@nixsys.be>
uid Philip Paeps <philip@fosdem.org>
uid Philip Paeps <philip@freebsd.org>
uid Philip Paeps <philip@pub.telenet.be>
sub 1024D/035EFC58 2006-10-22
sub 2048g/6E5FD7D6 2006-10-22

```

D.3.265. Josh Paetzel <jpaetzel@FreeBSD.org>

```

pub 2048D/F6F63F01 2012-09-21
    Key fingerprint = 1D8D 506E B58C BD10 DC8C 97E1 D6AD 8621 F6F6 3F01
uid Josh Paetzel <josh@tcbug.org>
uid Josh Paetzel <josh@ixsystems.com>
uid Josh Paetzel <jpaetzel@FreeBSD.org>
sub 2048R/F32EF801 2012-09-21
sub 2048R/51F1335D 2012-09-21
sub 2048g/9BC280CD 2012-09-21
sub 2048g/CC793500 2012-09-21

```

D.3.266. Gábor Páli <pgj@FreeBSD.org>

```

pub 4096R/6D7E445C 2013-06-14 [expires: 2018-06-13]
    Key fingerprint = 7AD5 76BA AF2D 14B9 6D45 440B C013 309D 6D7E 445C
uid Páli Gábor János (Primary identity) <pali.gabor@gmail.com>
uid Páli Gábor János (Eötvös Loránd University) <pgj@inf.elte.hu>
uid Gabor Pali (FreeBSD committer) <pgj@FreeBSD.org>
uid Páli Gábor János (Magyar BSD Egyesület) <pgj@bsd.hu>
uid Páli Gábor János (Eötvös Loránd University) <pgj@elte.hu>
sub 4096R/A57B06AB 2013-06-14 [expires: 2018-06-13]

```

D.3.267. Hiren Panchasara <hiren@FreeBSD.org>

```

pub 4096R/61913185 2013-04-13 [expires: 2014-04-13]
    Key fingerprint = 3336 8104 8D15 B238 2465 136B 4A61 462F 6191 3185
uid hiren panchasara <hiren@freebsd.org>

```

D.3.268. Hiten Pandya <hmp@FreeBSD.org>

```

pub 1024D/938CACA8 2004-02-13 Hiten Pandya (FreeBSD) <hmp@FreeBSD.org>
    Key fingerprint = 84EB C75E C75A 50ED 304E E446 D974 7842 938C ACA8
uid Hiten Pandya <hmp@backplane.com>

```

sub 2048g/783874B5 2004-02-13

D.3.269. Dima Panov <fluffy@FreeBSD.org>

```
pub 1024D/93E3B018 2006-11-08
   Key fingerprint = C73E 2B72 1FFD 61BD E206 1234 A626 76ED 93E3 B018
uid      Dima Panov (FreeBSD.ORG Committer) <fluffy@FreeBSD.ORG>
uid      Dima Panov (at home) <Fluffy@Fluffy.Khv.RU>
uid      Dima Panov (at home) <fluffy.khv@gmail.com>
sub 2048g/89047419 2006-11-08

pub 4096R/D5398F29 2009-08-09
   Key fingerprint = 2D30 2CCB 9984 130C 6F87 BAFB FB8B A09D D539 8F29
uid      Dima Panov (FreeBSD.ORG Committer) <fluffy@FreeBSD.ORG>
uid      Dima Panov (at Home) <fluffy@Fluffy.Khv.RU>
uid      Dima Panov (at GMail) <fluffy.khv@gmail.com>
sub 4096R/915A7785 2009-08-09
```

D.3.270. Andrew Pantyukhin <sat@FreeBSD.org>

```
pub 1024D/6F38A569 2006-05-06
   Key fingerprint = 4E94 994A C2EF CB86 C144 3B04 3381 67C0 6F38 A569
uid      Andrew Pantyukhin <infofarmer@gubkin.ru>
uid      Andrew Pantyukhin <sat@FreeBSD.org>
uid      Andrew Pantyukhin <infofarmer@gmail.com>
uid      Andrew Pantyukhin <infofarmer@mail.ru>
sub 2048g/5BD4D469 2006-05-06
```

D.3.271. Navdeep Parhar <np@FreeBSD.org>

```
pub 1024D/ACAB8812 2009-06-08
   Key fingerprint = C897 7AFB AFC0 4DA9 7B76 D991 CAB2 2B93 ACAB 8812
uid      Navdeep Parhar <np@FreeBSD.org>
sub 2048g/AB61D2DC 2009-06-08
```

D.3.272. Rui Paulo <rpaulo@FreeBSD.org>

```
pub 4096R/39CB4153 2010-02-03
   Key fingerprint = ABE8 8465 DE8F F04D E9C8 3FF6 AF89 B2E6 39CB 4153
uid      Rui Paulo <rpaulo@FreeBSD.org>
uid      Rui Paulo <rpaulo@gmail.com>
sub 4096R/F87D2F34 2010-02-03
```

D.3.273. Mark Peek <mp@FreeBSD.org>

```
pub 1024D/330D4D01 2002-01-27 Mark Peek <mp@FreeBSD.org>
   Key fingerprint = 510C 96EE B4FB 1B0A 2CF8 A0AF 74B0 0B0E 330D 4D01
sub 1024g/9C6CAC09 2002-01-27
```

D.3.274. Peter Pentchev <roam@FreeBSD.org>

```
pub 1024D/16194553 2002-02-01
   Key fingerprint = FDBA FD79 C26F 3C51 C95E DF9E ED18 B68D 1619 4553
uid      Peter Pentchev <roam@ringlet.net>
uid      Peter Pentchev <roam@cnsys.bg>
uid      Peter Pentchev <roam@sbnd.net>
uid      Peter Pentchev <roam@online.bg>
uid      Peter Pentchev <roam@orbitel.bg>
uid      Peter Pentchev <roam@FreeBSD.org>
uid      Peter Pentchev <roam@techlab.officel.bg>
uid      Peter Pentchev <roam@hoster.bg>
uid      Peter Pentchev <roam@space.bg>
sub 1024g/7074473C 2002-02-01

pub 4096R/2527DF13 2009-10-16
   Key fingerprint = 2EE7 A7A5 17FC 124C F115 C354 651E EFB0 2527 DF13
uid      Peter Pentchev <roam@ringlet.net>
uid      Peter Pentchev <roamer@users.sourceforge.net>
uid      Peter Pentchev <roam@cpan.org>
uid      Peter Pentchev <roam@cnsys.bg>
uid      Peter Pentchev <roam@sbnd.net>
uid      Peter Pentchev <roam@online.bg>
uid      Peter Pentchev <roam@orbitel.bg>
uid      Peter Pentchev <roam@FreeBSD.org>
uid      Peter Pentchev <roam@techlab.officel.bg>
uid      Peter Pentchev <roam@hoster.bg>
uid      Peter Pentchev <roam@space.bg>
uid      Peter Pentchev <roam-guest@alioth.debian.org>
uid      Peter Pentchev <ppentchev@alumni.princeton.edu>
sub 4096R/D0B337AA 2009-10-16
```

D.3.275. Denis Peplin <den@FreeBSD.org>

```
pub 1024D/485DDDF5 2003-09-11 Denis Peplin <den@FreeBSD.org>
   Key fingerprint = 495D 158C 8EC9 C2C1 80F5 EA96 6F72 7C1C 485D DDF5
sub 1024g/E70BA158 2003-09-11
```

D.3.276. Christian S.J. Peron <csjp@FreeBSD.org>

```
pub 1024D/033FA33C 2009-05-16
    Key fingerprint = 74AA 6040 89A7 936E D970 DDC0 CC71 6954 033F A33C
uid      Christian S.J. Peron <csjp@FreeBSD.ORG>
sub 2048g/856B194A 2009-05-16
```

D.3.277. Gerald Pfeifer <gerald@FreeBSD.org>

```
pub 1024D/745C015A 1999-11-09 Gerald Pfeifer <gerald@pfeifer.com>
    Key fingerprint = B215 C163 3BCA 0477 615F 1B35 A5B3 A004 745C 015A
uid      Gerald Pfeifer <Gerald.Pfeifer@vibe.at>
uid      Gerald Pfeifer <pfeifer@dbai.tuwien.ac.at>
uid      Gerald Pfeifer <gerald@pfeifer.at>
uid      Gerald Pfeifer <gerald@FreeBSD.org>
sub 1536g/F0156927 1999-11-09
```

D.3.278. Giuseppe Pilichi <jacula@FreeBSD.org>

```
pub 4096R/8B9F4B8B 2006-03-08
    Key fingerprint = 31AD 73AE 0EC0 16E5 4108 8391 D942 5F20 8B9F 4B8B
uid      Giuseppe Pilichi (Jacula Modyun) <jacula@FreeBSD.org>
uid      Giuseppe Pilichi (Jacula Modyun) <jaculamodyun@gmail.com>
uid      Giuseppe Pilichi (Jacula Modyun) <gpilch@gmail.com>
uid      Giuseppe Pilichi (Jacula Modyun) <jacula@gmail.com>
sub 4096R/FB4D05A3 2006-03-08
```

D.3.279. John Polstra <jdp@FreeBSD.org>

```
pub 1024R/BFBCF449 1997-02-14 John D. Polstra <jdp@polstra.com>
    Key fingerprint = 54 3A 90 59 6B A4 9D 61 BF 1D 03 09 35 8D F6 0D
```

D.3.280. Kirill Ponomarew <krion@FreeBSD.org>

```
pub 1024D/AEB426E5 2002-04-07
    Key fingerprint = 58E7 B953 57A2 D9DD 4960 2A2D 402D 46E9 AEB4 26E5
uid      Kirill Ponomarew <krion@voodoo.bawue.com>
uid      Kirill Ponomarew <krion@guug.de>
uid      Kirill Ponomarew <krion@FreeBSD.org>
sub 1024D/05AC7CA0 2006-01-30 [expires: 2008-01-30]
sub 2048g/C3EE5537 2006-01-30 [expires: 2008-01-30]
```

D.3.281. Stephane E. Potvin <sepotvin@FreeBSD.org>

```
pub 1024D/3097FE7B 2002-08-06
    Key fingerprint = 6B56 62FA ADE1 6F46 BB62 8B1C 99D3 97B5 3097 FE7B
uid          Stephane E. Potvin <sepotvin@videotron.ca>
uid          Stephane E. Potvin <stephane.potvin@telcobridges.com>
uid          Stephane E. Potvin <stephane_potvin@telcobridges.com>
uid          Stephane E. Potvin <sepotvin@FreeBSD.org>
sub 2048g/0C427BC9 2002-08-06
```

D.3.282. Mark Pulford <markp@FreeBSD.org>

```
pub 1024D/182C368F 2000-05-10 Mark Pulford <markp@FreeBSD.org>
    Key fingerprint = 58C9 C9BF C758 D8D4 7022 8EF5 559F 7F7B 182C 368F
uid          Mark Pulford <mark@kyne.com.au>
sub 2048g/380573E8 2000-05-10
```

D.3.283. Alejandro Pulver <alepulver@FreeBSD.org>

```
pub 1024D/945C3F61 2005-11-13
    Key fingerprint = 085F E8A2 4896 4B19 42A4 4179 895D 3912 945C 3F61
uid          Alejandro Pulver (Ale's GPG key pair) <alepulver@FreeBSD.org>
uid          Alejandro Pulver (Ale's GPG key pair) <alejandro@varnet.biz>
sub 2048g/6890C6CA 2005-11-13
```

D.3.284. Thomas Quinot <thomas@FreeBSD.org>

```
pub 1024D/393D2469 1999-09-23 Thomas Quinot <thomas@cuivre.fr.eu.org>
    Empreinte de la clé = 4737 A0AD E596 6D30 4356 29B8 004D 54B8 393D 2469
uid          Thomas Quinot <thomas@debian.org>
uid          Thomas Quinot <thomas@FreeBSD.org>
sub 1024g/8DE13BB2 1999-09-23
```

D.3.285. Herve Quiroz <hq@FreeBSD.org>

```
pub 1024D/85AC8A80 2004-07-22 Herve Quiroz <hq@FreeBSD.org>
    Key fingerprint = 14F5 BC56 D736 102D 41AF A07B 1D97 CE6C 85AC 8A80
uid          Herve Quiroz <herve.quiroz@esil.univ-mrs.fr>
sub 1024g/8ECCAFED 2004-07-22
```

D.3.286. Doug Rabson <dfr@FreeBSD.org>

