FreeBSD Handbook

The FreeBSD Documentation Project

FreeBSD Handbook

by The FreeBSD Documentation Project

Published February 1999

Welcome to FreeBSD! This handbook covers the installation and day to day use of *FreeBSD 7.4-RELEASE* and *FreeBSD 8.2-RELEASE*. This manual is a *work in progress* and is the work of many individuals. As such, some sections may become dated and require updating. If you are interested in helping out with this project, send email to the FreeBSD documentation project mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc). The latest version of this document is always available from the FreeBSD web site (http://www.FreeBSD.org/) (previous versions of this handbook can be obtained from http://docs.FreeBSD.org/doc/). It may also be downloaded in a variety of formats and compression options from the FreeBSD FTP server (ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/) or one of the numerous mirror sites. If you would prefer to have a hard copy of the handbook, you can purchase one at the FreeBSD Mall (http://www.freebsdmall.com/). You may also want to search the handbook (http://www.FreeBSD.org/search/index.html).

Table of Contents

List of Tables

Preface

Intended Audience

The FreeBSD newcomer will find that the first section of this book guides the user through the FreeBSD installation process and gently introduces the concepts and conventions that underpin UNIX®. Working through this section requires little more than the desire to explore, and the ability to take on board new concepts as they are introduced.

Once you have traveled this far, the second, far larger, section of the Handbook is a comprehensive reference to all manner of topics of interest to FreeBSD system administrators. Some of these chapters may recommend that you do some prior reading, and this is noted in the synopsis at the beginning of each chapter.

For a list of additional sources of information, please see Appendix B.

Changes from the Third Edition

The current online version of the Handbook represents the cumulative effort of many hundreds of contributors over the past 10 years. The following are some of the significant changes since the two volume third edition was published in 2004:

- Chapter 25, DTrace, has been added with information about the powerful DTrace performance analysis tool.
- Chapter 20, File Systems Support, has been added with information about non-native file systems in FreeBSD, such as ZFS from Sun™.
- Chapter 17, Security Event Auditing, has been added to cover the new auditing capabilities in FreeBSD and explain its use.
- Chapter 22, Virtualization, has been added with information about installing FreeBSD on virtualization software.

Changes from the Second Edition (2004)

The third edition was the culmination of over two years of work by the dedicated members of the FreeBSD Documentation Project. The printed edition grew to such a size that it was necessary to publish as two separate volumes. The following are the major changes in this new edition:

- Chapter 11, Configuration and Tuning, has been expanded with new information about the ACPI power and resource management, the `cron` system utility, and more kernel tuning options.
- Chapter 14, Security, has been expanded with new information about virtual private networks (VPNs), file system access control lists (ACLs), and security advisories.
- Chapter 16, Mandatory Access Control (MAC), is a new chapter with this edition. It explains what MAC is and how this mechanism can be used to secure a FreeBSD system.
- Chapter 18, Storage, has been expanded with new information about USB storage devices, file system snapshots, file system quotas, file and network backed filesystems, and encrypted disk partitions.

- Chapter 21, Vinum, is a new chapter with this edition. It describes how to use Vinum, a logical volume manager which provides device-independent logical disks, and software RAID-0, RAID-1 and RAID-5.
- A troubleshooting section has been added to Chapter 27, PPP and SLIP.
- Chapter 28, Electronic Mail, has been expanded with new information about using alternative transport agents, SMTP authentication, UUCP, **fetchmail**, **procmail**, and other advanced topics.
- Chapter 29, Network Servers, is all new with this edition. This chapter includes information about setting up the **Apache HTTP Server**, **ftpd**, and setting up a server for Microsoft® Windows® clients with **Samba**. Some sections from Chapter 31, Advanced Networking, were moved here to improve the presentation.
- Chapter 31, Advanced Networking, has been expanded with new information about using Bluetooth® devices with FreeBSD, setting up wireless networks, and Asynchronous Transfer Mode (ATM) networking.
- A glossary has been added to provide a central location for the definitions of technical terms used throughout the book.
- A number of aesthetic improvements have been made to the tables and figures throughout the book.

Changes from the First Edition (2001)

The second edition was the culmination of over two years of work by the dedicated members of the FreeBSD Documentation Project. The following were the major changes in this edition:

- A complete Index has been added.
- All ASCII figures have been replaced by graphical diagrams.
- A standard synopsis has been added to each chapter to give a quick summary of what information the chapter contains, and what the reader is expected to know.
- The content has been logically reorganized into three parts: “Getting Started”, “System Administration”, and “Appendices”.
- Chapter 2 (“Installing FreeBSD”) was completely rewritten with many screenshots to make it much easier for new users to grasp the text.
- Chapter 3 (“UNIX Basics”) has been expanded to contain additional information about processes, daemons, and signals.
- Chapter 4 (“Installing Applications”) has been expanded to contain additional information about binary package management.
- Chapter 5 (“The X Window System”) has been completely rewritten with an emphasis on using modern desktop technologies such as **KDE** and **GNOME** on XFree86™ 4.X.
- Chapter 12 (“The FreeBSD Booting Process”) has been expanded.
- Chapter 18 (“Storage”) has been written from what used to be two separate chapters on “Disks” and “Backups”. We feel that the topics are easier to comprehend when presented as a single chapter. A section on RAID (both hardware and software) has also been added.
- Chapter 26 (“Serial Communications”) has been completely reorganized and updated for FreeBSD 4.X/5.X.
- Chapter 27 (“PPP and SLIP”) has been substantially updated.
- Many new sections have been added to Chapter 31 (“Advanced Networking”).

- Chapter 28 (“Electronic Mail”) has been expanded to include more information about configuring **sendmail**.
- Chapter 10 (“Linux® Compatibility”) has been expanded to include information about installing **Oracle®** and **SAP® R/3®**.
- The following new topics are covered in this second edition:
  - Configuration and Tuning (Chapter 11).
  - Multimedia (Chapter 7)

Organization of This Book

This book is split into five logically distinct sections. The first section, *Getting Started*, covers the installation and basic usage of FreeBSD. It is expected that the reader will follow these chapters in sequence, possibly skipping chapters covering familiar topics. The second section, *Common Tasks*, covers some frequently used features of FreeBSD. This section, and all subsequent sections, can be read out of order. Each chapter begins with a succinct synopsis that describes what the chapter covers and what the reader is expected to already know. This is meant to allow the casual reader to skip around to find chapters of interest. The third section, *System Administration*, covers administration topics. The fourth section, *Network Communication*, covers networking and server topics. The fifth section contains appendices of reference information.

Chapter 1, Introduction

Introduces FreeBSD to a new user. It describes the history of the FreeBSD Project, its goals and development model.

Chapter 2, Installation

Walks a user through the entire installation process. Some advanced installation topics, such as installing through a serial console, are also covered.

Chapter 3, UNIX Basics

Covers the basic commands and functionality of the FreeBSD operating system. If you are familiar with Linux or another flavor of UNIX then you can probably skip this chapter.

Chapter 4, Installing Applications

Covers the installation of third-party software with both FreeBSD’s innovative “Ports Collection” and standard binary packages.

Chapter 5, The X Window System

Describes the X Window System in general and using X11 on FreeBSD in particular. Also describes common desktop environments such as **KDE** and **GNOME**.

Chapter 6, Desktop Applications

Lists some common desktop applications, such as web browsers and productivity suites, and describes how to install them on FreeBSD.

Chapter 7, Multimedia

Shows how to set up sound and video playback support for your system. Also describes some sample audio and video applications.

Chapter 8, Configuring the FreeBSD Kernel

Explains why you might need to configure a new kernel and provides detailed instructions for configuring, building, and installing a custom kernel.

Chapter 9, Printing

Describes managing printers on FreeBSD, including information about banner pages, printer accounting, and initial setup.

Chapter 10, Linux Binary Compatibility

Describes the Linux compatibility features of FreeBSD. Also provides detailed installation instructions for many popular Linux applications such as **Oracle** and **Mathematica®**.

Chapter 11, Configuration and Tuning

Describes the parameters available for system administrators to tune a FreeBSD system for optimum performance. Also describes the various configuration files used in FreeBSD and where to find them.

Chapter 12, Booting Process

Describes the FreeBSD boot process and explains how to control this process with configuration options.

Chapter 13, Users and Basic Account Management

Describes the creation and manipulation of user accounts. Also discusses resource limitations that can be set on users and other account management tasks.

Chapter 14, Security

Describes many different tools available to help keep your FreeBSD system secure, including Kerberos, IPsec and OpenSSH.

Chapter 15, Jails

Describes the jails framework, and the improvements of jails over the traditional chroot support of FreeBSD.

Chapter 16, Mandatory Access Control

Explains what Mandatory Access Control (MAC) is and how this mechanism can be used to secure a FreeBSD system.

Chapter 17, Security Event Auditing

Describes what FreeBSD Event Auditing is, how it can be installed, configured, and how audit trails can be inspected or monitored.

Chapter 18, Storage

Describes how to manage storage media and filesystems with FreeBSD. This includes physical disks, RAID arrays, optical and tape media, memory-backed disks, and network filesystems.

Chapter 19, GEOM

Describes what the GEOM framework in FreeBSD is and how to configure various supported RAID levels.

Chapter 20, File Systems Support

Examines support of non-native file systems in FreeBSD, like the Z File System from Sun.

Chapter 21, Vinum

Describes how to use Vinum, a logical volume manager which provides device-independent logical disks, and software RAID-0, RAID-1 and RAID-5.

Chapter 22, Virtualization

Describes what virtualization systems offer, and how they can be used with FreeBSD.

Chapter 23, Localization

Describes how to use FreeBSD in languages other than English. Covers both system and application level localization.

Chapter 24, Updating and Upgrading FreeBSD

Explains the differences between FreeBSD-STABLE, FreeBSD-CURRENT, and FreeBSD releases. Describes which users would benefit from tracking a development system and outlines that process. Covers the methods users may take to update their system to the latest security release.

Chapter 25, DTrace

Describes how to configure and use the DTrace tool from Sun in FreeBSD. Dynamic tracing can help locate performance issues, by performing real time system analysis.

Chapter 26, Serial Communications

Explains how to connect terminals and modems to your FreeBSD system for both dial in and dial out connections.

Chapter 27, PPP and SLIP

Describes how to use PPP, SLIP, or PPP over Ethernet to connect to remote systems with FreeBSD.

Chapter 28, Electronic Mail

Explains the different components of an email server and dives into simple configuration topics for the most popular mail server software: **sendmail**.

Chapter 29, Network Servers

Provides detailed instructions and example configuration files to set up your FreeBSD machine as a network filesystem server, domain name server, network information system server, or time synchronization server.

Chapter 30, Firewalls

Explains the philosophy behind software-based firewalls and provides detailed information about the configuration of the different firewalls available for FreeBSD.

Chapter 31, Advanced Networking

Describes many networking topics, including sharing an Internet connection with other computers on your LAN, advanced routing topics, wireless networking, Bluetooth, ATM, IPv6, and much more.

Appendix A, Obtaining FreeBSD

Lists different sources for obtaining FreeBSD media on CDROM or DVD as well as different sites on the Internet that allow you to download and install FreeBSD.

Appendix B, Bibliography

This book touches on many different subjects that may leave you hungry for a more detailed explanation. The bibliography lists many excellent books that are referenced in the text.

Appendix C, Resources on the Internet

Describes the many forums available for FreeBSD users to post questions and engage in technical conversations about FreeBSD.

Appendix D, PGP Keys

Lists the PGP fingerprints of several FreeBSD Developers.

Conventions used in this book

To provide a consistent and easy to read text, several conventions are followed throughout the book.

Typographic Conventions

Italic

An *italic* font is used for filenames, URLs, emphasized text, and the first usage of technical terms.

`Monospace`

A `monospaced` font is used for error messages, commands, environment variables, names of ports, hostnames, user names, group names, device names, variables, and code fragments.

Bold

A **bold** font is used for applications, commands, and keys.

User Input

Keys are shown in **bold** to stand out from other text. Key combinations that are meant to be typed simultaneously are shown with '+' between the keys, such as:

Ctrl+**Alt**+**Del**

Meaning the user should type the **Ctrl**, **Alt**, and **Del** keys at the same time.

Keys that are meant to be typed in sequence will be separated with commas, for example:

Ctrl+**X**, **Ctrl**+**S**

Would mean that the user is expected to type the **Ctrl** and **X** keys simultaneously and then to type the **Ctrl** and **S** keys simultaneously.

Examples

Examples starting with `E:\>` indicate a MS-DOS® command. Unless otherwise noted, these commands may be executed from a "Command Prompt" window in a modern Microsoft Windows environment.

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

Examples starting with # indicate a command that must be invoked as the superuser in FreeBSD. You can login as `root` to type the command, or login as your normal account and use su(1) to gain superuser privileges.

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

Examples starting with `%` indicate a command that should be invoked from a normal user account. Unless otherwise noted, C-shell syntax is used for setting environment variables and other shell commands.

```
% top
```

Acknowledgments

The book you are holding represents the efforts of many hundreds of people around the world. Whether they sent in fixes for typos, or submitted complete chapters, all the contributions have been useful.

Several companies have supported the development of this document by paying authors to work on it full-time, paying for publication, etc. In particular, BSDi (subsequently acquired by Wind River Systems (http://www.windriver.com)) paid members of the FreeBSD Documentation Project to work on improving this book full time leading up to the publication of the first printed edition in March 2000 (ISBN 1-57176-241-8). Wind River Systems then paid several additional authors to make a number of improvements to the print-output infrastructure and to add additional chapters to the text. This work culminated in the publication of the second printed edition in November 2001 (ISBN 1-57176-303-1). In 2003-2004, FreeBSD Mall, Inc (http://www.freebsdmall.com), paid several contributors to improve the Handbook in preparation for the third printed edition.

I. Getting Started

This part of the FreeBSD Handbook is for users and administrators who are new to FreeBSD. These chapters:

- Introduce you to FreeBSD.
- Guide you through the installation process.
- Teach you UNIX basics and fundamentals.
- Show you how to install the wealth of third party applications available for FreeBSD.
- Introduce you to X, the UNIX windowing system, and detail how to configure a desktop environment that makes you more productive.

We have tried to keep the number of forward references in the text to a minimum so that you can read this section of the Handbook from front to back with the minimum page flipping required.

Chapter 1
Introduction

1.1 Synopsis

Thank you for your interest in FreeBSD! The following chapter covers various aspects of the FreeBSD Project, such as its history, goals, development model, and so on.

After reading this chapter, you will know:

- How FreeBSD relates to other computer operating systems.
- The history of the FreeBSD Project.
- The goals of the FreeBSD Project.
- The basics of the FreeBSD open-source development model.
- And of course: where the name “FreeBSD” comes from.

1.2 Welcome to FreeBSD!

FreeBSD is a 4.4BSD-Lite based operating system for Intel (x86 and Itanium®), AMD64, Sun UltraSPARC® computers. Ports to other architectures are also underway. You can also read about the history of FreeBSD, or the current release. If you are interested in contributing something to the Project (code, hardware, funding), see the Contributing to FreeBSD (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributing/index.html) article.

1.2.1 What Can FreeBSD Do?

FreeBSD has many noteworthy features. Some of these are:

- *Preemptive multitasking* with dynamic priority adjustment to ensure smooth and fair sharing of the computer between applications and users, even under the heaviest of loads.
- *Multi-user facilities* which allow many people to use a FreeBSD system simultaneously for a variety of things. This means, for example, that system peripherals such as printers and tape drives are properly shared between all users on the system or the network and that individual resource limits can be placed on users or groups of users, protecting critical system resources from over-use.
- Strong *TCP/IP networking* with support for industry standards such as SCTP, DHCP, NFS, NIS, PPP, SLIP, IPsec, and IPv6. This means that your FreeBSD machine can interoperate easily with other systems as well as act as an enterprise server, providing vital functions such as NFS (remote file access) and email services or putting your organization on the Internet with WWW, FTP, routing and firewall (security) services.

- *Memory protection* ensures that applications (or users) cannot interfere with each other. One application crashing will not affect others in any way.
- FreeBSD is a *32-bit* operating system (*64-bit* on the Itanium, AMD64, and UltraSPARC) and was designed as such from the ground up.
- The industry standard *X Window System* (X11R7) provides a graphical user interface (GUI) for the cost of a common VGA card and monitor and comes with full sources.
- *Binary compatibility* with many programs built for Linux, SCO, SVR4, BSDI and NetBSD.
- Thousands of *ready-to-run* applications are available from the FreeBSD *ports* and *packages* collection. Why search the net when you can find it all right here?
- Thousands of additional and *easy-to-port* applications are available on the Internet. FreeBSD is source code compatible with most popular commercial UNIX systems and thus most applications require few, if any, changes to compile.
- Demand paged *virtual memory* and "merged VM/buffer cache" design efficiently satisfies applications with large appetites for memory while still maintaining interactive response to other users.
- *SMP* support for machines with multiple CPUs.
- A full complement of *C*, *C++*, and *Fortran* development tools. Many additional languages for advanced research and development are also available in the ports and packages collection.
- *Source code* for the entire system means you have the greatest degree of control over your environment. Why be locked into a proprietary solution at the mercy of your vendor when you can have a truly open system?
- Extensive *online documentation*.
- *And many more!*

FreeBSD is based on the 4.4BSD-Lite release from Computer Systems Research Group (CSRG) at the University of California at Berkeley, and carries on the distinguished tradition of BSD systems development. In addition to the fine work provided by CSRG, the FreeBSD Project has put in many thousands of hours in fine tuning the system for maximum performance and reliability in real-life load situations. As many of the commercial giants struggle to field PC operating systems with such features, performance and reliability, FreeBSD can offer them *now*!