```
pub 1024D/59F57821 2004-02-07
    Key fingerprint = 9451 C4FE 1A7E 117B B95F 1F8F B123 456E 59F5 7821
uid          Doug Rabson <dfr@nlsystems.com>
sub 1024g/6207AA32 2004-02-07
```

D.3.287. Lars Balkar Rasmussen <lbr@FreeBSD.org>

```
pub 1024D/9EF6F27F 2006-04-30
    Key fingerprint = F251 28B7 897C 293E 04F8 71EE 4697 F477 9EF6 F27F
uid          Lars Balkar Rasmussen <lbr@FreeBSD.org>
sub 2048g/A8C1CFD4 2006-04-30
```

D.3.288. Chris Rees <crees@FreeBSD.org>

```
pub 2048R/1E12E96A 2012-08-26
    Key fingerprint = 8C57 BE3B D320 5FFC C4C3 C0B0 900F 45A6 1E12 E96A
uid          Chris Rees <crees@FreeBSD.org>
sub 2048R/C10740CD 2012-08-26 [expires: 2013-08-26]
```

D.3.289. Jim Rees <rees@FreeBSD.org>

```
pub 512/B623C791 1995/02/21 Jim Rees <rees@umich.edu>
    Key fingerprint = 02 5F 1B 15 B4 6E F1 3E F1 C5 E0 1D EA CC 17 88
```

D.3.290. Benedict Reuschling <bcr@FreeBSD.org>

```
pub 1024D/4A819348 2009-05-24
    Key fingerprint = 2D8C BDF9 30FA 75A5 A0DF D724 4D26 502E 4A81 9348
uid          Benedict Reuschling <bcr@FreeBSD.org>
sub 2048g/8DA16EDD 2009-05-24
```

D.3.291. Tom Rhodes <trhodes@FreeBSD.org>

```
pub 1024D/FB7D88E1 2008-05-07
    Key fingerprint = 8279 3100 2DF2 F00E 7FDD AC2C 5776 23AB FB7D 88E1
uid          Tom Rhodes (trhodes) <trhodes@FreeBSD.org>
sub 4096g/7B0CD79F 2008-05-07
```

D.3.292. Benno Rice <benno@FreeBSD.org>

```
pub 4096R/C5F10BED 2013-05-21 [expires: 2017-05-21]
    Key fingerprint = 77EB 5A9E 97C7 2D2D 6D0A 1B6C C619 4C61 C5F1 0BED
uid Benno Rice <benno@FreeBSD.org>
uid Benno Rice <benno@jeamland.net>
sub 4096R/408068BC 2013-05-21 [expires: 2017-05-21]
```

D.3.293. Beech Rintoul <beech@FreeBSD.org>

```
pub 2048D/68DFAE1F 2013-02-26
    Key fingerprint = D58B 3E9D B0E3 E081 EC6F 69D9 CDA3 51DD 68DF AE1F
uid Beech Rintoul <beech@freebsd.org>
sub 2048g/960F45D9 2013-02-26
```

D.3.294. Matteo Rionato <matteo@FreeBSD.org>

```
pub 1024D/1EC56BEC 2003-01-05 [expires: 2009-09-07]
    Key fingerprint = F0F3 1B43 035D 65B1 08E9 4D66 D8CA 78A5 1EC5 6BEC
uid Matteo Rionato (Rionda) <matteo@FreeBSD.ORG>
uid Matteo Rionato (Rionda) <rionda@riondabsd.net>
uid Matteo Rionato (Rionda) <rionda@gufi.org>
uid Matteo Rionato (Rionda) <matteo@rionato.com>
uid Matteo Rionato (Rionda) <rionda@rionato.com>
uid Matteo Rionato (Rionda) <rionda@FreeSBIE.ORG>
uid Matteo Rionato (Rionda) <rionda@autistici.org>
sub 2048g/87C44A55 2008-09-23 [expires: 2009-09-23]
```

D.3.295. Ollivier Robert <roberto@FreeBSD.org>

```
pub 1024D/7DCAE9D3 1997-08-21
    Key fingerprint = 2945 61E7 D4E5 1D32 C100 DBEC A04F FB1B 7DCA E9D3
uid Ollivier Robert <roberto@keltia.freenix.fr>
uid Ollivier Robert <roberto@FreeBSD.org>
sub 2048g/C267084D 1997-08-21
```

D.3.296. Craig Rodrigues <rodrigc@FreeBSD.org>

```
pub 1024D/3998479D 2005-05-20
    Key fingerprint = F01F EBE6 F5C8 6DC2 954F 098F D20A 8A2A 3998 479D
uid Craig Rodrigues <rodrigc@freebsd.org>
uid Craig Rodrigues <rodrigc@crodrigues.org>
sub 2048g/AA77E09B 2005-05-20
```

D.3.297. Guido van Rooij <guido@FreeBSD.org>

```
pub 1024R/599F323D 1996-05-18 Guido van Rooij <guido@gvr.org>
   Key fingerprint = 16 79 09 F3 C0 E4 28 A7 32 62 FA F6 60 31 C0 ED
uid                               Guido van Rooij <guido@gvr.win.tue.nl>

pub 1024D/A95102C1 2000-10-25 Guido van Rooij <guido@madison-gurkha.nl>
   Key fingerprint = 5B3E 51B7 0E7A D170 0574 1E51 2471 117F A951 02C1
uid                               Guido van Rooij <guido@madison-gurkha.com>
sub 1024g/A5F20553 2000-10-25
```

D.3.298. Eygene Ryabinkin <rea@FreeBSD.org>

```
pub 3072D/8152ECFB 2010-10-27
   Key fingerprint = 82FE 06BC D497 C0DE 49EC 4FF0 16AF 9EAE 8152 ECFB
uid                               Eygene Ryabinkin <rea-fbsd@codelabs.ru>
uid                               Eygene Ryabinkin <rea@freebsd.org>
uid                               Eygene Ryabinkin <rea@codelabs.ru>
sub 3072g/5FC03749 2010-10-27
```

D.3.299. Aleksandr Rybalko <ray@FreeBSD.org>

```
pub 2048R/4B7B7A4E 2011-05-24
   Key fingerprint = BB9F D01D 7327 0B33 B2F5 6C72 EC49 E6ED 4B7B 7A4E
uid                               Aleksandr Rybalko (Aleksandr Rybalko FreeBSD project identification) <ray@fr
sub 2048R/99F9F9EF 2011-05-24
```

D.3.300. Niklas Saers <niklas@FreeBSD.org>

```
pub 1024D/C822A476 2004-03-09 Niklas Saers <niklas@saers.com>
   Key fingerprint = C41E F734 AF0E 3D21 7499 9EB1 9A31 2E7E C822 A476
sub 1024g/81E2FF36 2004-03-09
```

D.3.301. Boris Samorodov <bsam@FreeBSD.org>

```
pub 1024D/ADFD5C9A 2006-06-21
   Key fingerprint = 81AA FED0 6050 208C 0303 4007 6C03 7263 ADFD 5C9A
uid                               Boris Samorodov (FreeBSD) <bsam@freebsd.org>
sub 2048g/7753A3F1 2006-06-21
```

D.3.302. Mark Santcroos <marks@FreeBSD.org>

```
pub 1024D/DBE7EB8E 2005-03-08
    Key fingerprint = C0F0 44F3 3F15 520F 6E32 186B BE0A BA42 DBE7 EB8E
uid                               Mark Santcroos <marks@ripe.net>
uid                               Mark Santcroos <mark@santcroos.net>
uid                               Mark Santcroos <marks@freebsd.org>
sub 2048g/FFF80F85 2005-03-08
```

D.3.303. Bernhard Schmidt <bschmidt@FreeBSD.org>

```
pub 1024D/5F754FBC 2009-06-15
    Key fingerprint = 6B87 C8A9 6BA5 6B18 11CF 8C38 A1B7 0731 5F75 4FBC
uid                               Bernhard Schmidt <bschmidt@FreeBSD.org>
uid                               Bernhard Schmidt <bschmidt@techwires.net>
sub 1024g/1945DC1D 2009-06-15
```

D.3.304. Wolfram Schneider <wosch@FreeBSD.org>

```
Type Bits/KeyID      Date          User ID
pub 1024/2B7181AD 1997/08/09 Wolfram Schneider <wosch@FreeBSD.org>
    Key fingerprint = CA 16 91 D9 75 33 F1 07 1B F0 B4 9F 3E 95 B6 09
```

D.3.305. Ed Schouten <ed@FreeBSD.org>

```
pub 4096R/3491A2BB 2011-03-12 [expires: 2016-03-10]
    Key fingerprint = A110 5982 A887 74A2 F4B1 D70A 6E5E D8FE 3491 A2BB
uid                               Ed Schouten (The FreeBSD Project) <ed@FreeBSD.org>
uid                               Ed Schouten <ed@80386.nl>
sub 4096R/81BB41E6 2011-03-12 [expires: 2016-03-10]
```

D.3.306. David Schultz <das@FreeBSD.org>

```
pub 1024D/BE848B57 2001-07-19 David Schultz <das@FreeBSD.ORG>
    Key fingerprint = 0C12 797B A9CB 19D9 FDAF 2A39 2D76 A2DB BE84 8B57
uid David Schultz <dschultz@uclink.Berkeley.EDU>
uid David Schultz <das@FreeBSD.ORG>
sub 2048g/69206E8E 2001-07-19
```

D.3.307. Michael Scheidell <scheidell@FreeBSD.org>

```
pub 2048R/34622C1D 2011-11-16
    Key fingerprint = 0A0C 9ECA 18EC 47AC C715 2187 91B9 F9FE 3462 2C1D
uid                               Michael Scheidell <scheidell@freebsd.org>
```

```
sub 2048R/8F241971 2011-11-16
```

D.3.308. Jens Schweikhardt <schweikh@FreeBSD.org>

```
pub 1024D/0FF231FD 2002-01-27 Jens Schweikhardt <schweikh@FreeBSD.org>
    Key fingerprint = 3F35 E705 F02F 35A1 A23E 330E 16FE EA33 0FF2 31FD
uid                                Jens Schweikhardt <schweikh@schweikhardt.net>
sub 1024g/6E93CACC 2002-01-27 [expires: 2005-01-26]
```

D.3.309. Matthew Seaman <matthew@FreeBSD.org>

```
pub 1024D/60AE908C 2005-12-17 [expires: 2012-03-21]
    Key fingerprint = B555 2A96 274E D248 5734 0EB4 F0C8 E4E7 60AE 908C
uid                                Matthew Seaman <m.seaman@infracaninophile.co.uk>
uid                                Matthew Seaman <m.seaman@black-earth.co.uk>
uid                                Matthew Seaman <matthew@freebsd.org>
sub 2048g/58BFDA29 2005-12-17 [expires: 2012-03-21]
sub 1024D/9B19F956 2006-12-18 [expires: 2012-03-21]
```

D.3.310. Thomas-Martin Seck <tmseck@FreeBSD.org>

```
pub 1024D/DF46EE05 2000-11-22
    Key fingerprint = A38F AE66 6B11 6EB9 5D1A B67D 2444 2FE1 DF46 EE05
uid                                Thomas-Martin Seck (Privat 2) <tmseck@netcologne.de>
uid                                Thomas-Martin Seck (Privat) <tmseck@web.de>
uid                                Thomas-Martin Seck (FreeBSD) <tmseck@FreeBSD.org>
sub 2048g/3DC33B0F 2000-11-22
```

D.3.311. Stanislav Sedov <stas@FreeBSD.org>

```
pub 4096R/092FD9F0 2009-05-23
    Key fingerprint = B83A B15D 929A 364A D8BC B3F9 BF25 A231 092F D9F0
uid                                Stanislav Sedov <stas@FreeBSD.org>
uid                                Stanislav Sedov <stas@SpringDaemons.com>
uid                                Stanislav Sedov (Corporate email) <stas@deglitch.com>
uid                                Stanislav Sedov (Corporate email) <stas@ht-systems.ru>
uid                                Stanislav Sedov (Corporate email) <ssedov@3playnet.com>
uid                                Stanislav Sedov <ssedov@mbsd.msk.ru>
uid                                Stanislav Sedov (Corporate email) <ssedov@swifttest.com>
sub 4096R/6FD2025F 2009-05-23
```

D.3.312. Johan van Selst <johans@FreeBSD.org>