The applications to which FreeBSD can be put are truly limited only by your own imagination. From software development to factory automation, inventory control to azimuth correction of remote satellite antennae; if it can be done with a commercial UNIX product then it is more than likely that you can do it with FreeBSD too! FreeBSD also benefits significantly from literally thousands of high quality applications developed by research centers and universities around the world, often available at little to no cost. Commercial applications are also available and appearing in greater numbers every day.

Because the source code for FreeBSD itself is generally available, the system can also be customized to an almost unheard of degree for special applications or projects, and in ways not generally possible with operating systems from most major commercial vendors. Here is just a sampling of some of the applications in which people are currently using FreeBSD:

- *Internet Services:* The robust TCP/IP networking built into FreeBSD makes it an ideal platform for a variety of Internet services such as:
  - FTP servers
  - World Wide Web servers (standard or secure [SSL])

  - IPv4 and IPv6 routing
  - Firewalls and NAT (“IP masquerading”) gateways
  - Electronic Mail servers
  - USENET News or Bulletin Board Systems
  - And more...

  With FreeBSD, you can easily start out small with an inexpensive 386 class PC and upgrade all the way up to a quad-processor Xeon with RAID storage as your enterprise grows.

- *Education:* Are you a student of computer science or a related engineering field? There is no better way of learning about operating systems, computer architecture and networking than the hands on, under the hood experience that FreeBSD can provide. A number of freely available CAD, mathematical and graphic design packages also make it highly useful to those whose primary interest in a computer is to get *other* work done!
- *Research:* With source code for the entire system available, FreeBSD is an excellent platform for research in operating systems as well as other branches of computer science. FreeBSD’s freely available nature also makes it possible for remote groups to collaborate on ideas or shared development without having to worry about special licensing agreements or limitations on what may be discussed in open forums.
- *Networking:* Need a new router? A name server (DNS)? A firewall to keep people out of your internal network? FreeBSD can easily turn that unused 386 or 486 PC sitting in the corner into an advanced router with sophisticated packet-filtering capabilities.
- *X Window workstation:* FreeBSD is a fine choice for an inexpensive X terminal solution, using the freely available X11 server. Unlike an X terminal, FreeBSD allows many applications to be run locally if desired, thus relieving the burden on a central server. FreeBSD can even boot “diskless”, making individual workstations even cheaper and easier to administer.
- *Software Development:* The basic FreeBSD system comes with a full complement of development tools including the renowned GNU C/C++ compiler and debugger.

FreeBSD is available in both source and binary form on CD-ROM, DVD, and via anonymous FTP. Please see Appendix A for more information about obtaining FreeBSD.

1.2.2 Who Uses FreeBSD?

FreeBSD is used as a platform for devices and products from many of the world’s largest IT companies, including:

- Apple (http://www.apple.com/)
- Cisco (http://www.cisco.com/)
- Juniper (http://www.juniper.net/)
- NetApp (http://www.netapp.com/)

FreeBSD is also used to power some of the biggest sites on the Internet, including:

- 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/)

and many more.

1.3 About the FreeBSD Project

The following section provides some background information on the project, including a brief history, project goals, and the development model of the project.

1.3.1 A Brief History of FreeBSD

The FreeBSD Project had its genesis in the early part of 1993, partially as an outgrowth of the "Unofficial 386BSD Patchkit" by the patchkit's last 3 coordinators: Nate Williams, Rod Grimes and myself.

Our original goal was to produce an intermediate snapshot of 386BSD in order to fix a number of problems with it that the patchkit mechanism just was not capable of solving. Some of you may remember the early working title for the project being "386BSD 0.5" or "386BSD Interim" in reference to that fact.

386BSD was Bill Jolitz's operating system, which had been up to that point suffering rather severely from almost a year's worth of neglect. As the patchkit swelled ever more uncomfortably with each passing day, we were in unanimous agreement that something had to be done and decided to assist Bill by providing this interim "cleanup" snapshot. Those plans came to a rude halt when Bill Jolitz suddenly decided to withdraw his sanction from the project without any clear indication of what would be done instead.

It did not take us long to decide that the goal remained worthwhile, even without Bill's support, and so we adopted the name "FreeBSD", coined by David Greenman. Our initial objectives were set after consulting with the system's current users and, once it became clear that the project was on the road to perhaps even becoming a reality, I contacted Walnut Creek CDROM with an eye toward improving FreeBSD's distribution channels for those many unfortunates without easy access to the Internet. Walnut Creek CDROM not only supported the idea of distributing FreeBSD on CD but also went so far as to provide the project with a machine to work on and a fast Internet connection. Without Walnut Creek CDROM's almost unprecedented degree of faith in what was, at the time, a completely unknown project, it is quite unlikely that FreeBSD would have gotten as far, as fast, as it has today.

The first CD-ROM (and general net-wide) distribution was FreeBSD 1.0, released in December of 1993. This was based on the 4.3BSD-Lite ("Net/2") tape from U.C. Berkeley, with many components also provided by 386BSD and the Free Software Foundation. It was a fairly reasonable success for a first offering, and we followed it with the highly successful FreeBSD 1.1 release in May of 1994.

Around this time, some rather unexpected storm clouds formed on the horizon as Novell and U.C. Berkeley settled their long-running lawsuit over the legal status of the Berkeley Net/2 tape. A condition of that settlement was U.C. Berkeley's concession that large parts of Net/2 were "encumbered" code and the property of Novell, who had in turn acquired it from AT&T some time previously. What Berkeley got in return was Novell's "blessing" that the 4.4BSD-Lite release, when it was finally released, would be declared unencumbered and all existing Net/2 users would be strongly encouraged to switch. This included FreeBSD, and the project was given until the end of July 1994 to stop shipping its own Net/2 based product. Under the terms of that agreement, the project was allowed one last release before the deadline, that release being FreeBSD 1.1.5.1.

FreeBSD then set about the arduous task of literally re-inventing itself from a completely new and rather incomplete set of 4.4BSD-Lite bits. The "Lite" releases were light in part because Berkeley's CSRG had removed large chunks of code required for actually constructing a bootable running system (due to various legal requirements) and the fact that the Intel port of 4.4 was highly incomplete. It took the project until November of 1994 to make this transition, at which point it released FreeBSD 2.0 to the net and on CD-ROM (in late December). Despite being still more than a little rough around the edges, the release was a significant success and was followed by the more robust and easier to install FreeBSD 2.0.5 release in June of 1995.

We released FreeBSD 2.1.5 in August of 1996, and it appeared to be popular enough among the ISP and commercial communities that another release along the 2.1-STABLE branch was merited. This was FreeBSD 2.1.7.1, released in February 1997 and capping the end of mainstream development on 2.1-STABLE. Now in maintenance mode, only security enhancements and other critical bug fixes will be done on this branch (RELENG_2_1_0).

FreeBSD 2.2 was branched from the development mainline ("-CURRENT") in November 1996 as the RELENG_2_2 branch, and the first full release (2.2.1) was released in April 1997. Further releases along the 2.2 branch were done in the summer and fall of '97, the last of which (2.2.8) appeared in November 1998. The first official 3.0 release appeared in October 1998 and spelled the beginning of the end for the 2.2 branch.

The tree branched again on Jan 20, 1999, leading to the 4.0-CURRENT and 3.X-STABLE branches. From 3.X-STABLE, 3.1 was released on February 15, 1999, 3.2 on May 15, 1999, 3.3 on September 16, 1999, 3.4 on December 20, 1999, and 3.5 on June 24, 2000, which was followed a few days later by a minor point release update to 3.5.1, to incorporate some last-minute security fixes to Kerberos. This will be the final release in the 3.X branch.

There was another branch on March 13, 2000, which saw the emergence of the 4.X-STABLE branch. There have been several releases from it so far: 4.0-RELEASE was introduced in March 2000, and the last 4.11-RELEASE came out in January 2005.

The long-awaited 5.0-RELEASE was announced on January 19, 2003. The culmination of nearly three years of work, this release started FreeBSD on the path of advanced multiprocessor and application thread support and introduced support for the UltraSPARC and `ia64` platforms. This release was followed by 5.1 in June of 2003. The last 5.X release from the -CURRENT branch was 5.2.1-RELEASE, introduced in February 2004.

The RELENG_5 branch, created in August 2004, was followed by 5.3-RELEASE, which marked the beginning of the 5-STABLE branch releases. The most recent 5.5-RELEASE release came out in May 2006. There will be no additional releases from the RELENG_5 branch.

The tree was branched again in July 2005, this time for RELENG_6. 6.0-RELEASE, the first release of the 6.X branch, was released in November 2005. The most recent 6.4-RELEASE came out in November 2008. There will be no additional releases from the RELENG_6 branch. This branch is the last branch to support the Alpha architecture.

The RELENG_7 branch was created in October 2007. The first release of this branch was 7.0-RELEASE, which came out in February 2008. The most recent 7.4-RELEASE came out in Jan 2011. There will be additional releases from the RELENG_7 branch.

The tree was branched again in August 2009, this time for RELENG_8. 8.0-RELEASE, the first release of the 8.X branch, was released in November 2009. The most recent 8.2-RELEASE came out in Jan 2011. There will be additional releases from the RELENG_8 branch.

For now, long-term development projects continue to take place in the 9.X-CURRENT (trunk) branch, and SNAPshot releases of 9.X on CD-ROM (and, of course, on the net) are continually made available from the snapshot server (ftp://current.FreeBSD.org/pub/FreeBSD/snapshots/) as work progresses.

1.3.2 FreeBSD Project Goals

The goals of the FreeBSD Project are to provide software that may be used for any purpose and without strings attached. Many of us have a significant investment in the code (and project) and would certainly not mind a little financial compensation now and then, but we are definitely not prepared to insist on it. We believe that our first and foremost "mission" is to provide code to any and all comers, and for whatever purpose, so that the code gets the widest possible use and provides the widest possible benefit. This is, I believe, one of the most fundamental goals of Free Software and one that we enthusiastically support.

That code in our source tree which falls under the GNU General Public License (GPL) or Library General Public License (LGPL) comes with slightly more strings attached, though at least on the side of enforced access rather than the usual opposite. Due to the additional complexities that can evolve in the commercial use of GPL software we do, however, prefer software submitted under the more relaxed BSD copyright when it is a reasonable option to do so.

1.3.3 The FreeBSD Development Model

The development of FreeBSD is a very open and flexible process, being literally built from the contributions of hundreds of people around the world, as can be seen from our list of contributors (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/article.html). FreeBSD's development infrastructure allow these hundreds of developers to collaborate over the Internet. We are constantly on the lookout for new developers and ideas, and those interested in becoming more closely involved with the project need simply contact us at the FreeBSD technical discussions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers). The FreeBSD announcements mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce) is also available to those wishing to make other FreeBSD users aware of major areas of work.

Useful things to know about the FreeBSD Project and its development process, whether working independently or in close cooperation:

The SVN and CVS repositories

> For several years, the central source tree for FreeBSD was maintained by CVS (http://ximbiot.com/cvs/wiki/) (Concurrent Versions System), a freely available source code control tool that comes bundled with FreeBSD. In June 2008, the Project switched to using SVN (http://subversion.tigris.org) (Subversion). The switch was deemed necessary, as the technical limitations imposed by **CVS** were becoming obvious due to the rapid expansion of the source tree and the amount of history already stored. While the main repository now uses **SVN**, client side tools like **CVSup** and **csup** that depend on the older **CVS** infrastructure, continue to work normally — changes in the **SVN** repository are backported to **CVS** for this purpose. Currently, only the central source tree is controlled by **SVN**. The documentation, World Wide Web, and Ports repositories are still using **CVS**. The primary repository (http://www.FreeBSD.org/cgi/cvsweb.cgi) resides on a machine in Santa Clara CA, USA from where it is replicated to numerous mirror machines throughout the world. The **SVN** tree, which

contains the -CURRENT and -STABLE trees, can all be easily replicated to your own machine as well. Please refer to the Synchronizing your source tree section for more information on doing this.

The committers list

The *committers* are the people who have *write* access to the CVS tree, and are authorized to make modifications to the FreeBSD source (the term "committer" comes from the cvs(1) `commit` command, which is used to bring new changes into the CVS repository). The best way of making submissions for review by the committers list is to use the send-pr(1) command. If something appears to be jammed in the system, then you may also reach them by sending mail to the FreeBSD committer's mailing list.

The FreeBSD core team

The *FreeBSD core team* would be equivalent to the board of directors if the FreeBSD Project were a company. The primary task of the core team is to make sure the project, as a whole, is in good shape and is heading in the right directions. Inviting dedicated and responsible developers to join our group of committers is one of the functions of the core team, as is the recruitment of new core team members as others move on. The current core team was elected from a pool of committer candidates in July 2008. Elections are held every 2 years.

Some core team members also have specific areas of responsibility, meaning that they are committed to ensuring that some large portion of the system works as advertised. For a complete list of FreeBSD developers and their areas of responsibility, please see the Contributors List (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/article.html)

> **Note:** Most members of the core team are volunteers when it comes to FreeBSD development and do not benefit from the project financially, so "commitment" should also not be misconstrued as meaning "guaranteed support." The "board of directors" analogy above is not very accurate, and it may be more suitable to say that these are the people who gave up their lives in favor of FreeBSD against their better judgement!

Outside contributors

Last, but definitely not least, the largest group of developers are the users themselves who provide feedback and bug fixes to us on an almost constant basis. The primary way of keeping in touch with FreeBSD's more non-centralized development is to subscribe to the FreeBSD technical discussions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers) where such things are discussed. See Appendix C for more information about the various FreeBSD mailing lists.

The FreeBSD Contributors List (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/article.html) is a long and growing one, so why not join it by contributing something back to FreeBSD today?

Providing code is not the only way of contributing to the project; for a more complete list of things that need doing, please refer to the FreeBSD Project web site (http://www.FreeBSD.org/index.html).

In summary, our development model is organized as a loose set of concentric circles. The centralized model is designed for the convenience of the *users* of FreeBSD, who are provided with an easy way of tracking one central code base, not to keep potential contributors out! Our desire is to present a stable operating system with a large set of coherent application programs that the users can easily install and use — this model works very well in accomplishing that.

All we ask of those who would join us as FreeBSD developers is some of the same dedication its current people have to its continued success!

1.3.4 The Current FreeBSD Release

FreeBSD is a freely available, full source 4.4BSD-Lite based release for Intel i386™, i486™, Pentium®, Pentium Pro, Celeron®, Pentium II, Pentium III, Pentium 4 (or compatible), Xeon™, and Sun UltraSPARC based computer systems. It is based primarily on software from U.C. Berkeley's CSRG group, with some enhancements from NetBSD, OpenBSD, 386BSD, and the Free Software Foundation.

Since our release of FreeBSD 2.0 in late 1994, the performance, feature set, and stability of FreeBSD has improved dramatically. The largest change is a revamped virtual memory system with a merged VM/file buffer cache that not only increases performance, but also reduces FreeBSD's memory footprint, making a 5 MB configuration a more acceptable minimum. Other enhancements include full NIS client and server support, transaction TCP support, dial-on-demand PPP, integrated DHCP support, an improved SCSI subsystem, ISDN support, support for ATM, FDDI, Fast and Gigabit Ethernet (1000 Mbit) adapters, improved support for the latest Adaptec controllers, and many thousands of bug fixes.

In addition to the base distributions, FreeBSD offers a ported software collection with thousands of commonly sought-after programs. At the time of this printing, there were over 20,000 ports! The list of ports ranges from http (WWW) servers, to games, languages, editors, and almost everything in between. The entire Ports Collection requires approximately 417 MB of storage, all ports being expressed as "deltas" to their original sources. This makes it much easier for us to update ports, and greatly reduces the disk space demands made by the older 1.0 Ports Collection. To compile a port, you simply change to the directory of the program you wish to install, type `make install`, and let the system do the rest. The full original distribution for each port you build is retrieved dynamically off the CD-ROM or a local FTP site, so you need only enough disk space to build the ports you want. Almost every port is also provided as a pre-compiled "package", which can be installed with a simple command (`pkg_add`) by those who do not wish to compile their own ports from source. More information on packages and ports can be found in Chapter 4.

A number of additional documents which you may find very helpful in the process of installing and using FreeBSD may now also be found in the `/usr/share/doc` directory on any recent FreeBSD machine. You may view the locally installed manuals with any HTML capable browser using the following URLs:

The FreeBSD Handbook

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

The FreeBSD FAQ

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

You can also view the master (and most frequently updated) copies at http://www.FreeBSD.org/.

Chapter 2
Installing FreeBSD

2.1 Synopsis

FreeBSD is provided with a text-based, easy to use installation program called **sysinstall**. This is the default installation program for FreeBSD, although vendors are free to provide their own installation suite if they wish. This chapter describes how to use **sysinstall** to install FreeBSD.

After reading this chapter, you will know:

- How to create the FreeBSD installation disks.
- How FreeBSD refers to, and subdivides, your hard disks.
- How to start **sysinstall**.
- The questions **sysinstall** will ask you, what they mean, and how to answer them.

Before reading this chapter, you should:

- Read the supported hardware list that shipped with the version of FreeBSD you are installing, and verify that your hardware is supported.

> **Note:** In general, these installation instructions are written for i386 ("PC compatible") architecture computers. Where applicable, instructions specific to other platforms will be listed. Although this guide is kept as up to date as possible, you may find minor differences between the installer and what is shown here. It is suggested that you use this chapter as a general guide rather than a literal installation manual.