```

pub 4096R/D3AE8D3A 2009-09-01
   Key fingerprint = 31C8 D089 DDB6 96C6 F3C1 29C0 A9C8 6C8D D3AE 8D3A
uid          Johan van Selst
uid          Johan van Selst <johans@gletsjer.net>
uid          Johan van Selst <johans@stack.nl>
uid          Johan van Selst <johans@FreeBSD.org>
uid          Johan van Selst (GSWoT:NL50) <johans@gswot.org>
sub 2048R/B002E38C 2009-09-01
sub 2048R/1EBCAECB 2009-09-01
sub 2048R/639A1446 2009-09-01
sub 3072D/6F2708F4 2009-09-01
sub 4096g/D6F89E83 2009-09-01

```

D.3.313. Bakul Shah <bakul@FreeBSD.org>

```

pub 1024D/86AEE4CB 2006-04-20
   Key fingerprint = 0389 26E8 381C 6980 AEC0 10A5 E540 A157 86AE E4CB
uid          Bakul Shah <bakul@freebsd.org>
sub 2048g/5C3DCC24 2006-04-20

```

D.3.314. Gregory Neil Shapiro <gshapiro@FreeBSD.org>

```

pub 1024R/4FBE2ADD 2000-10-13 Gregory Neil Shapiro <gshapiro@gshapiro.net>
   Key fingerprint = 56 D5 FF A7 A6 54 A6 B5 59 10 00 B9 5F 5F 20 09
uid          Gregory Neil Shapiro <gshapiro@FreeBSD.org>

pub 1024D/F76A9BF5 2001-11-14 Gregory Neil Shapiro <gshapiro@FreeBSD.org>
   Key fingerprint = 3B5E DAF1 4B04 97BA EE20 F841 21F9 C5BC F76A 9BF5
uid          Gregory Neil Shapiro <gshapiro@gshapiro.net>
sub 2048g/935657DC 2001-11-14

pub 1024D/FCE56561 2000-10-14 Gregory Neil Shapiro <gshapiro@FreeBSD.org>
   Key fingerprint = 42C4 A87A FD85 C34F E77F 5EA1 88E1 7B1D FCE5 6561
uid          Gregory Neil Shapiro <gshapiro@gshapiro.net>
sub 1024g/285DC8A0 2000-10-14 [expires: 2001-10-14]

```

D.3.315. Arun Sharma <arun@FreeBSD.org>

```

pub 1024D/7D112181 2003-03-06 Arun Sharma <arun@sharma-home.net>
   Key fingerprint = A074 41D6 8537 C7D5 070E 0F78 0247 1AE2 7D11 2181
uid          Arun Sharma <arun@freebsd.org>
uid          Arun Sharma <arun.sharma@intel.com>
sub 1024g/ACAD98DA 2003-03-06 [expires: 2005-03-05]

```

D.3.316. Wesley Shields <wxs@FreeBSD.org>

```
pub 1024D/17F0AA37 2007-12-27
    Key fingerprint = 96D1 2E6B F61C 2F3D 83EF 8F0B BE54 310C 17F0 AA37
uid Wesley Shields <wxs@FreeBSD.org>
uid Wesley Shields <wxs@atarininja.org>
sub 2048g/2EDA1BB8 2007-12-27
```

D.3.317. Norikatsu Shigemura <nork@FreeBSD.org>

```
pub 1024D/7104EA4E 2005-02-14
    Key fingerprint = 9580 60A3 B58A 0864 79CB 779A 6FAE 229B 7104 EA4E
uid Norikatsu Shigemura <nork@cityfujisawa.ne.jp>
uid Norikatsu Shigemura <nork@ninth-nine.com>
uid Norikatsu Shigemura <nork@FreeBSD.org>
sub 4096g/EF56997E 2005-02-14
```

D.3.318. Shteryana Shopova <syrinx@FreeBSD.org>

```
pub 1024D/1C139BC5 2006-10-07
    Key fingerprint = B83D 2451 27AB B767 504F CB85 4FB1 C88B 1C13 9BC5
uid Shteryana Shopova (syrinx) <shteryana@FreeBSD.org>
sub 2048g/6D2E9C98 2006-10-07
```

D.3.319. Vanilla I. Shu <vanilla@FreeBSD.org>

```
pub 1024D/ACE75853 2001-11-20 Vanilla I. Shu <vanilla@FreeBSD.org>
    Key fingerprint = 290F 9DB8 42A3 6257 5D9A 5585 B25A 909E ACE7 5853
sub 1024g/CE695D0E 2001-11-20
```

D.3.320. Ashish SHUKLA <ashish@FreeBSD.org>

```
pub 4096R/E74FA4B0 2010-04-13
    Key fingerprint = F682 CDCC 39DC 0FEA E116 20B6 C746 CFA9 E74F A4B0
uid Ashish SHUKLA <wahjava@gmail.com>
uid Ashish SHUKLA <wahjava@googlemail.com>
uid Ashish SHUKLA <wahjava.ml@gmail.com>
uid Ashish SHUKLA <wahjava@members.fsf.org>
uid Ashish SHUKLA <wahjava@perl.org.in>
uid Ashish SHUKLA <wahjava@users.sourceforge.net>
uid Ashish SHUKLA <wah.java@yahoo.com>
uid Ashish SHUKLA <wah_java@hotmail.com>
uid Ashish SHUKLA <ashish.shukla@airtelmail.in>
uid Ashish SHUKLA <wahjava@member.fsf.org>
uid [jpeg image of size 4655]
uid Ashish SHUKLA (FreeBSD Committer Address) <ashish@FreeBSD.ORG>
```

sub 4096R/F20D202D 2010-04-13

D.3.321. Bruce M. Simpson <bms@FreeBSD.org>

pub 1024D/860DB53B 2003-08-06 Bruce M Simpson <bms@freebsd.org>
 Key fingerprint = 0D5F 1571 44DF 51B7 8B12 041E B9E5 2901 860D B53B
 sub 2048g/A2A32D8B 2003-08-06 [expires: 2006-08-05]

D.3.322. Dmitry Sivachenko <demon@FreeBSD.org>

pub 1024D/13D5DF80 2002-03-18 Dmitry Sivachenko <mitya@cavia.pp.ru>
 Key fingerprint = 72A9 12C9 BB02 46D4 4B13 E5FE 1194 9963 13D5 DF80
 uid Dmitry S. Sivachenko <demon@FreeBSD.org>
 sub 1024g/060F6DBD 2002-03-18

D.3.323. Jesper Skriver <jesper@FreeBSD.org>

pub 1024D/F9561C31 2001-03-09 Jesper Skriver <jesper@FreeBSD.org>
 Key fingerprint = 6B88 9CE8 66E9 E631 C9C5 5EB4 22AB F0EC F956 1C31
 uid Jesper Skriver <jesper@skriver.dk>
 uid Jesper Skriver <jesper@wheel.dk>
 sub 1024g/777C378C 2001-03-09

D.3.324. Ville Skyttä <scop@FreeBSD.org>

pub 1024D/BCD241CB 2002-04-07 Ville Skyttä <ville.skytta@iki.fi>
 Key fingerprint = 4E0D EBAB 3106 F1FA 3FA9 B875 D98C D635 BCD2 41CB
 uid Ville Skyttä <ville.skytta@xemacs.org>
 uid Ville Skyttä <scop@FreeBSD.org>
 sub 2048g/9426F4D1 2002-04-07

D.3.325. Andrey Slusar <anray@FreeBSD.org>

pub 1024D/AE7B5418 2005-12-12
 Key fingerprint = DE70 C24B 55A0 4A06 68A1 D425 3C59 9A9B AE7B 5418
 uid Andrey Slusar <anray@ext.by>
 uid Andrey Slusar <anrays@gmail.com>
 uid Andrey Slusar <anray@FreeBSD.org>
 sub 2048g/7D0EB77D 2005-12-12

D.3.326. Florian Smeets <flo@FreeBSD.org>

```
pub 1024D/C942BF09 2008-10-24
    Key fingerprint = 54BB 157B 8DB2 9E46 4A3C 69AB 6A9A 3C3F C942 BF09
uid          Florian Smeets <flo@smeets.im>
uid          Florian Smeets <flo@kasimir.com>
uid          Florian Smeets <flo@FreeBSD.org>
sub 2048g/4AAF040E 2008-10-24
```

D.3.327. Gleb Smirnov <glebius@FreeBSD.org>

```
pub 2048D/6C7E5E82 2013-01-30 [expires: 2023-08-25]
    Key fingerprint = 6E06 7260 B83D CF2C A93C 566F 5185 0968 6C7E 5E82
uid          Gleb Smirnov <glebius@FreeBSD.org>
sub 2048g/11E89DCE 2013-01-30 [expires: 2023-08-25]
```

D.3.328. Ken Smith <kensmith@FreeBSD.org>

```
pub 1024D/29AEA7F6 2003-12-02 Ken Smith <kensmith@cse.buffalo.edu>
    Key fingerprint = 4AB7 D302 0753 8215 31E7 F1AD FC6D 7855 29AE A7F6
uid          Ken Smith <kensmith@freebsd.org>
sub 1024g/0D509C6C 2003-12-02
```

D.3.329. Ben Smithurst <ben@FreeBSD.org>

```
pub 1024D/2CEF442C 2001-07-11 Ben Smithurst <ben@LSRfm.com>
    Key fingerprint = 355D 0FFF B83A 90A9 D648 E409 6CFC C9FB 2CEF 442C
uid          Ben Smithurst <ben@vinosystems.com>
uid          Ben Smithurst <ben@smithurst.org>
uid          Ben Smithurst <ben@FreeBSD.org>
uid          Ben Smithurst <csxbs@comp.leeds.ac.uk>
uid          Ben Smithurst <ben@scientia.demon.co.uk>
sub 1024g/347071FF 2001-07-11
```

D.3.330. Dag-Erling C. Smørgrav <des@FreeBSD.org>

```
pub 4096R/F94E87B2 2013-02-15 [expires: 2015-01-01]
    Key fingerprint = 578A 3F4F 9E04 9FCF 3576 BF82 BB9B 471B F94E 87B2
uid          Dag-Erling Smørgrav <des@usit.uio.no>
uid          Dag-Erling Smørgrav <des@des.no>
uid          Dag-Erling Smørgrav <des@freebsd.org>
uid          [jpeg image of size 4779]
sub 4096R/F4DE87F5 2013-02-15 [expires: 2015-01-01]
```

D.3.331. Maxim Sobolev <sobomax@FreeBSD.org>

```

pub 1024D/888205AF 2001-11-21 Maxim Sobolev <sobomax@FreeBSD.org>
    Key fingerprint = 85C9 DCB0 6828 087C C977 3034 A0DB B9B7 8882 05AF
uid                                     Maxim Sobolev <sobomax@mail.ru>
uid                                     Maxim Sobolev <sobomax@altavista.net>
uid                                     Maxim Sobolev <vegacap@i.com.ua>

pub 1024D/468EE6D8 2003-03-21 Maxim Sobolev <sobomax@portaone.com>
    Key fingerprint = 711B D315 3360 A58F 9A0E 89DB 6D40 2558 468E E6D8
uid                                     Maxim Sobolev <sobomax@FreeBSD.org>
uid                                     Maxim Sobolev <sobomax@mail.ru>
uid                                     Maxim Sobolev <vegacap@i.com.ua>

pub 1024D/6BEC980A 2004-02-13 Maxim Sobolev <sobomax@portaone.com>
    Key fingerprint = 09D5 47B4 8D23 626F B643 76EB DFEE 3794 6BEC 980A
uid                                     Maxim Sobolev <sobomax@FreeBSD.org>
uid                                     Maksym Sobolyev (It's how they call me in official documents. Pret
uid                                     Maksym Sobolyev (It's how they call me in official documents. Pret
sub 2048g/16D049AB 2004-02-13 [expires: 2005-02-12]
```

D.3.332. Alan Somers <asomers@FreeBSD.org>

```

pub 4096R/DA05FCE8 2013-04-25 [expires: 2018-04-24]
    Key fingerprint = 9CD4 C982 738F 8B90 25E8 E6B3 5F74 63BC DA05 FCE8
uid                                     Alan Somers <asomers@freebsd.org>
uid                                     Alan Somers <asomers@gmail.com>
sub 4096R/4E121B3E 2013-04-25 [expires: 2018-04-24]
```

D.3.333. Brian Somers <brian@FreeBSD.org>

```

pub 1024R/666A7421 1997-04-30 Brian Somers <brian@freebsd-services.com>
    Key fingerprint = 2D 91 BD C2 94 2C 46 8F 8F 09 C4 FC AD 12 3B 21
uid                                     Brian Somers <brian@awfulhak.org>
uid                                     Brian Somers <brian@FreeBSD.org>
uid                                     Brian Somers <brian@OpenBSD.org>
uid                                     Brian Somers <brian@uk.FreeBSD.org>
uid                                     Brian Somers <brian@uk.OpenBSD.org>
```

D.3.334. Stacey Son <sson@FreeBSD.org>

```

pub 1024D/CE8319F3 2008-07-08
    Key fingerprint = 64C7 8D92 C1DF B940 1171 5ED3 186A 758A CE83 19F3
uid                                     Stacey Son <sson@FreeBSD.org>
uid                                     Stacey Son <stacey@son.org>
uid                                     Stacey Son <sson@byu.net>
uid                                     Stacey Son <sson@secure.net>
uid                                     Stacey Son <sson@dev-random.com>
```

sub 2048g/0F724E52 2008-07-08

D.3.335. Nicolas Souchu <nsouch@FreeBSD.org>

pub 1024D/C744F18B 2002-02-13 Nicholas Souchu <nsouch@freebsd.org>
 Key fingerprint = 992A 144F AC0F 40BA 55AE DE6D 752D 0A6C C744 F18B
 sub 1024g/90BD3231 2002-02-13

D.3.336. Suleiman Souhlal <ssouhlal@FreeBSD.org>

pub 1024D/2EA50469 2004-07-24 Suleiman Souhlal <ssouhlal@FreeBSD.org>
 Key fingerprint = DACF 89DB 54C7 DA1D 37AF 9A94 EB55 E272 2EA5 0469
 sub 2048g/0CDCC535 2004-07-24

D.3.337. Luiz Otavio O Souza <loos@FreeBSD.org>

pub 2048R/39165690 2013-07-03
 Key fingerprint = ABC9 71D9 016E 8D4A 936D D748 6252 872F 3916 5690
 uid Luiz Otavio O Souza <loos@freebsd.org>
 sub 2048R/9D089395 2013-07-03

D.3.338. Ulrich Spörlein <uqs@FreeBSD.org>

pub 2048R/4AAF82CE 2010-01-27 [expires: 2015-01-26]
 Key fingerprint = 08DF A6A0 B1EB 98A5 EDDA 9005 A3A6 9864 4AAF 82CE
 uid Ulrich Spörlein <uqs@spoerlein.net>
 uid Ulrich Spoerlein <uspoerlein@gmail.com>
 uid Ulrich Spörlein (The FreeBSD Project) <uqs@FreeBSD.org>
 uid Ulrich Spörlein <ulrich.spoerlein@web.de>
 sub 2048R/162E8BD2 2010-01-27 [expires: 2015-01-26]

D.3.339. Rink Springer <rink@FreeBSD.org>

pub 1024D/ECEDBFFF 2003-09-19
 Key fingerprint = A8BE 9C82 9B81 4289 A905 418D 6F73 BAD2 ECED BFFF
 uid Rink Springer <rink@il.fontys.nl>
 uid Rink Springer (FreeBSD Project) <rink@FreeBSD.org>
 uid Rink Springer <rink@stack.nl>
 sub 2048g/3BC3E67E 2003-09-19

D.3.340. Vsevolod Stakhov <vsevolod@FreeBSD.org>

```
pub 4096R/90081437 2012-05-16 [expires: 2017-05-15]
    Key fingerprint = DD9A 126C E675 1EA5 2A97 04A3 0764 7B67 9008 1437
uid                               Vsevolod Stakhov <vsevolod@FreeBSD.org>
sub 4096R/4A5A0B54 2012-05-16 [expires: 2017-05-15]
```

D.3.341. Ryan Steinmetz <zi@FreeBSD.org>

```
pub 1024D/7AD7FAF2 2004-01-21
    Key fingerprint = EF36 D45A 5CA9 28B1 A550 18CD A43C D111 7AD7 FAF2
uid                               Ryan Steinmetz <zi@FreeBSD.org>
uid                               Ryan Steinmetz <rpsfa@rit.edu>
uid                               Ryan Steinmetz <zi@zi0r.com>
sub 1024g/058BC057 2004-01-21
sub 4096g/0EB108D2 2006-02-27
sub 1024D/FEF36DD7 2006-02-27
```

D.3.342. Randall R. Stewart <rrs@FreeBSD.org>

```
pub 1024D/0373B8B2 2006-09-01
    Key fingerprint = 74A6 810E 6DEA D69B 6496 5FA9 8AEF 4166 0373 B8B2
uid                               Randall R Stewart <randall@lakerest.net>
uid                               Randall R Stewart <rrs@cisco.com>
uid                               Randall R Stewart <rrs@FreeBSD.org>
sub 2048g/88027C0B 2006-09-01
```

D.3.343. Murray Stokely <murray@FreeBSD.org>

```
pub 1024D/0E451F7D 2001-02-12 Murray Stokely <murray@freebsd.org>
    Key fingerprint = E2CA 411D DD44 53FD BB4B 3CB5 B4D7 10A2 0E45 1F7D
sub 1024g/965A770C 2001-02-12
```

D.3.344. Volker Stolz <vs@FreeBSD.org>

```
pub 1024R/3FD1B6B5 1998-06-16 Volker Stolz <vs@freebsd.org>
    Key fingerprint = 69 6F BD A0 2E FE 19 66 CF B9 68 6E 41 7D F9 B9
uid                               Volker Stolz <stolz@i2.informatik.rwth-aachen.de> (LSK)
uid                               Volker Stolz <vs@foldr.org>
```

D.3.345. Ryan Stone <rstone@FreeBSD.org>

```
pub 1024D/3141B73A 2010-04-13
    Key fingerprint = 4A6D DC04 DDC5 0822 2687 A086 FD3F 16CB 3141 B73A
uid          Ryan Stone (FreeBSD) <rstone@freebsd.org>
sub 2048g/A8500B5F 2010-04-13
```

D.3.346. Søren Straarup <xride@FreeBSD.org>

```
pub 1024D/E683AD40 2006-09-28
    Key fingerprint = 8A0E 7E57 144B BC25 24A9 EC1A 0DBC 3408 E683 AD40
uid          Soeren Straarup <xride@xride.dk>
uid          Soeren Straarup <xride@FreeBSD.org>
uid          Soeren Straarup <xride@x12.dk>
sub 2048g/2B18B3B8 2006-09-28
```

D.3.347. Marius Strobl <marius@FreeBSD.org>

```
pub 1024D/E0AC6F8D 2004-04-16
    Key fingerprint = 3A6C 4FB1 8BB9 4F2E BDDC 4AB6 D035 799C E0AC 6F8D
uid          Marius Strobl <marius@FreeBSD.org>
uid          Marius Strobl <marius@alchemy.franken.de>
sub 1024g/08BBD875 2004-04-16
```

D.3.348. Carlo Strub <cs@FreeBSD.org>

```
pub 3072R/D06F0BD7 2012-11-25 [expires: 2017-11-24]
    Key fingerprint = 61A4 F2B8 2A6C B81E 5557 0798 78E7 DE70 D06F 0BD7
uid          Carlo Strub <cs@carlostrub.ch>
uid          Carlo Strub <cs@FreeBSD.org>
sub 3072R/71C75997 2012-11-25 [expires: 2017-11-24]
sub 3072R/318AEB16 2012-11-25 [expires: 2017-11-24]
```

D.3.349. Cheng-Lung Sung <clsung@FreeBSD.org>

```
pub 1024D/956E8BC1 2003-09-12 Cheng-Lung Sung <clsung@FreeBSD.org>
    Key fingerprint = E0BC 57F9 F44B 46C6 DB53 8462 F807 89F3 956E 8BC1
uid          Cheng-Lung Sung (Software Engineer) <clsung@dragon2.net>
uid          Cheng-Lung Sung (Alumnus of CSIE, NCTU, Taiwan) <clsung@sungsung.c
uid          Cheng-Lung Sung (AlanSung) <clsung@tiger2.net>
uid          Cheng-Lung Sung (FreeBSD@Taiwan) <clsung@freebsd.csie.nctu.edu.tw>
uid          Cheng-Lung Sung (Ph.D. Student of NTU.EECS) <d92921016@ntu.edu.tw>
uid          Cheng-Lung Sung (FreeBSD Freshman) <clsung@tw.freebsd.org>
uid          Cheng-Lung Sung (ports committer) <clsung@FreeBSD.org>
sub 1024g/1FB800C2 2003-09-12
```

D.3.350. Gregory Sutter <gsutter@FreeBSD.org>

```
pub 1024D/845DFEDD 2000-10-10 Gregory S. Sutter <gsutter@zer0.org>
    Key fingerprint = D161 E4EA 4BFA 2427 F3F9 5B1F 2015 31D5 845D FEDD
uid Gregory S. Sutter <gsutter@freebsd.org>
uid Gregory S. Sutter <gsutter@daemonnews.org>
uid Gregory S. Sutter <gsutter@pobox.com>
sub 2048g/0A37BBCE 2000-10-10
```

D.3.351. Koichi Suzuki <metal@FreeBSD.org>

```
pub 1024D/AE562682 2004-05-23 SUZUKI Koichi <metal@FreeBSD.org>
    Key fingerprint = 92B9 A202 B5AB 8CB6 89FC 6DD1 5737 C702 AE56 2682
sub 4096g/730E604B 2004-05-23
```

D.3.352. Ryusuke SUZUKI <ryusuke@FreeBSD.org>

```
pub 1024D/63D29724 2009-12-18
    Key fingerprint = B108 7109 2E62 BECB 0F78 FE65 1B9A D1BE 63D2 9724
uid Ryusuke SUZUKI <ryusuke@FreeBSD.org>
uid Ryusuke SUZUKI <ryusuke@jp.FreeBSD.org>
sub 1024g/5E4DD044 2009-12-18
```

D.3.353. Gary W. Swearingen <garys@FreeBSD.org>

```
pub 1024D/FAA48AD5 2005-08-22 [expires: 2007-08-22]
    Key fingerprint = 8292 CC3E 81B5 E54F E3DD F987 FA52 E643 FAA4 8AD5
uid Gary W. Swearingen <garys@freebsd.org>
sub 2048g/E34C3CA0 2005-08-22 [expires: 2007-08-22]
```

D.3.354. Yoshihiro Takahashi <nyan@FreeBSD.org>

```
pub 4096R/6624859E 2012-11-18
    Key fingerprint = 1CA5 445E 7ABD BC21 AEC0 7B89 47D7 4EFF 6624 859E
uid Yoshihiro TAKAHASHI <nyan@furiru.org>
uid Yoshihiro TAKAHASHI <nyan@FreeBSD.org>
uid Yoshihiro TAKAHASHI <nyan@jp.FreeBSD.org>
sub 4096R/362726EA 2012-11-18
```

D.3.355. Sahil Tandon <sahil@FreeBSD.org>