2.2 Hardware Requirements

2.2.1 Minimal Configuration

The minimal configuration to install FreeBSD varies with the FreeBSD version and the hardware architecture.

A summary of this information is given in the following sections. Depending on the method you choose to install FreeBSD, you may also need a floppy drive, a supported CDROM drive, and in some case a network adapter. This will be covered by the Section 2.3.7.

2.2.1.1 FreeBSD/i386 and FreeBSD/pc98

Both FreeBSD/i386 and FreeBSD/pc98 require a 486 or better processor and at least 24 MB of RAM. You will need at least 150 MB of free hard drive space for the most minimal installation.

> **Note:** In case of old configurations, most of time, getting more RAM and more hard drive space is more important than getting a faster processor.

2.2.1.2 FreeBSD/amd64

There are two classes of processors capable of running FreeBSD/amd64. The first are AMD64 processors, including the AMD Athlon™64, AMD Athlon64-FX, AMD Opteron™ or better processors.

The second class of processors that can use FreeBSD/amd64 includes those using the Intel® EM64T architecture. Examples of these processors include the Intel Core™ 2 Duo, Quad, and Extreme processor families and the Intel Xeon 3000, 5000, and 7000 sequences of processors.

If you have a machine based on an nVidia nForce3 Pro-150, you *must* use the BIOS setup to disable the IO APIC. If you do not have an option to do this, you will likely have to disable ACPI instead. There are bugs in the Pro-150 chipset that we have not found a workaround for yet.

2.2.1.3 FreeBSD/sparc64

To install FreeBSD/sparc64, you will need a supported platform (see Section 2.2.2).

You will need a dedicated disk for FreeBSD/sparc64. It is not possible to share a disk with another operating system at this time.

2.2.2 Supported Hardware

A list of supported hardware is provided with each FreeBSD release in the FreeBSD Hardware Notes. This document can usually be found in a file named `HARDWARE.TXT`, in the top-level directory of a CDROM or FTP distribution or in **sysinstall**'s documentation menu. It lists, for a given architecture, what hardware devices are known to be supported by each release of FreeBSD. Copies of the supported hardware list for various releases and architectures can also be found on the Release Information (http://www.FreeBSD.org/releases/index.html) page of the FreeBSD Web site.

2.3 Pre-installation Tasks

2.3.1 Inventory Your Computer

Before installing FreeBSD you should attempt to inventory the components in your computer. The FreeBSD installation routines will show you the components (hard disks, network cards, CDROM drives, and so forth) with their model number and manufacturer. FreeBSD will also attempt to determine the correct configuration for these

devices, which includes information about IRQ and IO port usage. Due to the vagaries of PC hardware this process is not always completely successful, and you may need to correct FreeBSD's determination of your configuration.

If you already have another operating system installed, such as Windows or Linux, it is a good idea to use the facilities provided by those operating systems to see how your hardware is already configured. If you are not sure what settings an expansion card is using, you may find it printed on the card itself. Popular IRQ numbers are 3, 5, and 7, and IO port addresses are normally written as hexadecimal numbers, such as 0x330.

We recommend you print or write down this information before installing FreeBSD. It may help to use a table, like this:

Table 2-1. Sample Device Inventory

Device Name	IRQ	IO port(s)	Notes
First hard disk	N/A	N/A	40 GB, made by Seagate, first IDE master
CDROM	N/A	N/A	First IDE slave
Second hard disk	N/A	N/A	20 GB, made by IBM, second IDE master
First IDE controller	14	0x1f0	
Network card	N/A	N/A	Intel 10/100
Modem	N/A	N/A	3Com® 56K faxmodem, on COM1
…			

Once the inventory of the components in your computer is done, you have to check if they match the hardware requirements of the FreeBSD release you want to install.

2.3.2 Backup Your Data

If the computer you will be installing FreeBSD on contains valuable data, then ensure you have it backed up, and that you have tested the backups before installing FreeBSD. The FreeBSD installation routine will prompt you before writing any data to your disk, but once that process has started it cannot be undone.

2.3.3 Decide Where to Install FreeBSD

If you want FreeBSD to use your entire hard disk, then there is nothing more to concern yourself with at this point — you can skip this section.

However, if you need FreeBSD to co-exist with other operating systems then you need to have a rough understanding of how data is laid out on the disk, and how this affects you.

2.3.3.1 Disk Layouts for FreeBSD/i386

A PC disk can be divided into discrete chunks. These chunks are called *partitions*. Since FreeBSD internally also has partitions, the naming can become confusing very quickly, therefore these disk chunks are referred to as disk slices or simply slices in FreeBSD itself. For example, the FreeBSD utility `fdisk` which operates on the PC disk partitions, refers to slices instead of partitions. By design, the PC only supports four partitions per disk. These partitions are called *primary partitions*. To work around this limitation and allow more than four partitions, a new partition type was created, the *extended partition*. A disk may contain only one extended partition. Special partitions, called *logical partitions*, can be created inside this extended partition.

Each partition has a *partition ID*, which is a number used to identify the type of data on the partition. FreeBSD partitions have the partition ID of `165`.

In general, each operating system that you use will identify partitions in a particular way. For example, DOS, and its descendants, like Windows, assign each primary and logical partition a *drive letter*, starting with `C:`.

FreeBSD must be installed into a primary partition. FreeBSD can keep all its data, including any files that you create, on this one partition. However, if you have multiple disks, then you can create a FreeBSD partition on all, or some, of them. When you install FreeBSD, you must have one partition available. This might be a blank partition that you have prepared, or it might be an existing partition that contains data that you no longer care about.

If you are already using all the partitions on all your disks, then you will have to free one of them for FreeBSD using the tools provided by the other operating systems you use (e.g., `fdisk` on DOS or Windows).

If you have a spare partition then you can use that. However, you may need to shrink one or more of your existing partitions first.

A minimal installation of FreeBSD takes as little as 100 MB of disk space. However, that is a *very* minimal install, leaving almost no space for your own files. A more realistic minimum is 250 MB without a graphical environment, and 350 MB or more if you want a graphical user interface. If you intend to install a lot of third-party software as well, then you will need more space.

You can use a commercial tool such as **PartitionMagic®**, or a free tool such as **GParted**, to resize your partitions and make space for FreeBSD. Both **PartitionMagic** and **GParted** are known to work on NTFS. **GParted** is available on a number of Live CD Linux distributions, such as SystemRescueCD (http://www.sysresccd.org/).

Problems have been reported resizing Microsoft Vista partitions. Having a Vista installation CDROM handy when attempting such an operation is recommended. As with all such disk maintenance tasks, a current set of backups is also strongly advised.

> **Warning:** Incorrect use of these tools can delete the data on your disk. Be sure that you have recent, working backups before using them.

Example 2-1. Using an Existing Partition Unchanged

Suppose that you have a computer with a single 4 GB disk that already has a version of Windows installed, and you have split the disk into two drive letters, `C:` and `D:`, each of which is 2 GB in size. You have 1 GB of data on `C:`, and 0.5 GB of data on `D:`.

This means that your disk has two partitions on it, one per drive letter. You can copy all your existing data from `D:` to `C:`, which will free up the second partition, ready for FreeBSD.

Example 2-2. Shrinking an Existing Partition

Suppose that you have a computer with a single 4 GB disk that already has a version of Windows installed. When you installed Windows you created one large partition, giving you a `C:` drive that is 4 GB in size. You are currently using 1.5 GB of space, and want FreeBSD to have 2 GB of space.

In order to install FreeBSD you will need to either:

1. Backup your Windows data, and then reinstall Windows, asking for a 2 GB partition at install time.
2. Use one of the tools such as **PartitionMagic**, described above, to shrink your Windows partition.

2.3.4 Collect Your Network Configuration Details

If you intend to connect to a network as part of your FreeBSD installation (for example, if you will be installing from an FTP site or an NFS server), then you need to know your network configuration. You will be prompted for this information during the installation so that FreeBSD can connect to the network to complete the install.

2.3.4.1 Connecting to an Ethernet Network or Cable/DSL Modem

If you connect to an Ethernet network, or you have an Internet connection using an Ethernet adapter via cable or DSL, then you will need the following information:

1. IP address
2. IP address of the default gateway
3. Hostname
4. DNS server IP addresses
5. Subnet Mask

If you do not know this information, then ask your system administrator or service provider. They may say that this information is assigned automatically, using *DHCP*. If so, make a note of this.

2.3.4.2 Connecting Using a Modem

If you dial up to an ISP using a regular modem then you can still install FreeBSD over the Internet, it will just take a very long time.

You will need to know:

1. The phone number to dial for your ISP
2. The COM: port your modem is connected to
3. The username and password for your ISP account

2.3.5 Check for FreeBSD Errata

Although the FreeBSD project strives to ensure that each release of FreeBSD is as stable as possible, bugs do occasionally creep into the process. On very rare occasions those bugs affect the installation process. As these problems are discovered and fixed, they are noted in the FreeBSD Errata (http://www.FreeBSD.org/releases/8.2R/errata.html), which is found on the FreeBSD web site. You should check the errata before installing to make sure that there are no late-breaking problems which you should be aware of.

Information about all the releases, including the errata for each release, can be found on the release information (http://www.FreeBSD.org/releases/index.html) section of the FreeBSD web site (http://www.FreeBSD.org/index.html).

2.3.6 Obtain the FreeBSD Installation Files

The FreeBSD installation process can install FreeBSD from files located in any of the following places:

Local Media

- A CDROM or DVD
- A USB Memory Stick
- A DOS partition on the same computer
- A SCSI or QIC tape
- Floppy disks

Network

- An FTP site, going through a firewall, or using an HTTP proxy, as necessary
- An NFS server
- A dedicated parallel or serial connection

If you have purchased FreeBSD on CD or DVD then you already have everything you need, and should proceed to the next section (Section 2.3.7).

If you have not obtained the FreeBSD installation files you should skip ahead to Section 2.13 which explains how to prepare to install FreeBSD from any of the above. After reading that section, you should come back here, and read on to Section 2.3.7.

2.3.7 Prepare the Boot Media

The FreeBSD installation process is started by booting your computer into the FreeBSD installer—it is not a program you run within another operating system. Your computer normally boots using the operating system installed on your hard disk, but it can also be configured to use a “bootable” floppy disk. Most modern computers can also boot from a CDROM in the CDROM drive or from a USB disk.

> **Tip:** If you have FreeBSD on CDROM or DVD (either one you purchased or you prepared yourself), and your computer allows you to boot from the CDROM or DVD (typically a BIOS option called “Boot Order” or similar), then you can skip this section. The FreeBSD CDROM and DVD images are bootable and can be used to install FreeBSD without any other special preparation.

To create a bootable memory stick, follow these steps:

1. Acquire the Memory Stick Image

   The memory stick image can be downloaded from the `ISO-IMAGES/` directory from `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/`*`arch`*`/ISO-IMAGES/`*`version`*`/FreeBSD-8.2-RELEASE-`*`arch`*`-mems`

Replace *arch* and *version* with the architecture and the version number which you want to install, respectively. For example, the memory stick images for FreeBSD/i386 8.2-RELEASE are available from ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.2/FreeBSD-8.2-RELEASE-i386-memstick.img.

The memory stick image has a `.img` extension. The `ISO-IMAGES/` directory contains a number of different images, and the one you will need to use will depend on the version of FreeBSD you are installing, and in some cases, the hardware you are installing to.

> **Important:** Before proceeding, *back up* the data you currently have on your USB stick, as this procedure will *erase* it.

2. Prepare the Memory Stick

> **Warning:** The example below lists `/dev/da0` as the target device from which you will be booting. Be very careful that you have the correct device as the output target, or you may destroy your existing data.

Set the `kern.geom.debugflags` sysctl to be able to write a master boot record to the target device.

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

3. Write the Image File to the Memory Stick

The `.img` file is *not* a regular file you copy to the memory stick. It is an image of the complete contents of the disk. This means that you *cannot* simply copy files from one disk to another. Instead, you must use dd(1) to write the image directly to the disk:

```
# dd if=FreeBSD-8.2-RELEASE-i386-memstick.img of=/dev/da0 bs=64k
```

To create boot floppy images, follow these steps:

1. Acquire the Boot Floppy Images

> **Important:** Please note, as of FreeBSD 8.0, floppy disk images are no longer available. Please see above for instructions on how to install FreeBSD using a USB memory stick or just use a CDROM or a DVD.

The boot disks are available on your installation media in the `floppies/` directory, and can also be downloaded from the floppies directory, `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/`*arch*`/`*version*`-RELEASE/floppies/`. Replace *arch* and *version* with the architecture and the version number which you want to install, respectively. For example, the boot floppy images for FreeBSD/i386 7.4-RELEASE are available from ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/7.4-RELEASE/floppies/.

The floppy images have a `.flp` extension. The `floppies/` directory contains a number of different images, and the ones you will need to use depends on the version of FreeBSD you are installing, and in some cases, the hardware you are installing to. In most cases you will need four floppies, `boot.flp`, `kern1.flp`, `kern2.flp`, and `kern3.flp`. Check `README.TXT` in the same directory for the most up to date information about these floppy images.

> **Important:** Your FTP program must use *binary mode* to download these disk images. Some web browsers have been known to use *text* (or *ASCII*) mode, which will be apparent if you cannot boot from the disks.

2. Prepare the Floppy Disks

   You must prepare one floppy disk per image file you had to download. It is imperative that these disks are free from defects. The easiest way to test this is to format the disks for yourself. Do not trust pre-formatted floppies. The format utility in Windows will not tell about the presence of bad blocks, it simply marks them as "bad" and ignores them. It is advised that you use brand new floppies if choosing this installation route.

   > **Important:** If you try to install FreeBSD and the installation program crashes, freezes, or otherwise misbehaves, one of the first things to suspect is the floppies. Try writing the floppy image files to new disks and try again.

3. Write the Image Files to the Floppy Disks

   The `.flp` files are *not* regular files you copy to the disk. They are images of the complete contents of the disk. This means that you *cannot* simply copy files from one disk to another. Instead, you must use specific tools to write the images directly to the disk.

   If you are creating the floppies on a computer running MS-DOS/Windows, then we provide a tool to do this called `fdimage`.

   If you are using the floppies from the CDROM, and your CDROM is the `E:` drive, then you would run this:

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

   Repeat this command for each `.flp` file, replacing the floppy disk each time, being sure to label the disks with the name of the file that you copied to them. Adjust the command line as necessary, depending on where you have placed the `.flp` files. If you do not have the CDROM, then `fdimage` can be downloaded from the `tools` directory (ftp://ftp.FreeBSD.org/pub/FreeBSD/tools/) on the FreeBSD FTP site.

   If you are writing the floppies on a UNIX system (such as another FreeBSD system) you can use the dd(1) command to write the image files directly to disk. On FreeBSD, you would run:

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

   On FreeBSD, `/dev/fd0` refers to the first floppy disk (the `A:` drive). `/dev/fd1` would be the `B:` drive, and so on. Other UNIX variants might have different names for the floppy disk devices, and you will need to check the documentation for the system as necessary.

You are now ready to start installing FreeBSD.