```
pub 2048R/C016D977 2010-04-08
    Key fingerprint = 6AD2 BA99 8E3A 8DA6 DFC1 53CF DBD0 6001 C016 D977
uid Sahil Tandon <sahil@tandon.net>
```

uid Sahil Tandon <sahil@FreeBSD.org>
sub 2048R/F7776FBC 2010-04-08

D.3.356. TAKATSU Tomonari <tota@FreeBSD.org>

pub 1024D/67F58F29 2009-05-17
 Key fingerprint = 6940 B575 FC4A FA26 C094 279A 4B9B 6326 67F5 8F29
uid TAKATSU Tomonari <tota@FreeBSD.org>
sub 2048g/18B112CD 2009-05-17

D.3.357. Romain Tartière <romain@FreeBSD.org>

pub 3072R/5112336F 2010-04-09
 Key fingerprint = 8234 9A78 E7C0 B807 0B59 80FF BA4D 1D95 5112 336F
uid Romain Tartière <romain@blogreen.org>
uid Romain Tartière (FreeBSD) <romain@FreeBSD.org>
sub 3072R/C1B2B656 2010-04-09
sub 3072R/8F8125F4 2010-04-09

D.3.358. Sylvio Cesar Teixeira <sylvio@FreeBSD.org>

pub 2048R/AA7395A1 2009-10-28
 Key fingerprint = B319 6AAF 0016 4308 6D93 E652 3C5F 21A2 AA73 95A1
uid Sylvio Cesar Teixeira (My key) <sylvio@FreeBSD.org>
sub 2048R/F758F556 2009-10-28

D.3.359. Ion-Mihai Tetcu <itetcu@FreeBSD.org>

pub 4096R/29597D20 2013-05-02
 Key fingerprint = AB6F 39B6 605D E6B7 0D54 ED3D BCA2 129A 2959 7D20
uid Ion-Mihai Tetcu (FreeBSD Committer key) <itetcu@FreeBSD.org>
sub 4096R/EC9E17E3 2013-05-02

D.3.360. Mikhail Teterin <mi@FreeBSD.org>

pub 1024R/3FC71479 1995-09-08 Mikhail Teterin <mi@aldan.star89.galstar.com>
 Key fingerprint = 5F 15 EA 78 A5 40 6A 0F 14 D7 D9 EA 6E 2B DA A4

D.3.361. Gordon Tetlow <gordon@FreeBSD.org>

```
pub 1024D/357D65FB 2002-05-14 Gordon Tetlow <gordont@gnf.org>
   Key fingerprint = 34EF AD12 10AF 560E C3AE CE55 46ED ADF4 357D 65FB
uid                                Gordon Tetlow <gordon@FreeBSD.org>
sub 1024g/243694AB 2002-05-14
```

D.3.362. Lars Thegler <lth@FreeBSD.org>

```
pub 1024D/56B0CA08 2004-05-31 Lars Thegler <lth@FreeBSD.org>
   Key fingerprint = ABAE F98C EA78 1C8D 6FDD CB27 1CA9 5A63 56B0 CA08
uid                                Lars Thegler <lars@thegler.dk>
sub 1024g/E8C58EF3 2004-05-31
```

D.3.363. Jase Thew <jase@FreeBSD.org>

```
pub 3072R/3EEAF1EB 2012-05-30
   Key fingerprint = F5FB 959F CF1B 6550 054E 2819 A484 BCDB 3EEA F1EB
uid                                Jase Thew (FreeBSD) <jase@FreeBSD.org>
uid                                Jase Thew <freebsd@beardz.net>
```

D.3.364. David Thiel <lth@FreeBSD.org>

```
pub 1024D/A887A9B4 2006-11-30 [expires: 2011-11-29]
   Key fingerprint = F08F 6A12 738F C9DF 51AC 8C62 1E30 7CBE A887 A9B4
uid                                David Thiel <lth@FreeBSD.org>
sub 2048g/B9BD92C5 2006-11-30 [expires: 2011-11-29]
```

D.3.365. Fabien Thomas <fabient@FreeBSD.org>

```
pub 1024D/07745930 2009-03-16
   Key fingerprint = D8AC EFA2 2FBD 7788 9628 4E8D 3F35 3B88 0774 5930
uid                                Fabien Thomas <fabient@FreeBSD.org>
sub 2048g/BC173395 2009-03-16
```

D.3.366. Thierry Thomas <thierry@FreeBSD.org>

```
pub 1024D/C71405A2 1997-10-11
   Key fingerprint = 3BB8 F358 C2F1 776C 65C9 AE51 73DE 698C C714 05A2
uid                                Thierry Thomas <thierry@pompo.net>
uid                                Thierry Thomas <tthomas@mail.dotcom.fr>
uid                                Thierry Thomas (FreeBSD committer) <thierry@FreeBSD.org>
sub 1024R/C5529925 2003-11-26
sub 2048g/05CF3992 2008-02-05
```

D.3.367. Andrew Thompson <thompsa@FreeBSD.org>

```
pub 1024D/BC6B839B 2005-05-05
    Key fingerprint = DE74 3F49 B97C A170 C8F1 8423 CAB6 9D57 BC6B 839B
uid Andrew Thompson <thompsa@freebsd.org>
uid Andrew Thompson <andy@fud.org.nz>
sub 2048g/92E370FB 2005-05-05
```

D.3.368. Florent Thoumie <flz@FreeBSD.org>

```
pub 1024D/5147DCF4 2004-12-04
    Key fingerprint = D203 AF5F F31A 63E2 BFD5 742B 3311 246D 5147 DCF4
uid Florent Thoumie (FreeBSD committer address) <flz@FreeBSD.org>
uid Florent Thoumie (flz) <florent@thoumie.net>
uid Florent Thoumie (flz) <flz@xbsd.org>
uid [jpeg image of size 1796]
sub 2048g/15D930B9 2004-12-04
```

D.3.369. Jilles Tjoelker <jilles@FreeBSD.org>

```
pub 4096R/D5AE6220 2011-07-02
    Key fingerprint = 4AF5 F1CC BDD7 700B F005 79A4 A2C4 C4D4 D5AE 6220
uid Jilles Tjoelker <jilles@stack.nl>
uid Jilles Tjoelker <tjoelker@zonnet.nl>
uid Jilles Tjoelker (FreeBSD) <jilles@FreeBSD.org>
sub 4096R/14CB5775 2011-07-02
```

D.3.370. Ganbold Tsagaankhuu <ganbold@FreeBSD.org>

```
pub 1024D/78F6425E 2008-02-26 [expires: 2013-02-24]
    Key fingerprint = 9B8E DC41 D3F4 F7FC D8EA 417C D4F7 2AEF 78F6 425E
uid Ganbold <ganbold@freebsd.org>
sub 2048g/716FCBF9 2008-02-26 [expires: 2013-02-24]
```

D.3.371. Michael Tuexen <tuexen@FreeBSD.org>

```
pub 1024D/04EEDABE 2009-06-08
    Key fingerprint = 493A CCB8 60E6 5510 A01D 360E 8497 B854 04EE DABE
uid Michael Tuexen <tuexen@FreeBSD.org>
sub 2048g/F653AA03 2009-06-08
```

D.3.372. Andrew Turner <andrew@FreeBSD.org>

```
pub 2048R/31B31614 2010-07-01
    Key fingerprint = 08AC 2C57 F14F FDD1 2232 B5CD AA16 EFB8 31B3 1614
uid      Andrew Turner <andrew@freebsd.org>
uid      Andrew Turner <andrew@fubar.geek.nz>
sub 2048R/9ACBF138 2010-07-01
```

D.3.373. Hajimu UMEMOTO <ume@FreeBSD.org>

```
pub 1024D/BF9071FE 2005-03-17
    Key fingerprint = 1F00 0B9E 2164 70FC 6DC5 BF5F 04E9 F086 BF90 71FE
uid      Hajimu UMEMOTO <ume@mahoroba.org>
uid      Hajimu UMEMOTO <ume@FreeBSD.org>
uid      Hajimu UMEMOTO <ume@jp.FreeBSD.org>
sub 2048g/748DB3B0 2005-03-17
```

D.3.374. Stephan Uphoff <ups@FreeBSD.org>

```
pub 2048R/D684B04A 2004-10-06 Stephan Uphoff <ups@freebsd.org>
    Key fingerprint = B5D2 04AE CA8F 7055 7474 3C85 F908 7F55 D684 B04A
uid      Stephan Uphoff <ups@tree.com>
sub 2048R/A15F921B 2004-10-06
```

D.3.375. Bryan Venteicher <bryanv@FreeBSD.org>

```
pub 4096R/E97DB7DB 2012-11-05
    Key fingerprint = 0F8F 11EF F4D2 EDCA ECEA CB16 744C BF25 E97D B7DB
uid      Bryan Venteicher (DITC) <bryanv@daemoninthecloset.org>
uid      Bryan Venteicher (FreeBSD) <bryanv@freebsd.org>
sub 4096R/2EBC1A46 2012-11-05
```

D.3.376. Jacques Vidrine <nectar@FreeBSD.org>

```
pub 2048R/33C1627B 2001-07-05 Jacques A. Vidrine <nectar@celabo.org>
    Key fingerprint = CB CE 7D A0 6E 01 DC 61 E5 91 0A BE 79 17 D3 82
uid      Jacques A. Vidrine <jvidrine@verio.net>
uid      Jacques A. Vidrine <n@nectar.com>
uid      Jacques A. Vidrine <jacques@vidrine.cc>
uid      Jacques A. Vidrine <nectar@FreeBSD.org>
uid      Jacques A. Vidrine <n@nectar.cc>

pub 1024D/1606DB95 2001-07-05 Jacques A. Vidrine <nectar@celabo.org>
    Key fingerprint = 46BC EA5B F70A CC81 5332 0832 8C32 8CFF 1606 DB95
uid      Jacques A. Vidrine <jvidrine@verio.net>
uid      Jacques A. Vidrine <n@nectar.com>
```

```
uid          Jacques A. Vidrine <jacques@vidrine.cc>
uid          Jacques A. Vidrine <nectar@FreeBSD.org>
uid          Jacques A. Vidrine <n@nectar.cc>
sub 2048g/57EDEA6F 2001-07-05
```

D.3.377. Alberto Villa <avilla@FreeBSD.org>

```
pub 1024R/44350A8B 2010-01-24
   Key fingerprint = F740 CE4E EDDD DA9B 4A1B 1445 DF18 82EA 4435 0A8B
uid          Alberto Villa <avilla@FreeBSD.org>
sub 1024R/F7C8254C 2010-01-24
```

D.3.378. Nicola Vitale <nivit@FreeBSD.org>

```
pub 1024D/F11699E5 2006-12-05
   Key fingerprint = 2C17 C591 2C6D 82BD F3DB F1BF 8FC9 6763 F116 99E5
uid          Nicola Vitale (Public key for nivit@FreeBSD.org) <nivit@FreeBSD.org>
sub 2048g/4C90805D 2006-12-05
```

D.3.379. Ivan Voras <ivoras@FreeBSD.org>

```
pub 1024D/569C05C8 2000-05-24
   Key fingerprint = AB9A A555 C47C B61D BF83 154C 95D9 C041 569C 05C8
uid          Ivan Voras <ivoras@fer.hr>
uid          Ivan Voras <ivan.voras@fer.hr>
uid          Ivan Voras <ivoras@geri.cc.fer.hr>
uid          [jpeg image of size 4567]
uid          Ivan Voras <ivoras@sharanet.org>
uid          Ivan Voras <ivoras@gmail.com>
uid          Ivan Voras <ivoras@yahoo.com>
uid          Ivan Voras <ivoras@freebsd.org>
uid          Ivan Voras <ivan.voras@zg.t-com.hr>
sub 1536g/149FDD60 2000-05-24
```

D.3.380. Stefan Walter <stefan@FreeBSD.org>

```
pub 3072R/12B9E0B3 2003-03-06
   Key fingerprint = 85D8 6A49 22C7 6CD9 B011 5D6A 5691 111B 12B9 E0B3
uid          Stefan Walter <stefan@freebsd.org>
uid          Stefan Walter <sw@gegenunendlich.de>
sub 3072R/6D35457A 2003-03-06
```

D.3.381. Kai Wang <kaiw@FreeBSD.org>

```

pub 1024D/AEB910EB 2006-09-27
    Key fingerprint = 3534 10A3 F143 B760 EF3E BEDF 8509 6A06 AEB9 10EB
uid      Kai Wang <kaiw@FreeBSD.org>
uid      Kai Wang <kaiw@student.chalmers.se>
uid      Kai Wang <kaiwang27@gmail.com>
uid      Kai Wang <kaiw27@gmail.com>
sub 2048g/1D5AA4DD 2006-09-27

```

D.3.382. Adam Weinberger <adamw@FreeBSD.org>

```

pub 2048D/C57CF3A8 2012-11-15
    Key fingerprint = CCD9 F28A BD1D 50A1 8D08 18A7 F48B B195 C57C F3A8
uid      Adam Weinberger (FreeBSD) <adamw@FreeBSD.org>
uid      Adam Weinberger (adamw.org) <adamw@adamw.org>
sub 2048g/9C6D0E30 2012-11-15

```

D.3.383. Peter Wemm <peter@FreeBSD.org>