2.4 Starting the Installation

> **Important:** By default, the installation will not make any changes to your disk(s) until you see the following message:
>
> ```
> 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!
```

The install can be exited at any time prior to the final warning without changing the contents of the hard drive. If you are concerned that you have configured something incorrectly you can just turn the computer off before this point, and no damage will be done.

2.4.1 Booting

2.4.1.1 Booting for the i386™

1. Start with your computer turned off.
2. Turn on the computer. As it starts it should display an option to enter the system set up menu, or BIOS, commonly reached by keys like **F2**, **F10**, **Del**, or **Alt**+**S**. Use whichever keystroke is indicated on screen. In some cases your computer may display a graphic while it starts. Typically, pressing **Esc** will dismiss the graphic and allow you to see the necessary messages.
3. Find the setting that controls which devices the system boots from. This is usually labeled as the "Boot Order" and commonly shown as a list of devices, such as `Floppy`, `CDROM`, `First Hard Disk`, and so on.

   If you are booting from the CDROM then make sure that the CDROM is selected. If you are booting from a USB disk or a floppy disk then make sure that is selected instead. In case of doubt, you should consult the manual that came with your computer, and/or its motherboard.

   Make the change, then save and exit. The computer should now restart.
4. If you prepared a "bootable" USB stick, as described in Section 2.3.7, then plug in your USB stick before turning on the computer.

   If you are booting from CDROM, then you will need to turn on the computer, and insert the CDROM at the first opportunity.

   > **Note:** For FreeBSD 7.3 and previous versions, installation boot floppies are available and can be prepared as described in Section 2.3.7. One of them will be the first boot disc: `boot.flp`. Put this disc in your floppy drive and boot the computer.

   If your computer starts up as normal and loads your existing operating system, then either:

   1. The disks were not inserted early enough in the boot process. Leave them in, and try restarting your computer.
   2. The BIOS changes earlier did not work correctly. You should redo that step until you get the right option.
   3. Your particular BIOS does not support booting from the desired media.
5. FreeBSD will start to boot. If you are booting from CDROM you will see a display similar to this (version information omitted):

   ```
   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
Consoles: internal video/keyboard
BIOS CD is cd0
BIOS drive C: is disk0
BIOS drive D: is disk1
BIOS 636kB/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]
\
```

If you are booting from floppy disc, you will see a display similar to this (version information omitted):

```
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...
```

Follow these instructions by removing the `boot.flp` disc, insert the `kern1.flp` disc, and press **Enter**. Boot from first floppy; when prompted, insert the other disks as required.

6. Whether you booted from CDROM, USB stick or floppy, the boot process will then get to the FreeBSD boot loader menu:

Figure 2-1. FreeBSD Boot Loader Menu

Either wait ten seconds, or press **Enter**.

2.4.1.2 Booting for SPARC64®

Most SPARC64® systems are set up to boot automatically from disk. To install FreeBSD, you need to boot over the network or from a CDROM, which requires you to break into the PROM (OpenFirmware).

To do this, reboot the system, and wait until the boot message appears. It depends on the model, but should look about like:

```
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.
```

If your system proceeds to boot from disk at this point, you need to press **L1**+**A** or **Stop**+**A** on the keyboard, or send a `BREAK` over the serial console (using for example ~# in tip(1) or cu(1)) to get to the PROM prompt. It looks like this:

```
ok          ❶
ok {0}      ❷
```

❶ This is the prompt used on systems with just one CPU.

❷ This is the prompt used on SMP systems, the digit indicates the number of the active CPU.

At this point, place the CDROM into your drive, and from the PROM prompt, type `boot cdrom`.

2.4.2 Reviewing the Device Probe Results

The last few hundred lines that have been displayed on screen are stored and can be reviewed.

To review the buffer, press **Scroll Lock**. This turns on scrolling in the display. You can then use the arrow keys, or **PageUp** and **PageDown** to view the results. Press **Scroll Lock** again to stop scrolling.

Do this now, to review the text that scrolled off the screen when the kernel was carrying out the device probes. You will see text similar to Figure 2-2, although the precise text will differ depending on the devices that you have in your computer.

Figure 2-2. Typical Device Probe Results

```
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

md1: 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
atkbdc0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq1 on atkbdc0
kbd0 at atkbd0
```

```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
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
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: 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 ata1-slave PIO4
Mounting root from ufs:/dev/md0c
/stand/sysinstall running as init on vty0
```

Check the probe results carefully to make sure that FreeBSD found all the devices you expected. If a device was not found, then it will not be listed. A custom kernel allows you to add in support for devices which are not in the `GENERIC` kernel, such as sound cards.

For FreeBSD 6.2 and later, after the procedure of device probing, you will see Figure 2-3. Use the arrow key to choose a country, region, or group. Then press **Enter**, it will set your country easily. It is also easy to exit the **sysinstall** program and start over again.

Figure 2-3. Selecting Country Menu

If you selected United States as country, the standard American keyboard map will be used, if a different country is chosen the following menu will be displayed. Use the arrow keys to choose the correct keyboard map and press **Enter**.

Figure 2-4. Selecting Keyboard Menu

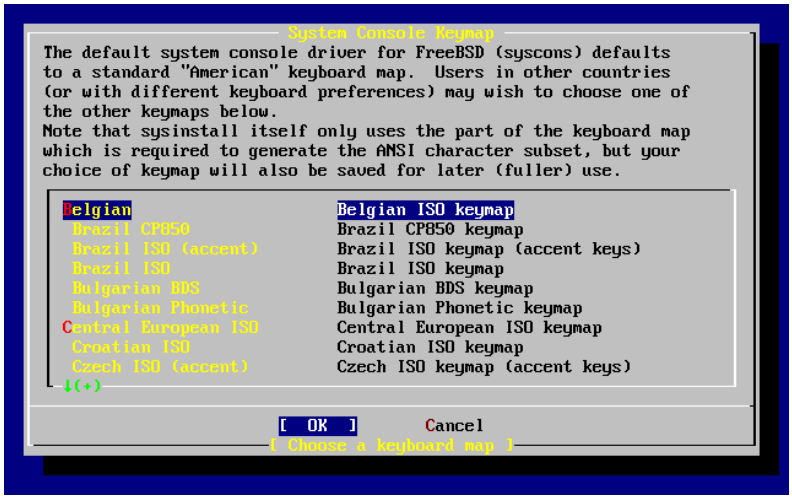
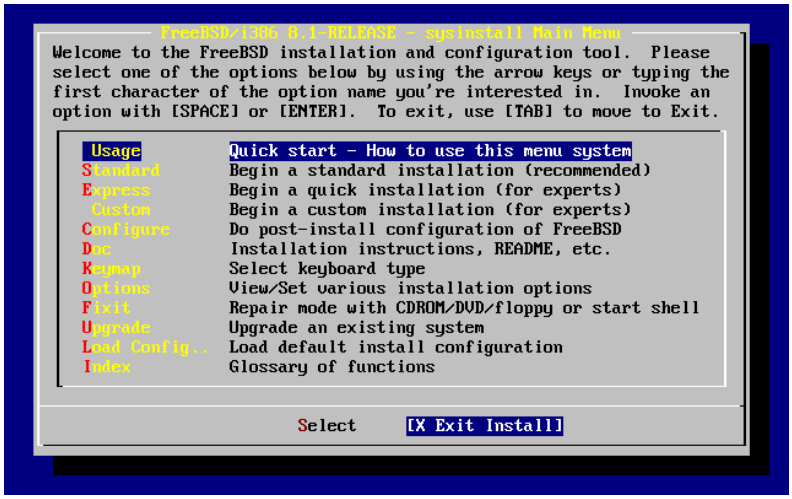


Figure 2-5. Select Sysinstall Exit



Use the arrow keys to select Exit Install from the Main Install Screen menu. The following message will display:

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

                        [ Yes ]    No
```

The install program will start again if the [Yes] is selected and the CDROM is left in the drive during the reboot. If you are booting from floppies it will be necessary to remove the `boot.flp` floppy before rebooting.

2.5 Introducing Sysinstall

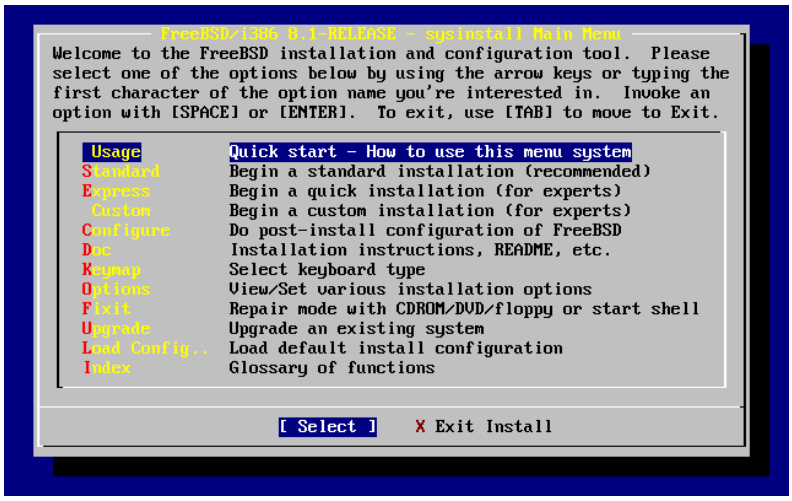
The **sysinstall** utility is the installation application provided by the FreeBSD Project. It is console based and is divided into a number of menus and screens that you can use to configure and control the installation process.

The **sysinstall** menu system is controlled by the arrow keys, **Enter**, **Tab**, **Space**, and other keys. A detailed description of these keys and what they do is contained in **sysinstall**'s usage information.

To review this information, ensure that the Usage entry is highlighted and that the [Select] button is selected, as shown in Figure 2-6, then press **Enter**.

The instructions for using the menu system will be displayed. After reviewing them, press **Enter** to return to the Main Menu.

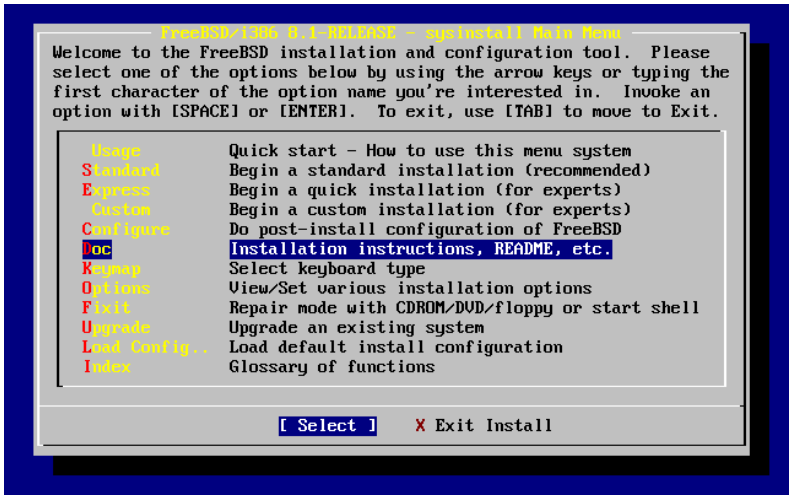
Figure 2-6. Selecting Usage from Sysinstall Main Menu



2.5.1 Selecting the Documentation Menu

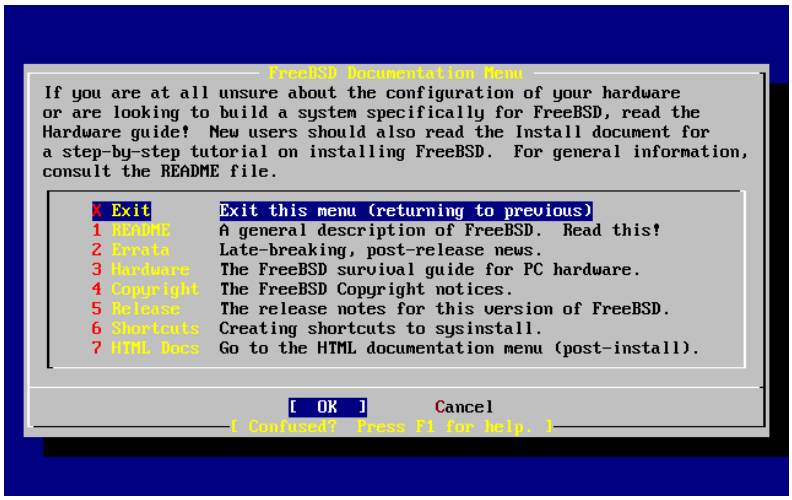
From the Main Menu, select Doc with the arrow keys and press **Enter**.

Figure 2-7. Selecting Documentation Menu



This will display the Documentation Menu.

Figure 2-8. Sysinstall Documentation Menu



It is important to read the documents provided.

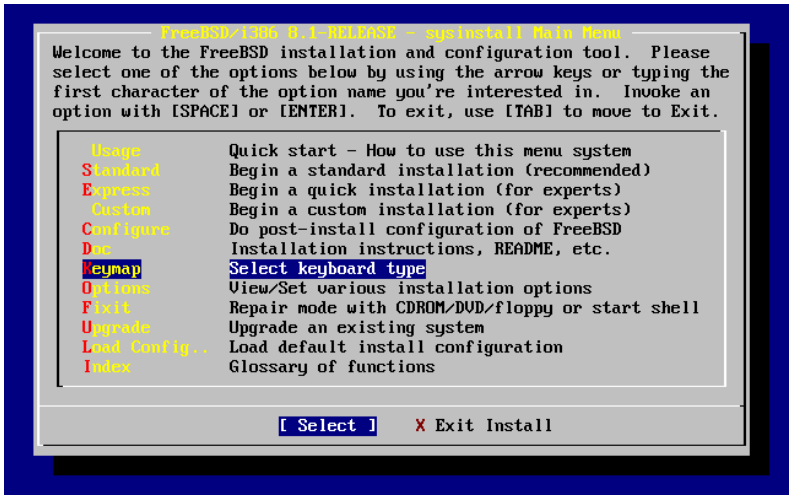
To view a document, select it with the arrow keys and press **Enter**. When finished reading a document, pressing **Enter** will return to the Documentation Menu.

To return to the Main Installation Menu, select Exit with the arrow keys and press **Enter**.

2.5.2 Selecting the Keymap Menu

To change the keyboard mapping, use the arrow keys to select Keymap from the menu and press **Enter**. This is only required if you are using a non-standard or non-US keyboard.

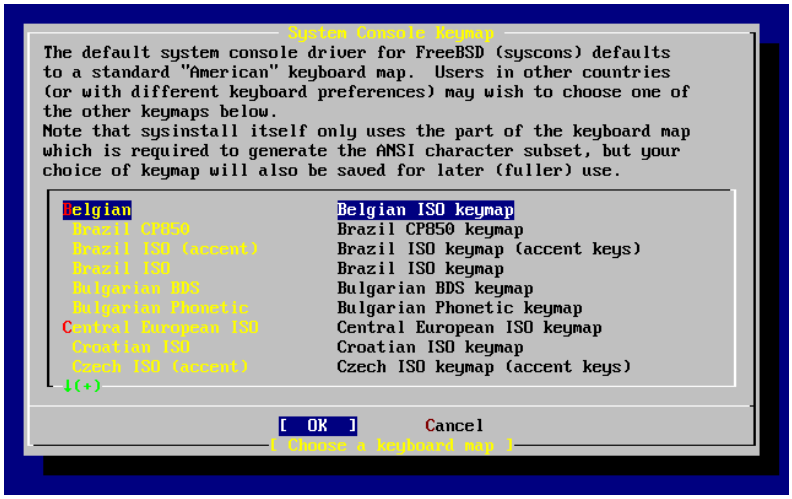
Figure 2-9. Sysinstall Main Menu



A different keyboard mapping may be chosen by selecting the menu item using up/down arrow keys and pressing **Space**. Pressing **Space** again will unselect the item. When finished, choose the [OK] using the arrow keys and press **Enter**.

Only a partial list is shown in this screen representation. Selecting [Cancel] by pressing **Tab** will use the default keymap and return to the Main Install Menu.

Figure 2-10. Sysinstall Keymap Menu



2.5.3 Installation Options Screen

Select Options and press **Enter**.

Figure 2-11. Sysinstall Main Menu

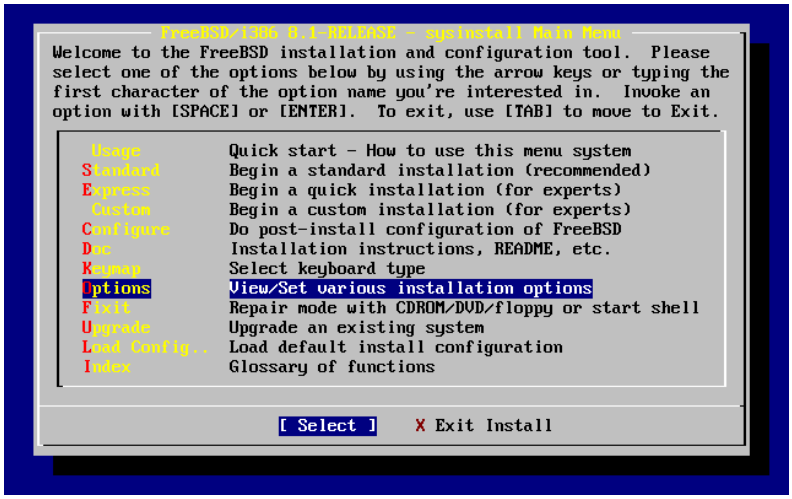
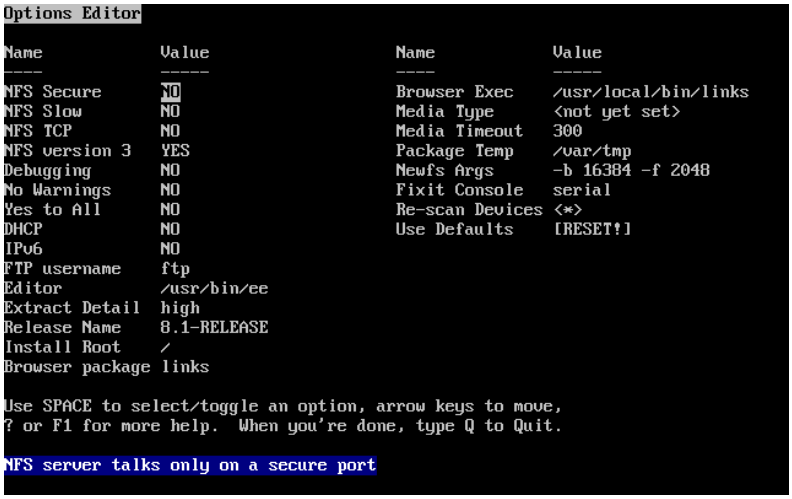


Figure 2-12. Sysinstall Options



The default values are usually fine for most users and do not need to be changed. The release name will vary according to the version being installed.

The description of the selected item will appear at the bottom of the screen highlighted in blue. Notice that one of the options is Use Defaults to reset all values to startup defaults.

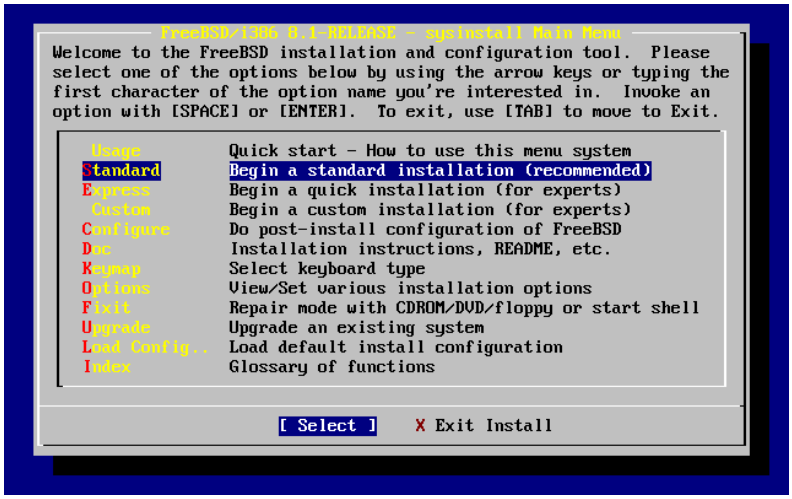
Press **F1** to read the help screen about the various options.