```

pub 1024D/7277717F 2003-12-14 Peter Wemm <peter@wemm.org>
    Key fingerprint = 622B 2282 E92B 3BAB 57D1 A417 1512 AE52 7277 717F
uid      Peter Wemm <peter@FreeBSD.ORG>
sub 1024g/8B40D9D1 2003-12-14
pub 1024R/D89CE319 1995-04-02 Peter Wemm <peter@netplex.com.au>
    Key fingerprint = 47 05 04 CA 4C EE F8 93 F6 DB 02 92 6D F5 58 8A
uid      Peter Wemm <peter@perth.dialix.oz.au>
uid      Peter Wemm <peter@haywire.dialix.com>

```

D.3.384. Nathan Whitehorn <nwhitehorn@FreeBSD.org>

```

pub 1024D/FC118258 2008-07-03
    Key fingerprint = A399 BEA0 8D2B 63B3 47B5 056D 8513 5B96 FC11 8258
uid      Nathan Whitehorn <nwhitehorn@freebsd.org>
uid      Nathan Whitehorn <nwhitehorn@icecube.wisc.edu>
uid      Nathan Whitehorn <nwhitehorn@physics.wisc.edu>
uid      Nathan Whitehorn <whitehorn@wisc.edu>
sub 2048g/EDB55363 2008-07-03

```

D.3.385. Martin Wilke <miwi@FreeBSD.org>

```

pub 1024D/B1E6FCE9 2009-01-31
    Key fingerprint = C022 7D60 F598 8188 2635 0F6E 74B2 4884 B1E6 FCE9
uid      Martin Wilke <miwi@FreeBSD.org>
sub 4096g/096DA69D 2009-01-31

```

D.3.386. Nate Williams <nate@FreeBSD.org>

```
pub 1024D/C2AC6BA4 2002-01-28 Nate Williams (FreeBSD) <nate@FreeBSD.org>
   Key fingerprint = 8EE8 5E72 8A94 51FA EA68 E001 FFF9 8AA9 C2AC 6BA4
sub 1024g/03EE46D2 2002-01-28
```

D.3.387. Steve Wills <swills@FreeBSD.org>

```
pub 2048R/207B1BA1 2010-09-02 [expires: 2011-09-02]
   Key fingerprint = 98FA 414A 5C2A 0EF9 CFD0 AD0D F5CF 62B3 207B 1BA1
uid Steve Wills <swills@freebsd.org>
uid Steve Wills <steve@mouf.net>
sub 2048R/E9B254FD 2010-09-02 [expires: 2011-09-02]
```

D.3.388. Thomas Wintergerst <twinterg@FreeBSD.org>

```
pub 1024D/C45CB978 2006-01-08
   Key fingerprint = 04EE 8114 7C6D 22CE CDC8 D7F8 112D 01DB C45C B978
uid Thomas Wintergerst <twinterg@gmx.de>
uid Thomas Wintergerst <twinterg@freebsd.org>
uid Thomas Wintergerst
uid Thomas Wintergerst <thomas.wintergerst@nord-com.net>
uid Thomas Wintergerst <thomas.wintergerst@materna.de>
sub 2048g/3BEBEF8A 2006-01-08
sub 1024D/8F631374 2006-01-08
sub 2048g/34F631DC 2006-01-08
```

D.3.389. Garrett Wollman <wollman@FreeBSD.org>

```
pub 1024D/0B92FAEA 2000-01-20 Garrett Wollman <wollman@FreeBSD.org>
   Key fingerprint = 4627 19AF 4649 31BF DE2E 3C66 3ECF 741B 0B92 FAEA
sub 1024g/90D5EBC2 2000-01-20
```

D.3.390. Jörg Wunsch <joerg@FreeBSD.org>

```
pub 1024D/69A85873 2001-12-11 Joerg Wunsch <j@uriah.heep.sax.de>
   Key fingerprint = 5E84 F980 C3CA FD4B B584 1070 F48C A81B 69A8 5873
pub 1024D/69A85873 2001-12-11 Joerg Wunsch <j@uriah.heep.sax.de>
uid Joerg Wunsch <joerg_wunsch@interface-systems.de>
uid Joerg Wunsch <joerg@FreeBSD.org>
uid Joerg Wunsch <j@ida.interface-business.de>
sub 1024g/21DC9924 2001-12-11
```

D.3.391. David Xu <davidxu@FreeBSD.org>

```
pub 1024D/48F2BDAB 2006-07-13 [expires: 2009-07-12]
    Key fingerprint = 7182 434F 8809 A4AF 9AE8 F1B5 12F6 3390 48F2 BDAB
uid David Xu <davidxu@freebsd.org>
sub 4096g/ED7DB38A 2006-07-13 [expires: 2009-07-12]
```

D.3.392. Maksim Yevmenkin <emax@FreeBSD.org>

```
pub 1024D/F050D2DD 2003-10-01 Maksim Yevmenkin <m_evmenkin@yahoo.com>
    Key fingerprint = 8F3F D359 E318 5641 8C81 34AD 791D 53F5 F050 D2DD
```

D.3.393. Bjoern A. Zeeb <bz@FreeBSD.org>

```
pub 1024D/3CCF1842 2007-02-20
    Key fingerprint = 1400 3F19 8FEF A3E7 7207 EE8D 2B58 B8F8 3CCF 1842
uid Bjoern A. Zeeb <bz@zabbadoz.net>
uid Bjoern A. Zeeb <bzeeb@zabbadoz.net>
uid Bjoern A. Zeeb <bz@FreeBSD.org>
uid Bjoern A. Zeeb <bzeeb-lists@lists.zabbadoz.net>
sub 4096g/F36BDC5D 2007-02-20
```

D.3.394. Niclas Zeising <zeising@FreeBSD.org>

```
pub 4096R/EA4BF1EC 2012-11-28 [expires: 2013-12-31]
    Key fingerprint = A8DE D126 D346 E9CB 6176 AECB 0401 4392 EA4B F1EC
uid Niclas Zeising <zeising@daemonic.se>
uid Niclas Zeising (FreeBSD Project) <zeising@freebsd.org>
uid Niclas Zeising (Lysator ACS) <zeising@lysator.liu.se>
sub 4096R/BB8B5551 2012-11-29 [expires: 2013-12-31]
sub 4096R/B8D43CD2 2012-11-29 [expires: 2013-12-31]
```

D.3.395. Alexey Zelkin <phantom@FreeBSD.org>

```
pub 1024D/9196B7D9 2002-01-28 Alexey Zelkin <phantom@FreeBSD.org>
    Key fingerprint = 4465 F2A4 28C1 C2E4 BB95 1EA0 C70D 4964 9196 B7D9
sub 1024g/E590ABA4 2002-01-28
```

D.3.396. Sepherosa Ziehau <sephe@FreeBSD.org>

```
pub 2048R/3E51FB42 2005-10-21
    Key fingerprint = 5F47 3861 7ABA 8773 9E32 0474 5C33 841C 3E51 FB42
uid Sepherosa Ziehau (freebsd) <sephe@freebsd.org>
uid Sepherosa Ziehau (sephe) <sepherosa@gmail.com>
```

sub 2048R/7AA31321 2005-10-21

D.3.397. Andrey Zonov <zont@FreeBSD.org>

pub 2048R/E8A68B1C 2012-08-17 [expires: 2016-08-17]
Key fingerprint = 3DFF AA2F C10A A979 2FB9 A764 F145 4BB6 E8A6 8B1C
uid Andrey Zonov <zont@FreeBSD.org>
uid Andrey Zonov <andrey@zonov.org>
sub 2048R/57FC2BD3 2012-08-17 [expires: 2016-08-17]

A FreeBSD-s szakkifejezések gyűjteménye

Ebben a szójegyzékben azok a fogalmak és rövidítések szerepelnek, amelyekkel a FreeBSD-s közösségen belül és a hozzá tartozó különböző leírásokban találkozhatunk.

A

ACL

Lásd: Access Control List

ACPI

Lásd: Advanced Configuration and Power Interface

AMD

Lásd: Automatic Mount Daemon

AML

Lásd: ACPI Machine Language

API

Lásd: Application Programming Interface

APIC

Lásd: Advanced Programmable Interrupt Controller

APM

Lásd: Advanced Power Management

APOP

Lásd: Authenticated Post Office Protocol

ASL

Lásd: ACPI Source Language

ATA

Lásd: Advanced Technology Attachment

ATM

Lásd: Asynchronous Transfer Mode

ACPI Machine Language

Olyan pszeudókód, amit egy ACPI szabvánnyal kompatibilis operációs rendszerben megtalálható virtuális géppel lehet értelmezni. Feladata a rendelkezésre álló hardveren az operációs rendszer felé dokumentált felület kialakítása.

ACPI Source Language

Az a programozási nyelv, amiben az AML-kódok íródnak.

Access Control List

Egy objektumhoz, például egy állományhoz vagy hálózati eszközhöz tartozó engedélyeket tartalmazó felsorolás.

Advanced Configuration and Power Interface

Az a specifikáció, aminek köszönhetően a hardver egy absztrakt felületet képes nyújtani az operációs rendszer számára. Ezen a felületen keresztül tudja az operációs rendszer elérni a rendelkezésre álló hardvert annak konkrét ismerete nélkül. Az ACPI a korábban az APM, PNPBIOS és a hozzájuk hasonló megoldások által szolgáltatott lehetőségeket igyekszik kiterjeszteni és felülmúlni. Ennek keretében lehetőséget ad többek közt az energiafogyasztás szabályozására, az energiatakarék mód aktiválására, az eszközök ki- és bekapcsolására stb.

Application Programming Interface

Eljárások, protokollok és segédprogramok összesége, melyek egy vagy több programrész között írják le az általános összefüggéseket: hogyan, mikor és miért kell összedolgozniuk, illetve milyen adatokat osszanak meg egymás között vagy milyen adatokkal dolgozzanak.

Advanced Power Management

Egy olyan API, amely lehetővé teszi az operációs rendszer számára, hogy a BIOS-szal együtt energiagazdálkodást tudjon megvalósítani. A legtöbb esetben azonban már az APM-et leváltotta a sokkal általánosabb és kidolgozottabb ACPI specifikáció.

Advanced Programmable Interrupt Controller

Advanced Technology Attachment

Asynchronous Transfer Mode

Authenticated Post Office Protocol

Automatic Mount Daemon

Egy olyan démon, ami önműködően csatlakoztatja az állományrendszereket, amikor azokon valamilyen állományt vagy könyvtárat el akarunk érni.

B

BAR

Lásd: Base Address Register

BIND

Lásd: Berkeley Internet Name Domain

BIOS

Lásd: Basic Input/Output System

BSD

Lásd: Berkeley Software Distribution

Base Address Register

Egy PCI eszköz címtartományának megadásáért felelős regiszterek.

Basic Input/Output System

A BIOS meghatározása némileg a környezetétől is függ. Egyesek szerint BIOS az a ROM chip, ami a szoftver és hardver közti kapcsolatot megteremtő alapvető rutinokat tartalmazza. Mások szerint viszont azok a chipen tárolt rutinok, amelyek a rendszer betöltéséért felelősek. De akadnak olyanok is, akik ilyenkor arra a képernyőre gondolnak, amin a rendszer betöltésének folyamatát tudjuk beállítani. Noha a BIOS leginkább a PC típusú rendszerekre jellemző, más esetekben is találkozhatunk hasonlóval.

Berkeley Internet Name Domain

A névfeloldásért felelős DNS protokollok egyik implementációja.

Berkeley Software Distribution

A Kaliforniai Egyetem (Berkeley) (<http://www.berkeley.edu>) számítógépes rendszerekkel foglalkozó kutatócsoportja (CSRG) ebben foglalta össze az AT&T 32V UNIX rendszerén végzett változtatásait és javításait. Maga a FreeBSD is ennek az egyik leszármazottja.

Bikeshed Building

A “bikeshed building”, vagyis a “biciklitároló építés” az a jelenség, amikor egy egyszerűbb témához mindenki hozzá akar szólni, miközben egy sokkal bonyolultabb témával alig vagy egyáltalán nem foglalkoznak. Ennek kialakulásáról részletesebben a GYIK (http://www.FreeBSD.org/doc/hu_HU.ISO8859-2/books/faq/misc.html#BIKESHED-PAINTING)-ban lehet olvasni.

C

CD

Lásd: Carrier Detect

CHAP

Lásd: Challenge Handshake Authentication Protocol

CLIP

Lásd: Classical IP over ATM

COFF

Lásd: Common Object File Format

CPU

Lásd: Central Processing Unit

CTS

Lásd: Clear To Send

CVS

Lásd: Concurrent Versions System

Carrier Detect

A kommunikációs csatorna létrejöttét jelző RS232C szabványú jel.

Central Processing Unit

Másik nevén processzor. Lényegében ez a számítógép agya, ahol a különféle számítások történnek. Rengeteg különböző architektúrája és utasításkészlete lehet. Közülük a legismertebbek az Intel x86 és annak leszármazottai, valamint a Sun SPARC, PowerPC és Alpha.

Challenge Handshake Authentication Protocol

A felhasználók hitelesítésére használt módszer, amely a kliens és a szerver közt megosztott titkos információkon alapszik.

Classical IP over ATM

Clear To Send

A távoli rendszer számára a küldést engedélyező RS232C szabványú jel.

Lásd még: Request To Send.

Common Object File Format

Concurrent Versions System

Egy verziókezelő rendszer, aminek használatával egyszerre több változatot tudunk nyilvántartani és használni adott állományokból. A CVS segítségével képesek vagyunk egy vagy több változtatást kivonni, összefésülni és visszavonni, valamint nyomon követhetjük, hogy melyiküket ki, mikor és miért hajtotta végre.

D

DAC

Lásd: Discretionary Access Control

DDB

Lásd: Debugger

DES

Lásd: Data Encryption Standard

DHCP

Lásd: Dynamic Host Configuration Protocol

DNS

Lásd: Domain Name System

DSDT

Lásd: Differentiated System Description Table

DSR

Lásd: Data Set Ready

DTR

Lásd: Data Terminal Ready

DVMRP

Lásd: Distance-Vector Multicast Routing Protocol

Discretionary Access Control

Data Encryption Standard

Az információ titkosítására szánt módszer, amelyet általában a UNIX-os jelszavak és crypt(3) funkció használ.

Data Set Ready

Ezt az RS232C szabványú jelet küldi egy modem a számítógépünknek vagy a terminálunknak, amikor készen áll az adatok fogadására és küldésére.

Lásd még: Data Terminal Ready.

Data Terminal Ready

Ezt az RS232C szabványú jelet küldi számítógépünk vagy a terminálunk a modemnek, amikor készen áll az adatok fogadására és küldésére.

Debugger

A rendszermagban megtalálható interaktív nyomkövetési lehetőség, amin keresztül meg tudjuk vizsgálni rendszerünk aktuális állapotát. Leggyakrabban a rendszer összeomlásáért felelős körülmények elemzésében alkalmazzák.

Differentiated System Description Table

Egy olyan ACPI táblázat, amely az alaprendszerről nyújt alapvető konfigurációs információkat.

Distance-Vector Multicast Routing Protocol

Domain Name System

Az internetes címek (pl. levelezes.valami.net) emberek és gépek által is olvasható formája közti leképezéséért felelős rendszer.

Dynamic Host Configuration Protocol

A számítógépek IP-címeinek szerveren keresztüli dinamikus kiosztásáért felelős protokoll. Az így keletkező cím alapú hozzárendelést “bérletnek” (“lease”) nevezzük.

E

ECOFF

Lásd: Extended COFF

ELF

Lásd: Executable and Linking Format

ESP

Lásd: Encapsulated Security Payload

Encapsulated Security Payload

Executable and Linking Format

Extended COFF

F

FADT

Lásd: Fixed ACPI Description Table

FAT

Lásd: File Allocation Table

FAT16

Lásd: File Allocation Table (16-bit)

FTP

Lásd: File Transfer Protocol

File Allocation Table

File Allocation Table (16-bit)

File Transfer Protocol

A TCP felett implementált magasabb szintű protokollok családjának egyik tagja, aminek segítségével állományokat tudunk átmásolni egy TCP/IP-hálózaton keresztül.

Fixed ACPI Description Table

G

GUI

Lásd: Graphical User Interface

Giant

Annak a kölcsönös kizárásért felelős megoldásnak (alvó (sleep) mutex-nek) a neve, ami a rendszermag erőforrásainak jelentős részét védi. Amikor még a számítógépek csupán néhány programot futtattak egyetlen hálózati kártyával és általában egyetlen processzoron, akkor még elegendő volt egy egyszerűbb zárolási mechanizmus használata, azonban napjainkban ez már egy elfogadhatatlanul szűk keresztmetszetet képez. A FreeBSD fejlesztői folyamatosan dolgoznak, hogy ezt olyan zárolásokkal váltsák fel, amelyek csak az egyes erőforrásokat védik. Ennek köszönhetően sokkal nagyobb fokú párhuzamosítás érhető el mind az egyprocesszoros mind pedig a többprocesszoros rendszerekben egyaránt.

Graphical User Interface

Olyan rendszer, ahol a felhasználó és a számítógép grafikus megoldásokon keresztül érintkezik.

H

HTML

Lásd: HyperText Markup Language

HUP

Lásd: HangUp

HangUp

HyperText Markup Language

Honlapok előállítására használt jelölőnyelv.

I

I/O

Lásd: Input/Output

IASL

Lásd: Intel's ASL compiler

IMAP

Lásd: Internet Message Access Protocol

IP

Lásd: Internet Protocol

IPFW

Lásd: IP Firewall

IPP

Lásd: Internet Printing Protocol

IPv4

Lásd: IP Version 4

IPv6

Lásd: IP Version 6

ISP

Lásd: Internet Service Provider

IP Firewall

IP Version 4

Az IP protokoll 4-es változata, ahol 32 biten adunk meg címeket. Ez a változat még napjainkban is széles körben alkalmazott, azonban lassanként felváltja az IPv6.

Lásd még: IP Version 6.

IP Version 6

Az új IP protokoll. Azért alkották meg, mert az IPv4 által felkínált címtér már túlságosan kicsinek bizonyult. 128 bites címekkel dolgozik.

Input/Output

Intel's ASL compiler

Az Intel által kifejlesztett fordítóprogram, amivel ASL-programokat lehet AML-kódra fordítani.

Internet Message Access Protocol

A levelező szervereken tárolt elektronikus levelek elérésére használt protokoll, aminek egyik fontos jellemzője, hogy az elolvasott leveleket a szerveren tartja és nem tölti le a levelező klienssel.

Lásd még: Post Office Protocol Version 3.

Internet Printing Protocol

Internet Protocol

Csomagok átküldését leíró protokoll, amire egész internet épül. Eredetileg az Egyesült Államok Védelmi Minisztériuma számára készült, és a TCP/IP protokollkészlet egyik meghatározó eleme. Enélkül az internet nem nyerte volna el mai alakját. Részletesebb információkért ld. az RFC 791 (<ftp://ftp.rfc-editor.org/in-notes/rfc791.txt>).

Internet Service Provider

Egy olyan cég, ami lehetőséget kínál az internet elérésére.

K

KAME

A KAME japánul “teknőst” jelent, de informatikai körökben ezt gyakran a KAME projekttel (<http://www.kame.net/>) azonosítják, amely az IPv6 implementációján dolgozik.

KDC

Lásd: Key Distribution Center

KLD

Lásd: Kernel ld(1)

KSE

Lásd: Kernel Scheduler Entities

KVA

Lásd: Kernel Virtual Address

Kbps

Lásd: Kilo Bits Per Second

Kernel ld(1)

Egy olyan módszer, aminek segítségével a FreeBSD rendszermag funkcionalitását anélkül tudjuk dinamikusan bővíteni, hogy a újra kellene indítanunk hozzá a rendszerünket.

Kernel Scheduler Entities

A rendszermag által támogatott szálkezelési rendszer. Ennek pontosabb részleteit ld. a hozzá tartozó projekt honlapján (<http://www.FreeBSD.org/kse>).

Kernel Virtual Address

Key Distribution Center

Kilo Bits Per Second

A sávszélesség (vagyis egy adott idő alatt mennyi adatot vagyunk képesek átküldeni) meghatározására használt mérték. Itt a Kilo helyett még szerepelhet a Mega, Giga, Tera és így tovább.

L

LAN

Lásd: Local Area Network

LOR

Lásd: Lock Order Reversal

LPD

Lásd: Line Printer Daemon

Line Printer Daemon

Local Area Network

Egy viszonylag kis környezetben, például irodában, otthon stb. használt hálózat.

Lock Order Reversal

A FreeBSD rendszermagja az erőforrások megfelelő zárolásával igyekszik megosztani azokat. A zárolási hibák keletkező holtpontok felderítésére a FreeBSD-CURRENT rendszermagokban található (de a kiadásokból már eltávolított) egy zárolásokat ellenőrző futás idejű rendszer, aminek a neve witness(4). (A witness(4) jelen pillanatban kissé még szigorú, ezért előfordulhat, hogy vakriasztást ad.) A tőle származó valós jelentésekben olvashatjuk, hogy “ha pórul jártunk volna, akkor most itt lett volna egy holtpont”.

Az ilyen hibákat általában gyorsan kijavítják, ezért mielőtt egy ilyen hibát beküldenénk, nézzünk szét a <http://lists.FreeBSD.org/mailman/listinfo/freebsd-current> címen és az észlelt LOR-ok (<http://sources.zabbadoz.net/freebsd/lor.html>) honlapján.

M

MAC

Lásd: Mandatory Access Control

MADT

Lásd: Multiple APIC Description Table

MFC

Lásd: Merge From Current

MFP4

Lásd: Merge From Perforce

MFS

Lásd: Merge From Stable

MIT

Lásd: Massachusetts Institute of Technology

MLS

Lásd: Multi-Level Security

MOTD

Lásd: Message Of The Day

MTA

Lásd: Mail Transfer Agent

MUA

Lásd: Mail User Agent

Mail Transfer Agent