Pressing **Q** will return to the Main Install menu.

2.5.4 Begin a Standard Installation

The Standard installation is the option recommended for those new to UNIX or FreeBSD. Use the arrow keys to select Standard and then press **Enter** to start the installation.

Figure 2-13. Begin Standard Installation



2.6 Allocating Disk Space

Your first task is to allocate disk space for FreeBSD, and label that space so that **sysinstall** can prepare it. In order to do this you need to know how FreeBSD expects to find information on the disk.

2.6.1 BIOS Drive Numbering

Before you install and configure FreeBSD on your system, there is an important subject that you should be aware of, especially if you have multiple hard drives.

In a PC running a BIOS-dependent operating system such as MS-DOS or Microsoft Windows, the BIOS is able to abstract the normal disk drive order, and the operating system goes along with the change. This allows the user to boot from a disk drive other than the so-called “primary master”. This is especially convenient for some users who have found that the simplest and cheapest way to keep a system backup is to buy an identical second hard drive, and perform routine copies of the first drive to the second drive using **Ghost®** or **XCOPY** . Then, if the first drive fails, or is attacked by a virus, or is scribbled upon by an operating system defect, he can easily recover by instructing the BIOS to logically swap the drives. It is like switching the cables on the drives, but without having to open the case.

More expensive systems with SCSI controllers often include BIOS extensions which allow the SCSI drives to be re-ordered in a similar fashion for up to seven drives.

A user who is accustomed to taking advantage of these features may become surprised when the results with FreeBSD are not as expected. FreeBSD does not use the BIOS, and does not know the “logical BIOS drive mapping”. This can lead to very perplexing situations, especially when drives are physically identical in geometry, and have also been made as data clones of one another.

When using FreeBSD, always restore the BIOS to natural drive numbering before installing FreeBSD, and then leave it that way. If you need to switch drives around, then do so, but do it the hard way, and open the case and move the jumpers and cables.

An Illustration from the Files of Bill and Fred's Exceptional Adventures:

Bill breaks-down an older Wintel box to make another FreeBSD box for Fred. Bill installs a single SCSI drive as SCSI unit zero and installs FreeBSD on it.

Fred begins using the system, but after several days notices that the older SCSI drive is reporting numerous soft errors and reports this fact to Bill.

After several more days, Bill decides it is time to address the situation, so he grabs an identical SCSI drive from the disk drive "archive" in the back room. An initial surface scan indicates that this drive is functioning well, so Bill installs this drive as SCSI unit four and makes an image copy from drive zero to drive four. Now that the new drive is installed and functioning nicely, Bill decides that it is a good idea to start using it, so he uses features in the SCSI BIOS to re-order the disk drives so that the system boots from SCSI unit four. FreeBSD boots and runs just fine.

Fred continues his work for several days, and soon Bill and Fred decide that it is time for a new adventure — time to upgrade to a newer version of FreeBSD. Bill removes SCSI unit zero because it was a bit flaky and replaces it with another identical disk drive from the "archive". Bill then installs the new version of FreeBSD onto the new SCSI unit zero using Fred's magic Internet FTP floppies. The installation goes well.

Fred uses the new version of FreeBSD for a few days, and certifies that it is good enough for use in the engineering department. It is time to copy all of his work from the old version. So Fred mounts SCSI unit four (the latest copy of the older FreeBSD version). Fred is dismayed to find that none of his precious work is present on SCSI unit four.

Where did the data go?

When Bill made an image copy of the original SCSI unit zero onto SCSI unit four, unit four became the "new clone". When Bill re-ordered the SCSI BIOS so that he could boot from SCSI unit four, he was only fooling himself. FreeBSD was still running on SCSI unit zero. Making this kind of BIOS change will cause some or all of the Boot and Loader code to be fetched from the selected BIOS drive, but when the FreeBSD kernel drivers take-over, the BIOS drive numbering will be ignored, and FreeBSD will transition back to normal drive numbering. In the illustration at hand, the system continued to operate on the original SCSI unit zero, and all of Fred's data was there, not on SCSI unit four. The fact that the system appeared to be running on SCSI unit four was simply an artifact of human expectations.

We are delighted to mention that no data bytes were killed or harmed in any way by our discovery of this phenomenon. The older SCSI unit zero was retrieved from the bone pile, and all of Fred's work was returned to him, (and now Bill knows that he can count as high as zero).

Although SCSI drives were used in this illustration, the concepts apply equally to IDE drives.

2.6.2 Creating Slices Using FDisk

> **Note:** No changes you make at this point will be written to the disk. If you think you have made a mistake and want to start again you can use the menus to exit **sysinstall** and try again or press **U** to use the Undo option. If you get confused and can not see how to exit you can always turn your computer off.

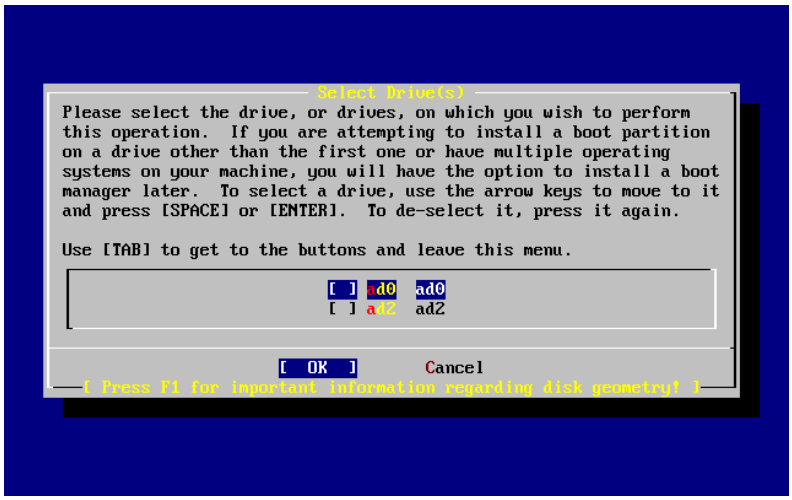
After choosing to begin a standard installation in **sysinstall** you will be shown this message:

```
                              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 ]
```

Press **Enter** as instructed. You will then be shown a list of all the hard drives that the kernel found when it carried out the device probes. Figure 2-14 shows an example from a system with two IDE disks. They have been called `ad0` and `ad2`.

Figure 2-14. Select Drive for FDisk



You might be wondering why `ad1` is not listed here. Why has it been missed?

Consider what would happen if you had two IDE hard disks, one as the master on the first IDE controller, and one as the master on the second IDE controller. If FreeBSD numbered these as it found them, as `ad0` and `ad1` then everything would work.

But if you then added a third disk, as the slave device on the first IDE controller, it would now be `ad1`, and the previous `ad1` would become `ad2`. Because device names (such as `ad1s1a`) are used to find filesystems, you may suddenly discover that some of your filesystems no longer appear correctly, and you would need to change your FreeBSD configuration.

To work around this, the kernel can be configured to name IDE disks based on where they are, and not the order in which they were found. With this scheme the master disk on the second IDE controller will *always* be `ad2`, even if there are no `ad0` or `ad1` devices.

This configuration is the default for the FreeBSD kernel, which is why this display shows `ad0` and `ad2`. The machine on which this screenshot was taken had IDE disks on both master channels of the IDE controllers, and no disks on the slave channels.

You should select the disk on which you want to install FreeBSD, and then press [OK]. **FDisk** will start, with a display similar to that shown in Figure 2-15.

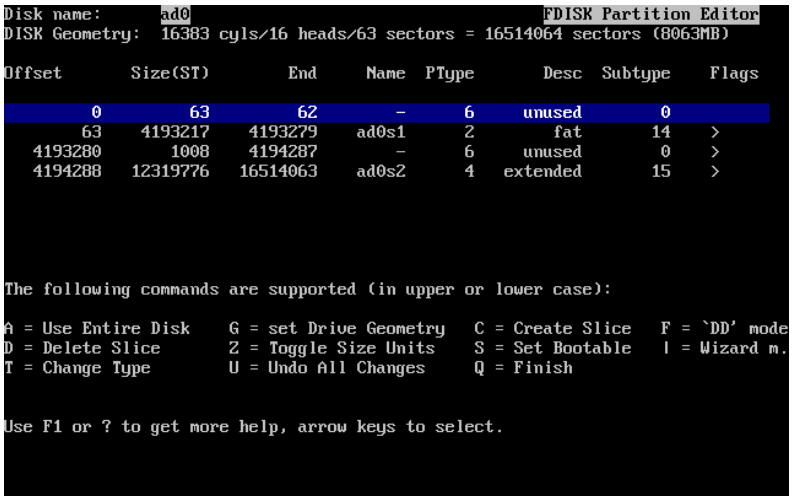
The **FDisk** display is broken into three sections.

The first section, covering the first two lines of the display, shows details about the currently selected disk, including its FreeBSD name, the disk geometry, and the total size of the disk.

The second section shows the slices that are currently on the disk, where they start and end, how large they are, the name FreeBSD gives them, and their description and sub-type. This example shows two small unused slices, which are artifacts of disk layout schemes on the PC. It also shows one large FAT slice, which almost certainly appears as `C:` in MS-DOS / Windows, and an extended slice, which may contain other drive letters for MS-DOS / Windows.

The third section shows the commands that are available in **FDisk**.

Figure 2-15. Typical Fdisk Partitions before Editing



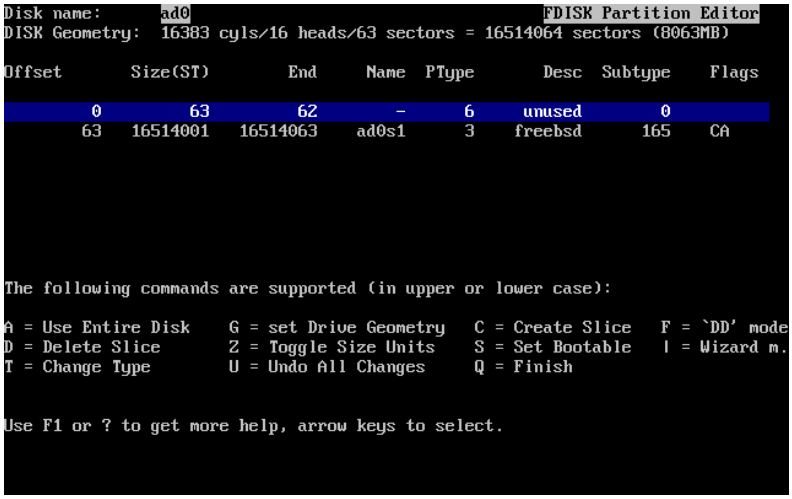
What you do now will depend on how you want to slice up your disk.

If you want to use FreeBSD for the entire disk (which will delete all the other data on this disk when you confirm that you want **sysinstall** to continue later in the installation process) then you can press **A**, which corresponds to the Use Entire Disk option. The existing slices will be removed, and replaced with a small area flagged as `unused` (again, an artifact of PC disk layout), and then one large slice for FreeBSD. If you do this, then you should select the newly created FreeBSD slice using the arrow keys, and press **S** to mark the slice as being bootable. The screen will then look very similar to Figure 2-16. Note the `A` in the `Flags` column, which indicates that this slice is *active*, and will be booted from.

If you will be deleting an existing slice to make space for FreeBSD then you should select the slice using the arrow keys, and then press **D**. You can then press **C**, and be prompted for size of slice you want to create. Enter the appropriate figure and press **Enter**. The default value in this box represents the largest possible slice you can make, which could be the largest contiguous block of unallocated space or the size of the entire hard disk.

If you have already made space for FreeBSD (perhaps by using a tool such as **PartitionMagic**) then you can press **C** to create a new slice. Again, you will be prompted for the size of slice you would like to create.

Figure 2-16. Fdisk Partition Using Entire Disk



When finished, press **Q**. Your changes will be saved in **sysinstall**, but will not yet be written to disk.

2.6.3 Install a Boot Manager

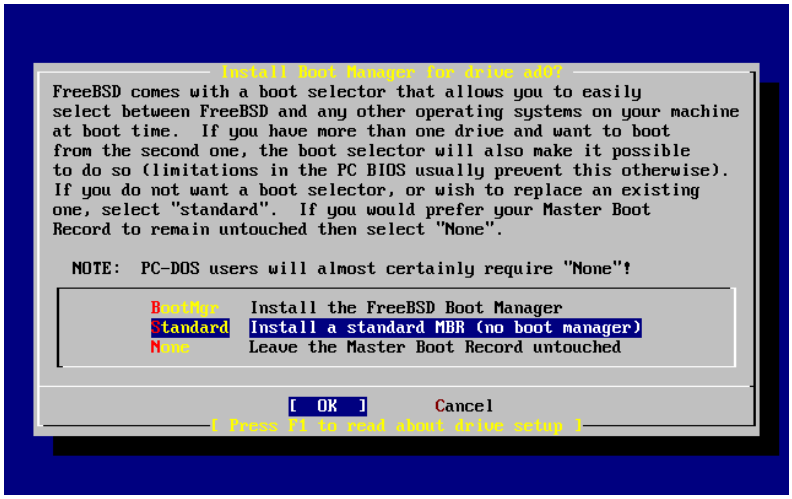
You now have the option to install a boot manager. In general, you should choose to install the FreeBSD boot manager if:

- You have more than one drive, and have installed FreeBSD onto a drive other than the first one.
- You have installed FreeBSD alongside another operating system on the same disk, and you want to choose whether to start FreeBSD or the other operating system when you start the computer.

If FreeBSD is going to be the only operating system on this machine, installed on the first hard disk, then the Standard boot manager will suffice. Choose None if you are using a third-party boot manager capable of booting FreeBSD.

Make your choice and press **Enter**.

Figure 2-17. Sysinstall Boot Manager Menu



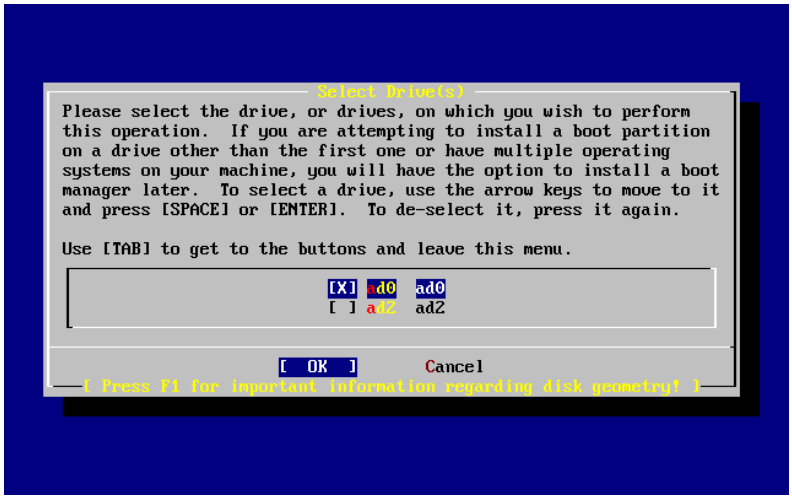
The help screen, reached by pressing **F1**, discusses the problems that can be encountered when trying to share the hard disk between operating systems.

2.6.4 Creating Slices on Another Drive

If there is more than one drive, it will return to the Select Drives screen after the boot manager selection. If you wish to install FreeBSD on to more than one disk, then you can select another disk here and repeat the slice process using **FDisk**.

> **Important:** If you are installing FreeBSD on a drive other than your first, then the FreeBSD boot manager needs to be installed on both drives.

Figure 2-18. Exit Select Drive



The **Tab** key toggles between the last drive selected, [OK], and [Cancel].

Press the **Tab** once to toggle to the [OK], then press **Enter** to continue with the installation.

2.6.5 Creating Partitions Using Disklabel

You must now create some partitions inside each slice that you have just created. Remember that each partition is lettered, from a through to h, and that partitions b, c, and d have conventional meanings that you should adhere to.

Certain applications can benefit from particular partition schemes, especially if you are laying out partitions across more than one disk. However, for this, your first FreeBSD installation, you do not need to give too much thought to how you partition the disk. It is more important that you install FreeBSD and start learning how to use it. You can always re-install FreeBSD to change your partition scheme when you are more familiar with the operating system.

This scheme features four partitions—one for swap space, and three for filesystems.

Table 2-2. Partition Layout for First Disk

Partition	Filesystem	Size	Description
a	/	1 GB	This is the root filesystem. Every other filesystem will be mounted somewhere under this one. 1 GB is a reasonable size for this filesystem. You will not be storing too much data on it, as a regular FreeBSD install will put about 128 MB of data here. The remaining space is for temporary data, and also leaves expansion space if future versions of FreeBSD need more space in /.

Partition	Filesystem	Size	Description
b	N/A	2-3 x RAM	The system's swap space is kept on the b partition. Choosing the right amount of swap space can be a bit of an art. A good rule of thumb is that your swap space should be two or three times as much as the available physical memory (RAM). You should also have at least 64 MB of swap, so if you have less than 32 MB of RAM in your computer then set the swap amount to 64 MB. If you have more than one disk then you can put swap space on each disk. FreeBSD will then use each disk for swap, which effectively speeds up the act of swapping. In this case, calculate the total amount of swap you need (e.g., 128 MB), and then divide this by the number of disks you have (e.g., two disks) to give the amount of swap you should put on each disk, in this example, 64 MB of swap per disk.
e	/var	512 MB to 4096 MB	The /var directory contains files that are constantly varying; log files, and other administrative files. Many of these files are read-from or written-to extensively during FreeBSD's day-to-day running. Putting these files on another filesystem allows FreeBSD to optimize the access of these files without affecting other files in other directories that do not have the same access pattern.
f	/usr	Rest of disk (at least 8 GB)	All your other files will typically be stored in /usr and its subdirectories.