A levelek továbbítására használt alkalmazás, melyek a BSD alaprendszerekben már régóta megtalálhatóak. Közülük manapság a Sendmail szerepel itt, de rajta kívül még több más MTA is létezik, mint például a postfix, qmail és az Exim.

Mail User Agent

Az elektronikus levelek megjelenítésére és írására alkalmas alkalmazás.

Mandatory Access Control

Massachusetts Institute of Technology

Merge From Current

A -CURRENT ágból származó valamelyik funkcionalitás vagy módosítás beolvasztása egy másik ágba, ami a legtöbb esetben a -STABLE.

Merge From Perforce

A Perforce repository-ból származó funkcionalitás vagy módosítás beolvasztása a -CURRENT ágba.

Lásd még: Perforce.

Merge From Stable

A FreeBSD fejlesztésének megszokott menete szerint egy változtatás először a -CURRENT ágba kerül be tesztelésre, majd csak ezt követően a -STABLE ágba. Esetenként azonban előfordul, hogy egy változtatás először a -STABLE ágba kerül, majd csak ezután a -CURRENT ágba.

Ezt a kifejezést használjuk abban az esetben is, amikor egy módosítást a -STABLE ágból olvasztunk be a biztonsági javításokat tartalmazó ágba.

Lásd még: Merge From Current.

Message Of The Day

Általában a bejelentkezéskor megjelenő üzenet, amiben valamilyen információt továbbítunk a rendszer felhasználói számára.

Multi-Level Security

Multiple APIC Description Table

N

NAT

Lásd: Network Address Translation

NDISulator

Lásd: Project Evil

NFS

Lásd: Network File System

NTFS

Lásd: New Technology File System

NTP

Lásd: Network Time Protocol

Network Address Translation

Egy olyan technikai megoldás, amelynek használata során az átjárón keresztül haladó IP-csomagok információt módosítják, és ezáltal lehetővé teszik az átjáró mögött levő gépek számára, hogy hatékonyan osztozzanak egyetlen IP-címen.

Network File System

New Technology File System

A Microsoft által kidolgozott állományrendszer, ami általuk fejlesztett “új technológiájú” operációs rendszerekben érhető el, tehát például a Windows 2000, Windows NT és Windows XP rendszerekben.

Network Time Protocol

A számítógépek óráinak hálózaton keresztüli egyeztetésének egyik módszere.

O

OBE

Lásd: Overtaken By Events

ODMR

Lásd: On-Demand Mail Relay

OS

Lásd: Operating System

On-Demand Mail Relay

Operating System

Programok, függvénykönyvtárak és segédprogramok összesége, amelyeken keresztül hozzá tudunk férni a számítógépben található hardverek által felkínált erőforrásokhoz. Napjaink operációs rendszerei egészen az egy időben egyetlen programot futtatni és egyetlen eszközt elérni képes rendszerektől a többfelhasználós, többfeladatos és egyszerre több programot is futtatni tudó, több ezer, egyenként különböző alkalmazásokat futtató felhasználót kiszolgáló rendszerekig terjedhet.

Overtaken By Events

Olyan javasolt változtatásra (hibajelentésre vagy egy új funkció igénylésére) utal, ami a legfrissebb változtatások, például a FreeBSD hálózati szabványainak megváltozása, az adott hardver elavulása stb. következtében már nem lényeges vagy nem érvényes.

P

p4

Lásd: Perforce

PAE

Lásd: Physical Address Extensions

PAM

Lásd: Pluggable Authentication Modules

PAP

Lásd: Password Authentication Protocol

PC

Lásd: Personal Computer

PCNSFD

Lásd: Personal Computer Network File System Daemon

PDF

Lásd: Portable Document Format

PID

Lásd: Process ID

POLA

Lásd: Principle Of Least Astonishment

POP

Lásd: Post Office Protocol

POP3

Lásd: Post Office Protocol Version 3

PPD

Lásd: PostScript Printer Description

PPP

Lásd: Point-to-Point Protocol

PPPoA

Lásd: PPP over ATM

PPPoE

Lásd: PPP over Ethernet

PPP over ATM

PPP over Ethernet

PR

Lásd: Problem Report

PXE

Lásd: Preboot eXecution Environment

Password Authentication Protocol

Perforce

A Perforce Software (<http://www.perforce.com/>) által fejlesztett forráskódkezelő termék, ami a CVS-nél jóval több lehetőséget kínál. Annak ellenére, hogy nem nyílt forráskódú, használata ingyenes olyan nyílt forráskódú projektek számára, mint amilyen a FreeBSD.

Egyes FreeBSD fejlesztők a Perforce repository-ban dolgoznak olyan kódokkal, amelyek használata a -CURRENT ágban túlságosan kockázatos lenne.

Personal Computer

Personal Computer Network File System Daemon

Physical Address Extensions

Egy olyan módszer, aminek segítségével egészen 64 GB-nyi központi memóriát tudunk elérni azokon a rendszereken, amelyek fizikailag csak 32 bites címtérrel rendelkeznek (és ezáltal a PAE nélkül csak 4 GB memóriát képesek használni).

Pluggable Authentication Modules

Point-to-Point Protocol

Pointy Hat

Egy misztikus eredetű fejrevaló, ami leginkább a számárfüles sapkához hasonlítható, és minden olyan FreeBSD committer jutalma, aki miatt nem fordul a rendszer, visszafele halad a verziók számozása, vagy bármilyen egyéb pusztítást végez a források között. Az ügyetlenebb committerek szép számmal be tudnak ilyeneket gyűjteni. Többnyire (csak?) humoros értelemben használják.

Portable Document Format

Post Office Protocol

Lásd még: Post Office Protocol Version 3.

Post Office Protocol Version 3

A levelező szerverken tárolt elektronikus levelek elérésére használatos protokoll, aminek egyik fontos jellemzője, hogy az elolvasandó leveleket a levelező kliens letölti, nem pedig a szerveren hagyja.

Lásd még: Internet Message Access Protocol.

PostScript Printer Description

Preboot eXecution Environment

Principle Of Least Astonishment

A FreeBSD fejlődése során igyekezni kell elkerülni a felhasználók elé tárt hirtelen változtatásokat. Például az `/etc/defaults/rc.conf` állományban található, rendszerindításért felelős változók átrendezése sérti “a legkisebb meglepetés elvét” (POLA). A fejlesztőknek tehát figyelembe kell venniük ezt az elvet, amikor a felhasználók számára is észlelhető változtatásokat hoznak létre.

Problem Report

A FreeBSD forrásában vagy dokumentációjában talált hiba leírása. Erről bővebben ld. a FreeBSD hibajelentések írása (http://www.FreeBSD.org/doc/hu_HU.ISO8859-2/articles/problem-reports/index.html) című cikket (angolul).

Process ID

A rendszerben egy adott futó programot egyértelműen azonosító szám, amivel hivatkozni tudunk rá és műveleteket végrehajtani vele.

Project Evil

A Bill Paul által készített NDISulator munkacíme, amivel a szerző elsősorban arra szeretett volna (filozófiai szemszögből) utalni, hogy milyen szörnyűséget kellett művelnie. Az NDISulator egy olyan speciális kompatibilitási modul, aminek révén a FreeBSD/i386 változatában képesek vagyunk a Microsoft Windows;trade; NDIS miniport hálózati meghajtóit. Általában csak ez az egyetlen módja a zárt forráskódú meghajtókkal rendelkező kártyák használatának. Ld. `src/sys/compat/ndis/subr_ndis.c`.

R

RA

Lásd: Router Advertisement

RAID

Lásd: Redundant Array of Inexpensive Disks

RAM

Lásd: Random Access Memory

RD

Lásd: Received Data

RFC

Lásd: Request For Comments

RISC

Lásd: Reduced Instruction Set Computer

RPC

Lásd: Remote Procedure Call

RS232C

Lásd: Recommended Standard 232C

RTS

Lásd: Request To Send

Random Access Memory

Revision Control System

A *Revision Control System* (RCS) azon egyik legrégebbi szoftverek egyike, amelyek egyszerű állományok esetén megvalósítanak valamilyen “verziókövetést”. Segítségével állományok különböző változatait tudjuk tárolni, visszatölteni, archiválni, naplózni, azonosítani és összefésülni. Az RCS több egymással együttműködő

apró segédprogram összesége. Tudása több tekintetben is alulmarad a mai modern verziókövető rendszerekéhez, mint például a CVS-hez vagy a Subversionhöz képest, azonban kevés állomány esetén nagyon egyszerűen telepíthető, konfigurálható és használható. Az RCS valamilyen változata minden fontosabb UNIX-szerű operációs rendszerben megtalálható.

Received Data

Az az RS232C szabványú tű vagy vezeték, amin keresztül az adat érkezik.

Lásd még: Transmitted Data.

Recommended Standard 232C

A soros vonali eszközök közti kommunikációt leíró szabvány.

Reduced Instruction Set Computer

Olyan megközelítés a processzorok tervezésében, ahol a hardver által végezhető műveletek ugyan leegyszerűsítettek, de a lehető legjobban általánosítottak. Ezzel csökkenthető az energiafogyasztás, kevesebb tranzistorra van szükség és egyes esetekben akár nagyobb teljesítményt és megnövekedett kódsűrűséget is eredményezhet. RISC processzorok például az Alpha, SPARC, ARM és PowerPC.

Redundant Array of Inexpensive Disks

Remote Procedure Call

repocopy

Lásd: Repository Copy

Repository Copy

Állományok közvetlen másolása a CVS repository-n belül.

Repocopy nélkül a committer csak úgy tudná a repository egyik részéből a másikkra áthelyezni az állományokat, ha először a `cvsv add` paranccsal felvenné ezeket az új helyre, majd a `cvsv rm` paranccsal törölné a régi helyről.

Ennek a megoldásnak egyik hátránya, hogy az állományokhoz tartozó előzmények (tehát a CVS naplókban szereplő bejegyzések) ilyenkor nem másolódnak át az új helyre. Mivel a FreeBSD projekt ezeket viszont nagyon fontosnak tartja, ezért ehelyett gyakran a “repository copy” módszerét alkalmazzák. Ennek folyamán a repository-k karbantartásáért felelős tagok (repository mesterek) fogják a `cvsv(1)` használata helyett átmásolni az állományokat, közvetlenül a repository-n belül.

Request For Comments

Az internet működéséhez kapcsolódó szabványok, protokollok és egyéb leírását tartalmazó dokumentumok.
Ld. www.rfc-editor.org (<http://www.rfc-editor.org/>).

Gyakran viszont abban az értelemben is használják, amikor valaki szeretné kikérni a véleményét egy általa javasolt módosításról.

Request To Send

Egy RS232C szabványú jel, amivel megkérjük a távoli rendszert az adatok átküldésének megkezdésére.

Lásd még: Clear To Send.

Router Advertisement

S

SCI

Lásd: System Control Interrupt

SCSI

Lásd: Small Computer System Interface

SG

Lásd: Signal Ground

SMB

Lásd: Server Message Block

SMP

Lásd: Symmetric MultiProcessor

SMTP

Lásd: Simple Mail Transfer Protocol

SMTP AUTH

Lásd: SMTP Authentication

SSH

Lásd: Secure Shell

STR

Lásd: Suspend To RAM

SVN

Lásd: Subversion

SMTP Authentication

Server Message Block

Signal Ground

Egy RS232 szabványú tű vagy vezeték, ami a jelek számára a referencia földet adja.

Simple Mail Transfer Protocol

Secure Shell

Small Computer System Interface

Subversion

A Subversion egy CVS-hez hasonló verziókezelő rendszer, némileg bővebb tudással.

Lásd még: Concurrent Versions System.

Suspend To RAM

Symmetric MultiProcessor

System Control Interrupt

T

TCP

Lásd: Transmission Control Protocol

TCP/IP

Lásd: Transmission Control Protocol/Internet Protocol

TD

Lásd: Transmitted Data

TFTP

Lásd: Trivial FTP

TGT

Lásd: Ticket-Granting Ticket

TSC

Lásd: Time Stamp Counter

Ticket-Granting Ticket

Time Stamp Counter

A modern Pentium processzorokban megtalálható precíz belső számláló, amely a mag frekvenciájával érkező órajeleket számolja.

Transmission Control Protocol

(Például) Az IP protokoll felett ülő protokoll, amely garantálja, hogy a csomagok megbízható, sorbarendezt módon jutnak el a céljukba.

Transmission Control Protocol/Internet Protocol

Az IP protokoll és felette futó TCP protokoll kombinációjára utaló fogalom. Az internet legnagyobb része a TCP/IP protokollon keresztül működik.

Transmitted Data

Egy RS232C szabványú tű vagy vezeték, amin keresztül az adat átküldésre kerül.

Lásd még: Received Data.

Trivial FTP

U

UDP

Lásd: User Datagram Protocol

UFS1

Lásd: Unix File System Version 1

UFS2

Lásd: Unix File System Version 2

UID

Lásd: User ID

URL

Lásd: Uniform Resource Locator

USB

Lásd: Universal Serial Bus

Uniform Resource Locator

Az interneten található erőforrások, például dokumentumok helyének meghatározására és azonosítására alkalmas módszer.

Unix File System Version 1

Az eredeti UNIX állományrendszer, amelyet gyakran Berkeley Fast File Systemnek neveznek.

Unix File System Version 2

Az UFS1 FreeBSD 5-CURRENT változatában megjelent kiegészítése. Az UFS2 különböző kiegészítéseket, és 64 bites blokkmutatókat tartalmaz (így lehetővé teszi az 1 TB-os határ átlépését), valamint kibővített állománytárolást tesz lehetővé.

Universal Serial Bus

Különféle számítógépes perifériákat egy univerzális interfészen keresztül összekapcsoló hardveres szabvány.

User ID

A számítógép minden egyes felhasználója számára kiosztott egyedi azonosítószám, aminek segítségével a az erőforrások és engedélyek egyértelműen hozzájuk kapcsolhatóak.

User Datagram Protocol

TCP/IP hálózatokban adatkommunikációra használható egyszerű, nem megbízható, datagram alapú protokoll. Az UDP nem tartalmaz a TCP-hez hasonló hibaérzékelést és -javítást.

V

VPN

Lásd: Virtual Private Network

Virtual Private Network

Helyi, például egy vállalati hálózat, valamilyen publikus hírközlési eszközön, például interneten keresztüli távoli elérésére alkalmas módszer.

Végszó

Ez a könyv “A FreeBSD Dokumentációs Projekt” több száz résztvevőjének együttes munkájával jött létre. A forrása SGML-ben készül a DocBook DTD alapján, és ebből számos különböző formátumba kerül átalakításra a **Jade**, a nyílt forráskódú DSSSL engine segítségével. Egy további réteg hozzáadásával Norm Walsh DSSSL stíluslapjait használtuk fel a **Jade** vezérlésére. A dokumentum nyomtatott változata nem jöhetett volna létre Donald Knuth \TeX betűszedő rendszere, illetve Leslie Lamport \LaTeX , valamint Sebastian Rahtz **JadeTeX** makrócsomagja nélkül.