Warning: The values above are given as example and should be used by experienced users only. Users are encouraged to use the automatic partition layout called `Auto Defaults` by the FreeBSD partition editor.

If you will be installing FreeBSD on to more than one disk then you must also create partitions in the other slices that you configured. The easiest way to do this is to create two partitions on each disk, one for the swap space, and one for a filesystem.

Table 2-3. Partition Layout for Subsequent Disks

Partition	Filesystem	Size	Description
b	N/A	See description	As already discussed, you can split swap space across each disk. Even though the a partition is free, convention dictates that swap space stays on the b partition.

Partition	Filesystem	Size	Description
e	/disk*n*	Rest of disk	The rest of the disk is taken up with one big partition. This could easily be put on the a partition, instead of the e partition. However, convention says that the a partition on a slice is reserved for the filesystem that will be the root (/) filesystem. You do not have to follow this convention, but **sysinstall** does, so following it yourself makes the installation slightly cleaner. You can choose to mount this filesystem anywhere; this example suggests that you mount them as directories /disk*n*, where *n* is a number that changes for each disk. But you can use another scheme if you prefer.

Having chosen your partition layout you can now create it using **sysinstall**. You will see this message:

```
                              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 (1GB 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 ]
```

Press **Enter** to start the FreeBSD partition editor, called **Disklabel**.

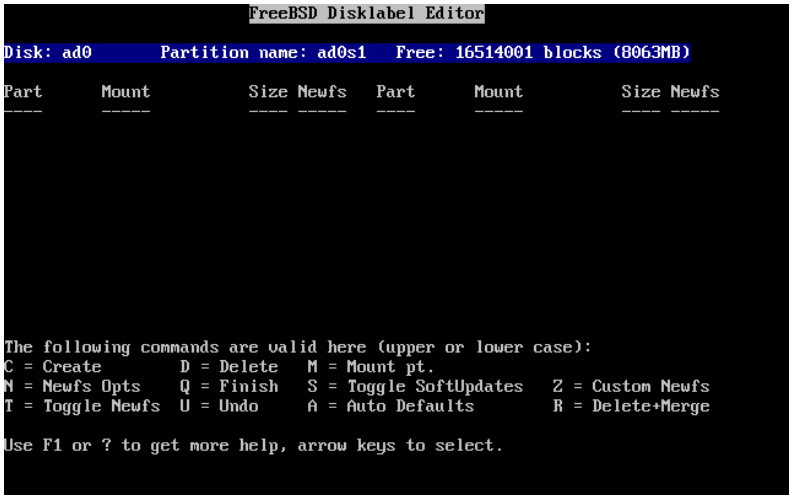
Figure 2-19 shows the display when you first start **Disklabel**. The display is divided in to three sections.

The first few lines show the name of the disk you are currently working on, and the slice that contains the partitions you are creating (at this point **Disklabel** calls this the `Partition name` rather than slice name). This display also shows the amount of free space within the slice; that is, space that was set aside in the slice, but that has not yet been assigned to a partition.

The middle of the display shows the partitions that have been created, the name of the filesystem that each partition contains, their size, and some options pertaining to the creation of the filesystem.

The bottom third of the screen shows the keystrokes that are valid in **Disklabel**.

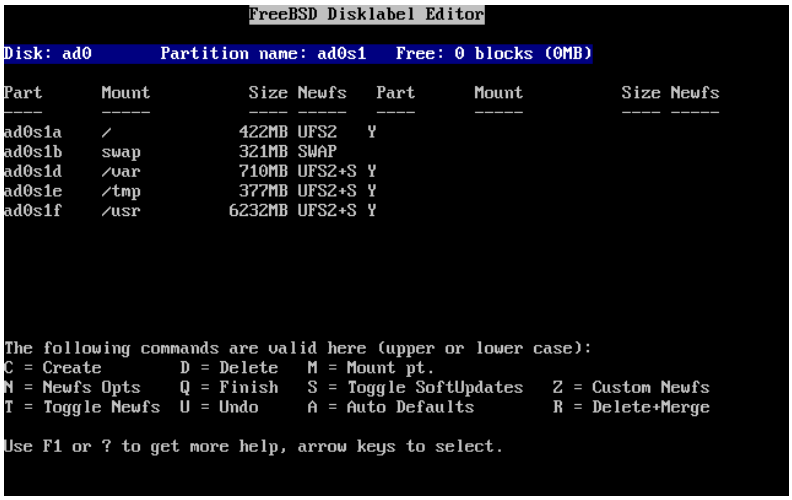
Figure 2-19. Sysinstall Disklabel Editor



Disklabel can automatically create partitions for you and assign them default sizes. The default sizes are calculated with the help of an internal partition sizing algorithm based on the disk size. Try this now, by Pressing **A**. You will see a display similar to that shown in Figure 2-20. Depending on the size of the disk you are using, the defaults may or may not be appropriate. This does not matter, as you do not have to accept the defaults.

> **Note:** The default partitioning assigns the `/tmp` directory its own partition instead of being part of the `/` partition. This helps avoid filling the `/` partition with temporary files.

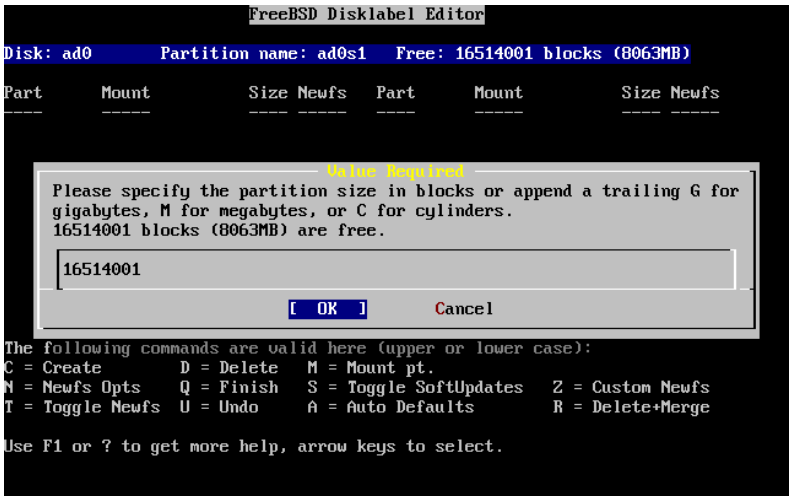
Figure 2-20. Sysinstall Disklabel Editor with Auto Defaults



If you choose to not use the default partitions and wish to replace them with your own, use the arrow keys to select the first partition, and press **D** to delete it. Repeat this to delete all the suggested partitions.

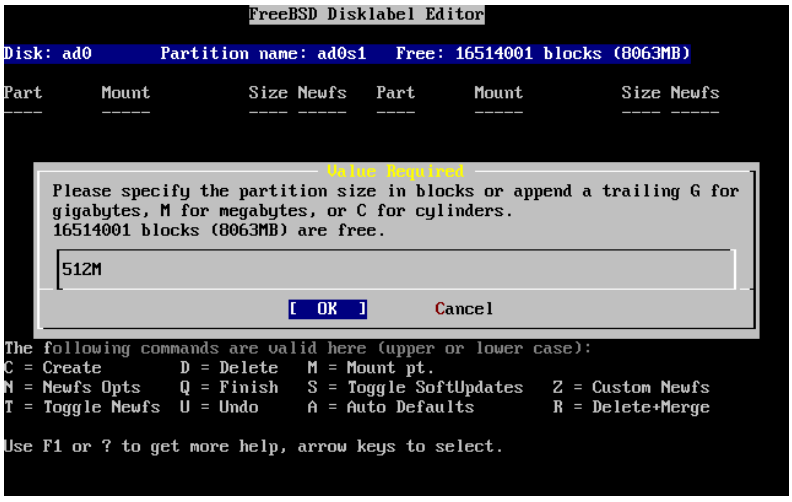
To create the first partition (a, mounted as / — root), make sure the proper disk slice at the top of the screen is selected and press **C**. A dialog box will appear prompting you for the size of the new partition (as shown in Figure 2-21). You can enter the size as the number of disk blocks you want to use, or as a number followed by either M for megabytes, G for gigabytes, or C for cylinders.

Figure 2-21. Free Space for Root Partition



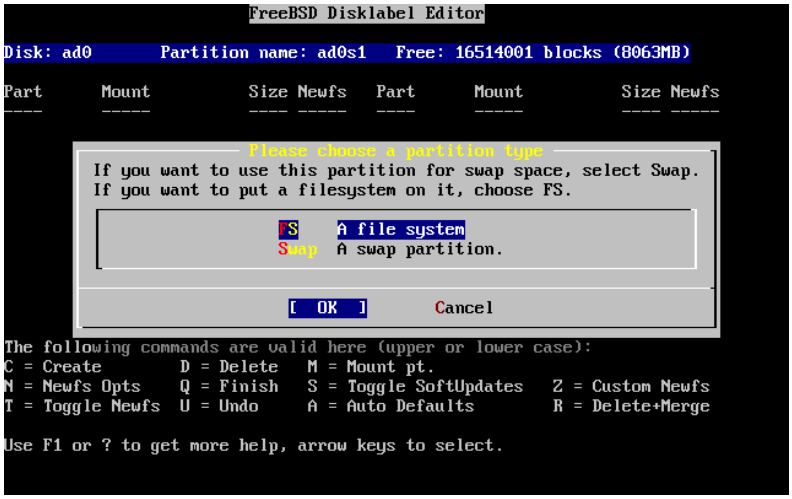
The default size shown will create a partition that takes up the rest of the slice. If you are using the partition sizes described in the earlier example, then delete the existing figure using **Backspace**, and then type in **512M**, as shown in Figure 2-22. Then press [OK].

Figure 2-22. Edit Root Partition Size



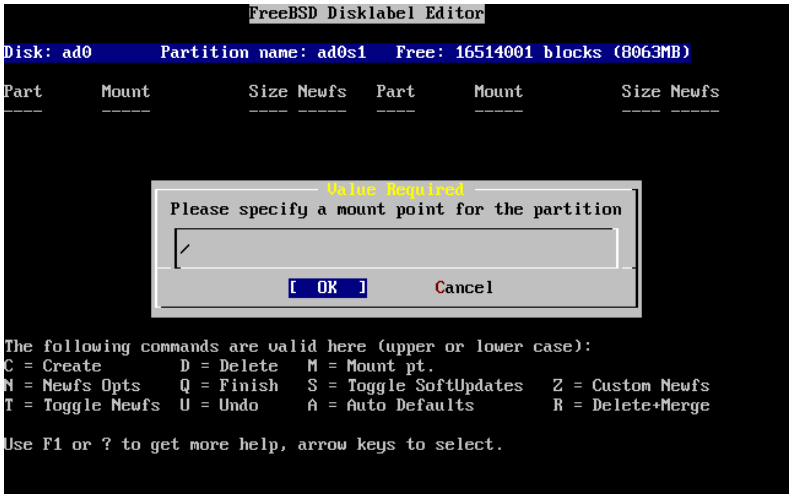
Having chosen the partition's size you will then be asked whether this partition will contain a filesystem or swap space. The dialog box is shown in Figure 2-23. This first partition will contain a filesystem, so check that FS is selected and press **Enter**.

Figure 2-23. Choose the Root Partition Type



Finally, because you are creating a filesystem, you must tell **Disklabel** where the filesystem is to be mounted. The dialog box is shown in Figure 2-24. The root filesystem's mount point is `/`, so type `/`, and then press **Enter**.

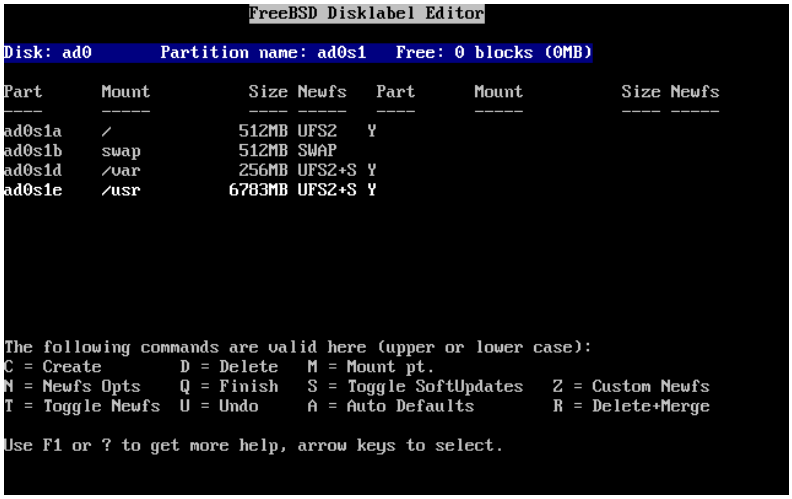
Figure 2-24. Choose the Root Mount Point



The display will then update to show you the newly created partition. You should repeat this procedure for the other partitions. When you create the swap partition, you will not be prompted for the filesystem mount point, as swap partitions are never mounted. When you create the final partition, `/usr`, you can leave the suggested size as is, to use the rest of the slice.

Your final FreeBSD DiskLabel Editor screen will appear similar to Figure 2-25, although your values chosen may be different. Press **Q** to finish.

Figure 2-25. Sysinstall Disklabel Editor



2.7 Choosing What to Install

2.7.1 Select the Distribution Set

Deciding which distribution set to install will depend largely on the intended use of the system and the amount of disk space available. The predefined options range from installing the smallest possible configuration to everything. Those who are new to UNIX and/or FreeBSD should almost certainly select one of these canned options. Customizing a distribution set is typically for the more experienced user.

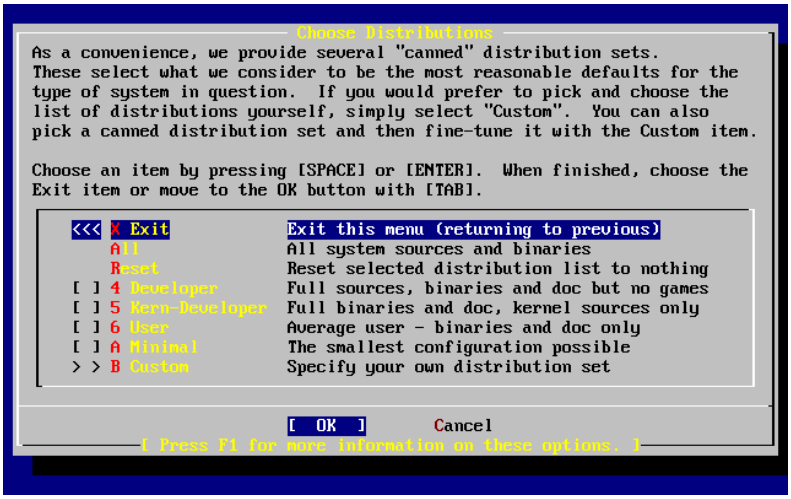
Press **F1** for more information on the distribution set options and what they contain. When finished reviewing the help, pressing **Enter** will return to the Select Distributions Menu.

If a graphical user interface is desired then the configuration of the X server and selection of a default desktop must be done after the installation of FreeBSD. More information regarding the installation and configuration of a X server can be found in Chapter 5.

If compiling a custom kernel is anticipated, select an option which includes the source code. For more information on why a custom kernel should be built or how to build a custom kernel, see Chapter 8.

Obviously, the most versatile system is one that includes everything. If there is adequate disk space, select All as shown in Figure 2-26 by using the arrow keys and press **Enter**. If there is a concern about disk space consider using an option that is more suitable for the situation. Do not fret over the perfect choice, as other distributions can be added after installation.

Figure 2-26. Choose Distributions



2.7.2 Installing the Ports Collection

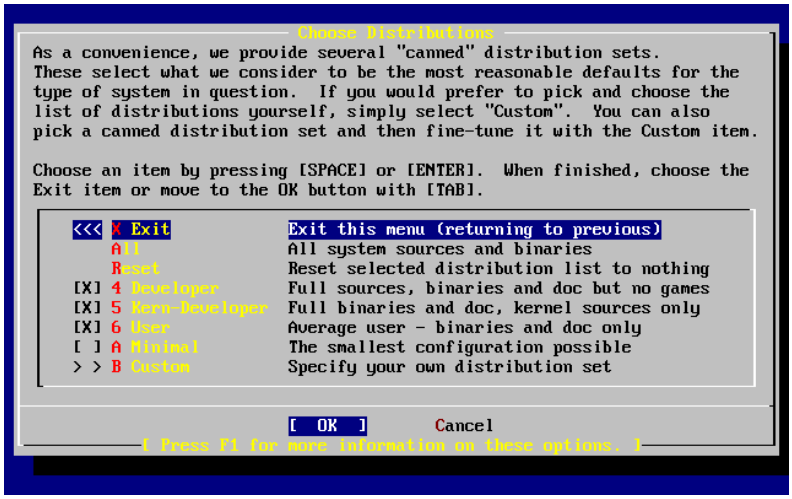
After selecting the desired distribution, an opportunity to install the FreeBSD Ports Collection is presented. The ports collection is an easy and convenient way to install software. The Ports Collection does not contain the source code necessary to compile the software. Instead, it is a collection of files which automates the downloading, compiling and installation of third-party software packages. Chapter 4 discusses how to use the ports collection.

The installation program does not check to see if you have adequate space. Select this option only if you have adequate hard disk space. As of FreeBSD 8.2, the FreeBSD Ports Collection takes up about 417 MB of disk space. You can safely assume a larger value for more recent versions of FreeBSD.

```
                        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
```

Select [Yes] with the arrow keys to install the Ports Collection or [No] to skip this option. Press **Enter** to continue. The Choose Distributions menu will redisplay.

Figure 2-27. Confirm Distributions



If satisfied with the options, select Exit with the arrow keys, ensure that [OK] is highlighted, and pressing **Enter** to continue.

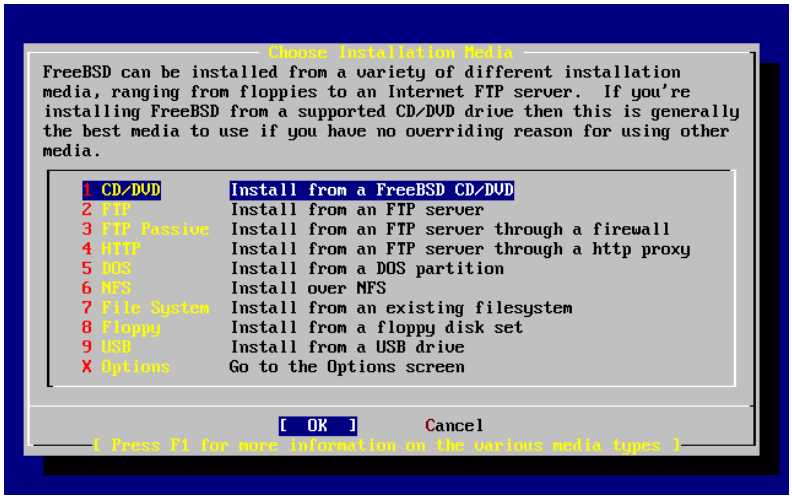
2.8 Choosing Your Installation Media

If Installing from a CDROM or DVD, use the arrow keys to highlight Install from a FreeBSD CD/DVD. Ensure that [OK] is highlighted, then press **Enter** to proceed with the installation.

For other methods of installation, select the appropriate option and follow the instructions.

Press **F1** to display the Online Help for installation media. Press **Enter** to return to the media selection menu.

Figure 2-28. Choose Installation Media



FTP Installation Modes: There are three FTP installation modes you can choose from: active FTP, passive FTP, or via a HTTP proxy.

FTP Active: Install from an FTP server

This option will make all FTP transfers use “Active” mode. This will not work through firewalls, but will often work with older FTP servers that do not support passive mode. If your connection hangs with passive mode (the default), try active!

FTP Passive: Install from an FTP server through a firewall

This option instructs **sysinstall** to use “Passive” mode for all FTP operations. This allows the user to pass through firewalls that do not allow incoming connections on random TCP ports.

FTP via a HTTP proxy: Install from an FTP server through a http proxy

This option instructs **sysinstall** to use the HTTP protocol (like a web browser) to connect to a proxy for all FTP operations. The proxy will translate the requests and send them to the FTP server. This allows the user to pass through firewalls that do not allow FTP at all, but offer a HTTP proxy. In this case, you have to specify the proxy in addition to the FTP server.

For a proxy FTP server, you should usually give the name of the server you really want as a part of the username, after an “@” sign. The proxy server then “fakes” the real server. For example, assuming you want to install from `ftp.FreeBSD.org`, using the proxy FTP server `foo.example.com`, listening on port 1234.

In this case, you go to the options menu, set the FTP username to `ftp@ftp.FreeBSD.org`, and the password to your email address. As your installation media, you specify FTP (or passive FTP, if the proxy supports it), and the URL `ftp://foo.example.com:1234/pub/FreeBSD`.

Since `/pub/FreeBSD` from `ftp.FreeBSD.org` is proxied under `foo.example.com`, you are able to install from *that* machine (which will fetch the files from `ftp.FreeBSD.org` as your installation requests them).

2.9 Committing to the Installation

The installation can now proceed if desired. This is also the last chance for aborting the installation to prevent changes to the hard drive.

```
                      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
```

Select [Yes] and press **Enter** to proceed.

The installation time will vary according to the distribution chosen, installation media, and the speed of the computer. There will be a series of messages displayed indicating the status.

The installation is complete when the following message is displayed:

```
                                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  ]
```

Press **Enter** to proceed with post-installation configurations.

Selecting [No] and pressing **Enter** will abort the installation so no changes will be made to your system. The following message will appear:

```
                                Message
Installation complete with some errors.  You may wish to scroll
through the debugging messages on VTY1 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 ]
```

This message is generated because nothing was installed. Pressing **Enter** will return to the Main Installation Menu to exit the installation.

2.10 Post-installation

Configuration of various options follows the successful installation. An option can be configured by re-entering the configuration options before booting the new FreeBSD system or after installation using `sysinstall` and selecting Configure.

2.10.1 Network Device Configuration

If you previously configured PPP for an FTP install, this screen will not display and can be configured later as described above.

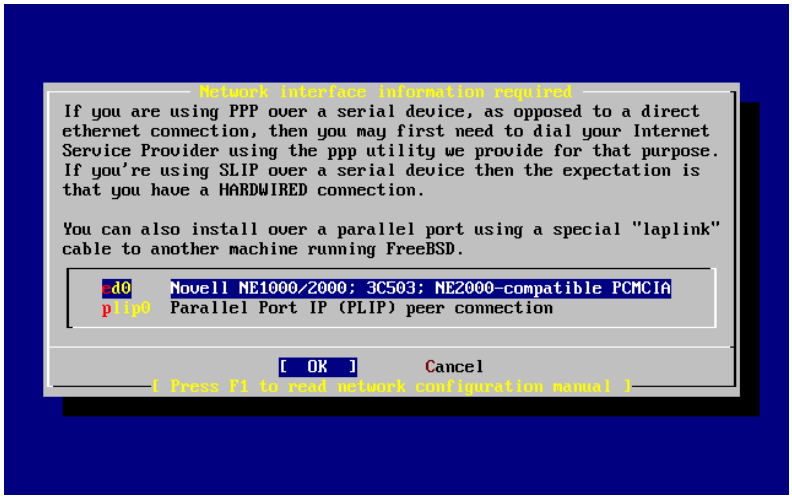
For detailed information on Local Area Networks and configuring FreeBSD as a gateway/router refer to the Advanced Networking chapter.

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

```
[ Yes ]   No
```

To configure a network device, select [Yes] and press **Enter**. Otherwise, select [No] to continue.

Figure 2-29. Selecting an Ethernet Device



Select the interface to be configured with the arrow keys and press **Enter**.

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

                          Yes   [ No ]
```

In this private local area network, the current Internet type protocol (IPv4) was sufficient and [No] was selected with the arrow keys and **Enter** pressed.

If you are connected to an existing IPv6 network with an RA server, then choose [Yes] and press **Enter**. It will take several seconds to scan for RA servers.

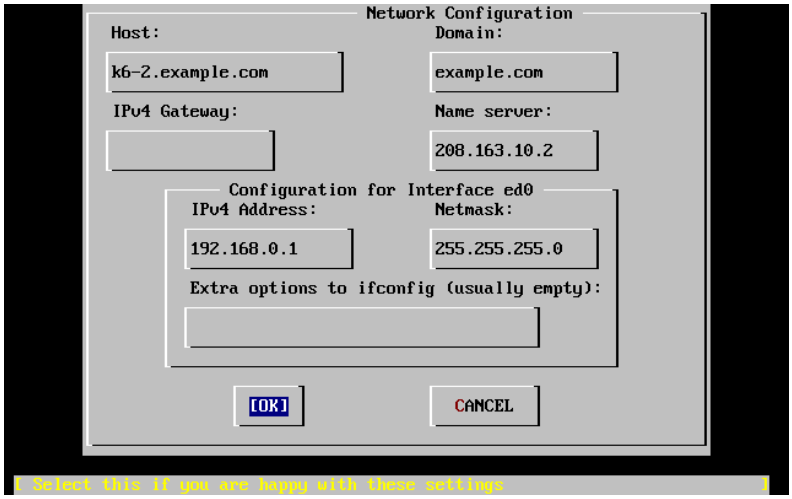
```
                      User Confirmation Requested
    Do you want to try DHCP configuration of the interface?

                          Yes   [ No ]
```

If DHCP (Dynamic Host Configuration Protocol) is not required select [No] with the arrow keys and press **Enter**.

Selecting [Yes] will execute **dhclient**, and if successful, will fill in the network configuration information automatically. Refer to Section 29.5 for more information.

The following Network Configuration screen shows the configuration of the Ethernet device for a system that will act as the gateway for a Local Area Network.

Figure 2-30. Set Network Configuration for ed0



Use **Tab** to select the information fields and fill in appropriate information:

Host

The fully-qualified hostname, such as `k6-2.example.com` in this case.

Domain

The name of the domain that your machine is in, such as `example.com` for this case.

IPv4 Gateway

IP address of host forwarding packets to non-local destinations. You must fill this in if the machine is a node on the network. *Leave this field blank* if the machine is the gateway to the Internet for the network. The IPv4 Gateway is also known as the default gateway or default route.

Name server

IP address of your local DNS server. There is no local DNS server on this private local area network so the IP address of the provider's DNS server (`208.163.10.2`) was used.

IPv4 address

The IP address to be used for this interface was `192.168.0.1`

Netmask

The address block being used for this local area network is `192.168.0.0` - `192.168.0.255` with a netmask of `255.255.255.0`.

Extra options to ifconfig

Any interface-specific options to `ifconfig` you would like to add. There were none in this case.

Use **Tab** to select [OK] when finished and press **Enter**.

```
User Confirmation Requested
```

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

                              [ Yes ]   No
```

Choosing [Yes] and pressing **Enter** will bring the machine up on the network and be ready for use. However, this does not accomplish much during installation, since the machine still needs to be rebooted.

2.10.2 Configure Gateway

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

                               [ Yes ]    No
```

If the machine will be acting as the gateway for a local area network and forwarding packets between other machines then select [Yes] and press **Enter**. If the machine is a node on a network then select [No] and press **Enter** to continue.

2.10.3 Configure Internet Services

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

                                 Yes   [ No ]
```

If [No] is selected, various services such **telnetd** will not be enabled. This means that remote users will not be able to **telnet** into this machine. Local users will still be able to access remote machines with **telnet**.

These services can be enabled after installation by editing `/etc/inetd.conf` with your favorite text editor. See Section 29.2.1 for more information.

Select [Yes] if you wish to configure these services during install. An additional confirmation will display:

```
                        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
```

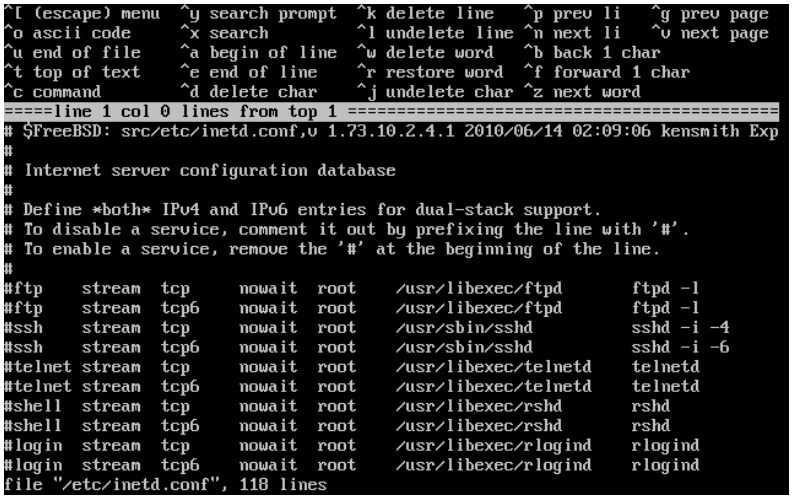
Select [Yes] to continue.

```
                        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
```

Selecting [Yes] will allow adding services by deleting the # at the beginning of a line.

Figure 2-31. Editing `inetd.conf`



After adding the desired services, pressing **Esc** will display a menu which will allow exiting and saving the changes.

2.10.4 Enabling SSH login

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

Selecting [Yes] will enable sshd(8), the daemon program for **OpenSSH**. This will allow secure remote access to your machine. For more information about **OpenSSH** see Section 14.10.

2.10.5 Anonymous FTP

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

                                Yes    [ No ]
```

2.10.5.1 Deny Anonymous FTP

Selecting the default [No] and pressing **Enter** will still allow users who have accounts with passwords to use FTP to access the machine.

2.10.5.2 Allow Anonymous FTP

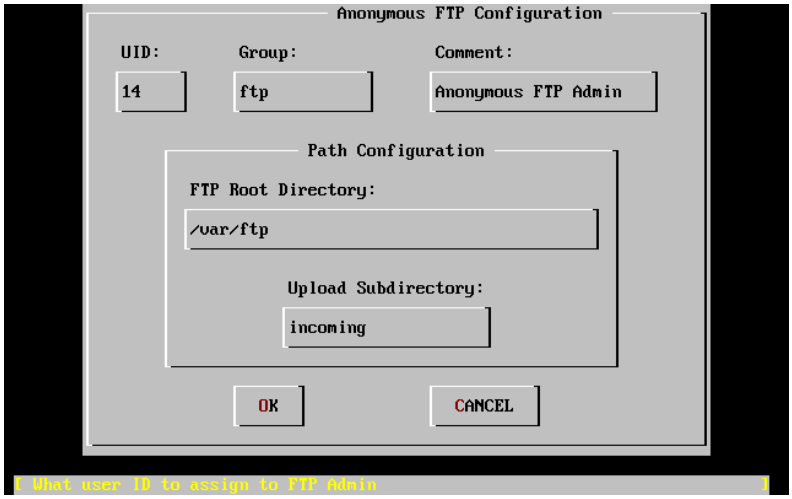
Anyone can access your machine if you elect to allow anonymous FTP connections. The security implications should be considered before enabling this option. For more information about security see Chapter 14.

To allow anonymous FTP, use the arrow keys to select [Yes] and press **Enter**. An additional confirmation will display:

```
                      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
```

This message informs you that the FTP service will also have to be enabled in `/etc/inetd.conf` if you want to allow anonymous FTP connections, see Section 2.10.3. Select [Yes] and press **Enter** to continue; the following screen will display:

Figure 2-32. Default Anonymous FTP Configuration



Use **Tab** to select the information fields and fill in appropriate information:

UID

The user ID you wish to assign to the anonymous FTP user. All files uploaded will be owned by this ID.

Group

Which group you wish the anonymous FTP user to be in.

Comment

String describing this user in `/etc/passwd`.

FTP Root Directory

Where files available for anonymous FTP will be kept.

Upload Subdirectory

Where files uploaded by anonymous FTP users will go.

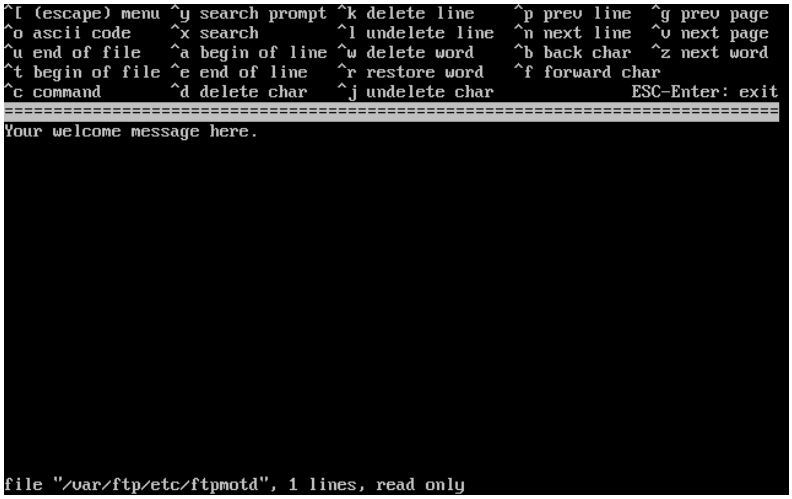
The FTP root directory will be put in `/var` by default. If you do not have enough room there for the anticipated FTP needs, the `/usr` directory could be used by setting the FTP root directory to `/usr/ftp`.

When you are satisfied with the values, press **Enter** to continue.

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

                              [ Yes ]    No
```

If you select [Yes] and press **Enter**, an editor will automatically start allowing you to edit the message.

Figure 2-33. Edit the FTP Welcome Message



This is a text editor called ee. Use the instructions to change the message or change the message later using a text editor of your choice. Note the file name/location at the bottom of the editor screen.

Press **Esc** and a pop-up menu will default to a) leave editor. Press **Enter** to exit and continue. Press **Enter** again to save changes if you made any.

2.10.6 Configure Network File System

Network File System (NFS) allows sharing of files across a network. A machine can be configured as a server, a client, or both. Refer to Section 29.3 for a more information.

2.10.6.1 NFS Server

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

                                 Yes    [ No ]
```

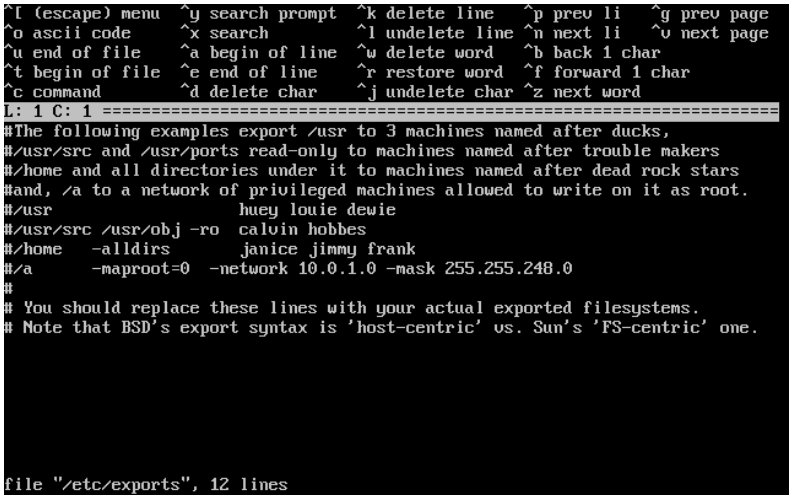
If there is no need for a Network File System server, select [No] and press **Enter**.

If [Yes] is chosen, a message will pop-up indicating that the `exports` file must be created.

```
                                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 ]
```

Press **Enter** to continue. A text editor will start allowing the `exports` file to be created and edited.

Figure 2-34. Editing `exports`



Use the instructions to add the actual exported filesystems now or later using a text editor of your choice. Note the file name/location at the bottom of the editor screen.

Press **Esc** and a pop-up menu will default to a) leave editor. Press **Enter** to exit and continue.

2.10.6.2 NFS Client

The NFS client allows your machine to access NFS servers.

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

                                  Yes   [ No ]
```

With the arrow keys, select [Yes] or [No] as appropriate and press **Enter**.

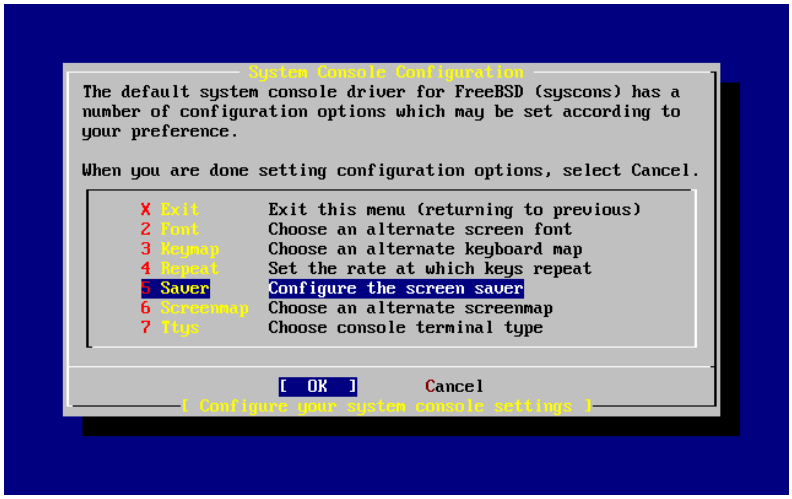
2.10.7 System Console Settings

There are several options available to customize the system console.

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

                                [ Yes ]  No
```

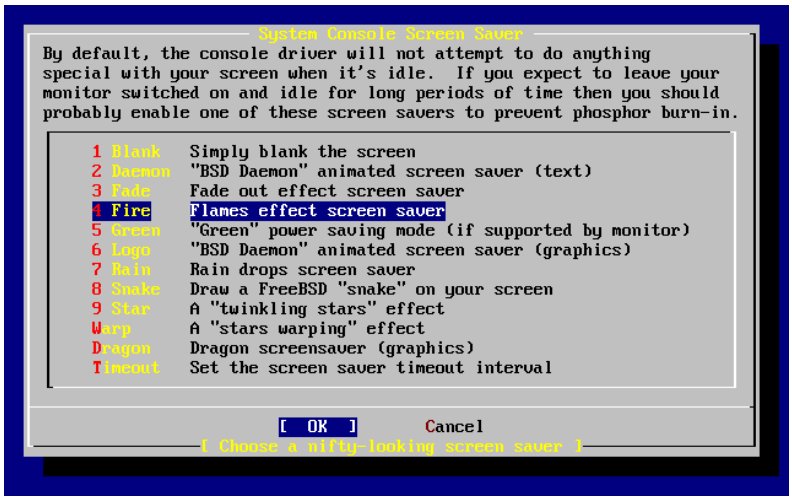
To view and configure the options, select [Yes] and press **Enter**.

Figure 2-35. System Console Configuration Options



A commonly used option is the screen saver. Use the arrow keys to select Saver and then press **Enter**.

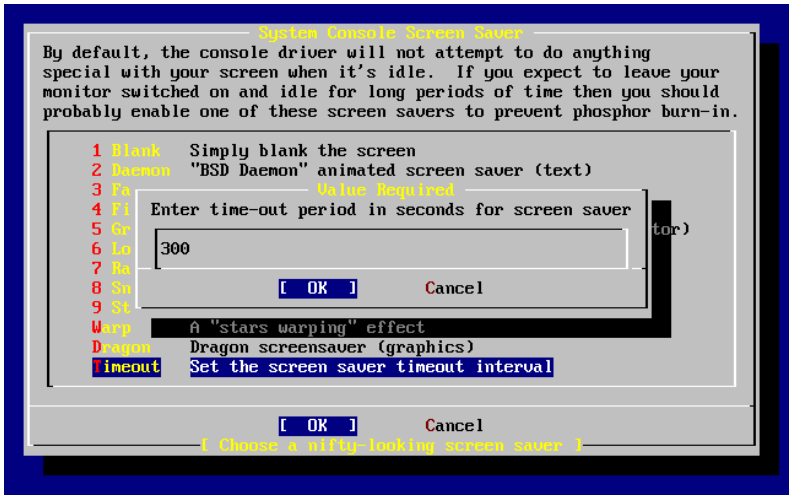
Figure 2-36. Screen Saver Options



Select the desired screen saver using the arrow keys and then press **Enter**. The System Console Configuration menu will redisplay.

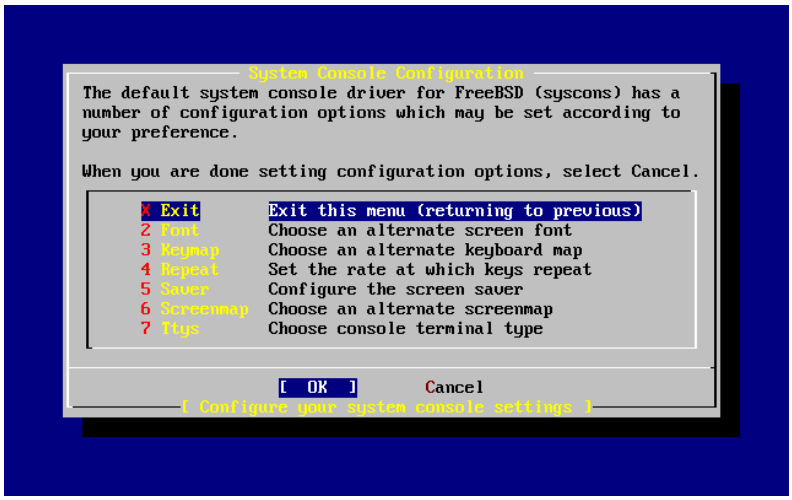
The default time interval is 300 seconds. To change the time interval, select Saver again. At the Screen Saver Options menu, select Timeout using the arrow keys and press **Enter**. A pop-up menu will appear:

Figure 2-37. Screen Saver Timeout



The value can be changed, then select [OK] and press **Enter** to return to the System Console Configuration menu.

Figure 2-38. System Console Configuration Exit



Selecting Exit and pressing **Enter** will continue with the post-installation configurations.

2.10.8 Setting the Time Zone

Setting the time zone for your machine will allow it to automatically correct for any regional time changes and perform other time zone related functions properly.

The example shown is for a machine located in the Eastern time zone of the United States. Your selections will vary according to your geographical location.

```
User Confirmation Requested
```

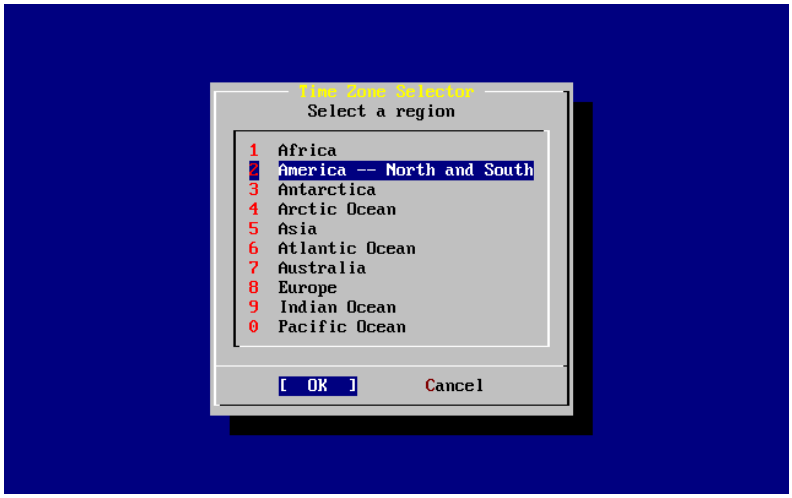
```
        Would you like to set this machine's time zone now?

                              [ Yes ]   No
```

Select [Yes] and press **Enter** to set the time zone.

```
                        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 ]
```

Select [Yes] or [No] according to how the machine's clock is configured and press **Enter**.

Figure 2-39. Select Your Region

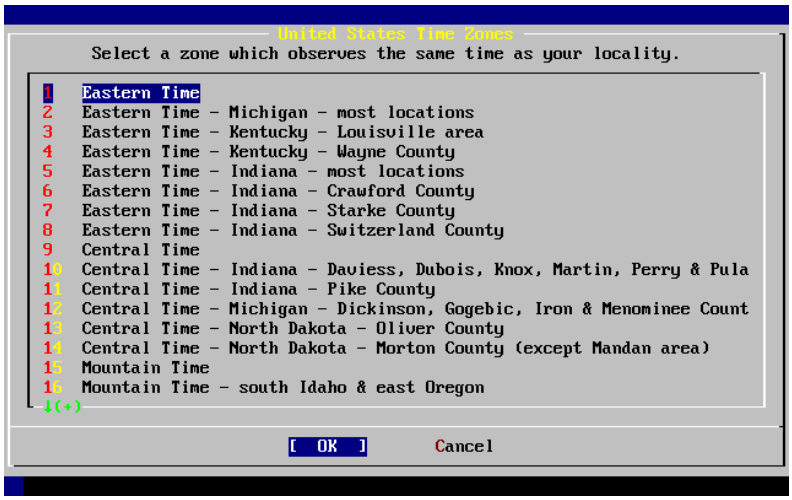


The appropriate region is selected using the arrow keys and then pressing **Enter**.

Figure 2-40. Select Your Country

Select the appropriate country using the arrow keys and press **Enter**.

Figure 2-41. Select Your Time Zone



The appropriate time zone is selected using the arrow keys and pressing **Enter**.

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

                    [ Yes ]   No
```

Confirm the abbreviation for the time zone is correct. If it looks okay, press **Enter** to continue with the post-installation configuration.

2.10.9 Linux Compatibility

Note: This part only applies to FreeBSD 7.X installation, if you install FreeBSD 8.X this screen will not be proposed.

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

                              [ Yes ]   No
```

Selecting [Yes] and pressing **Enter** will allow running Linux software on FreeBSD. The install will add the appropriate packages for Linux compatibility.

If installing by FTP, the machine will need to be connected to the Internet. Sometimes a remote ftp site will not have all the distributions like the Linux binary compatibility. This can be installed later if necessary.

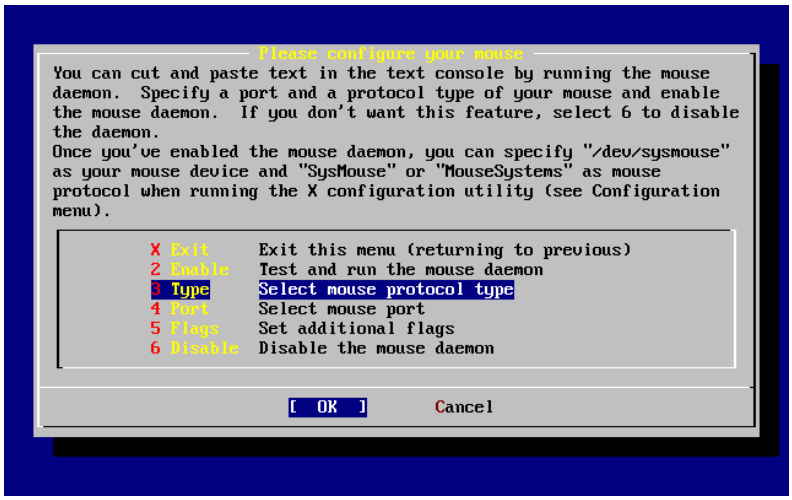
2.10.10 Mouse Settings

This option will allow you to cut and paste text in the console and user programs with a 3-button mouse. If using a 2-button mouse, refer to manual page, moused(8), after installation for details on emulating the 3-button style. This example depicts a non-USB mouse configuration (such as a PS/2 or COM port mouse):

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

                              [ Yes ]    No
```

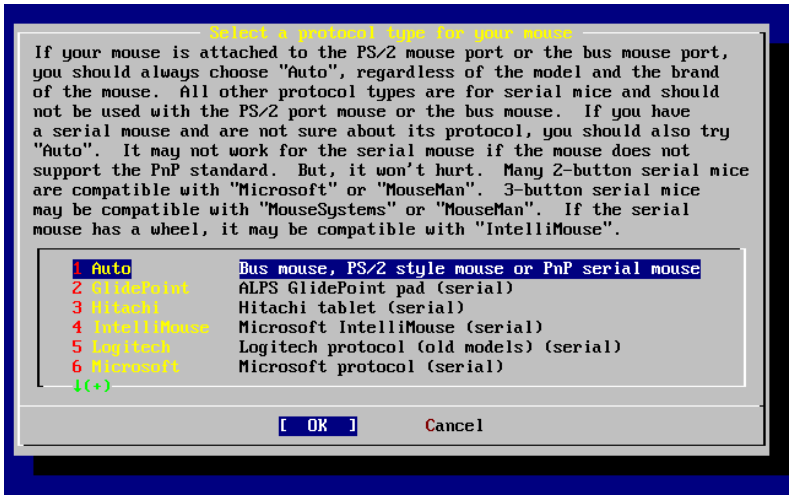
Select [Yes] for a PS/2, serial or bus mouse, or [No] for a USB mouse and press **Enter**.

Figure 2-42. Select Mouse Protocol Type



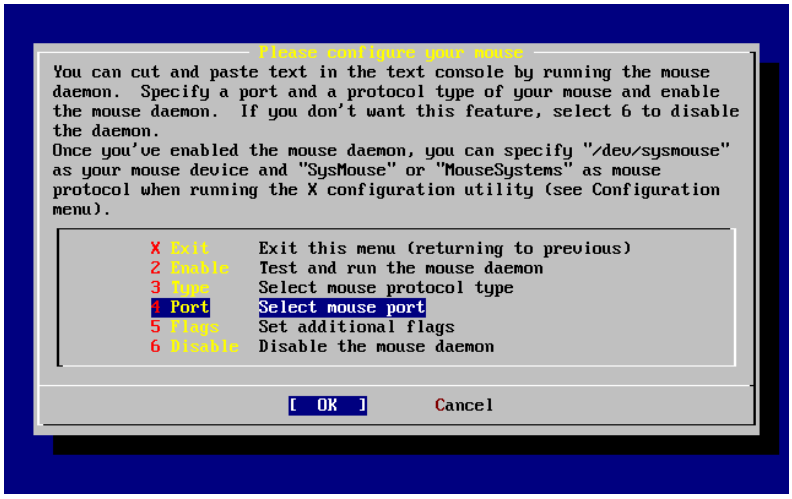
Use the arrow keys to select Type and press **Enter**.

Figure 2-43. Set Mouse Protocol



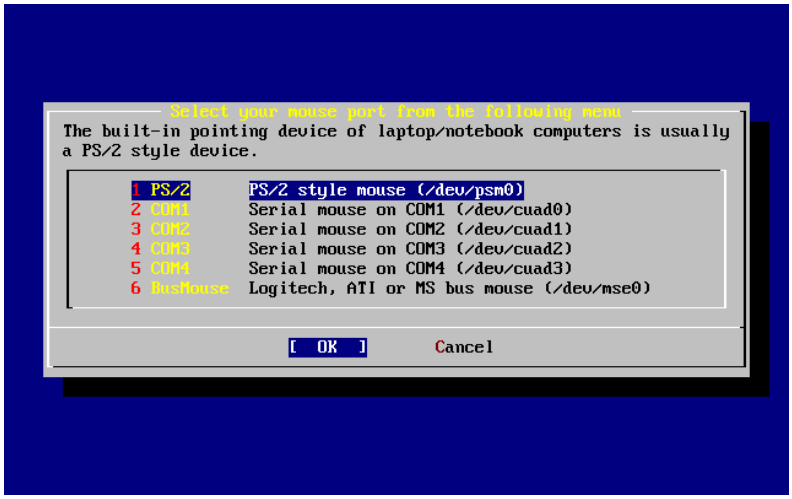
The mouse used in this example is a PS/2 type, so the default Auto was appropriate. To change protocol, use the arrow keys to select another option. Ensure that [OK] is highlighted and press **Enter** to exit this menu.

Figure 2-44. Configure Mouse Port



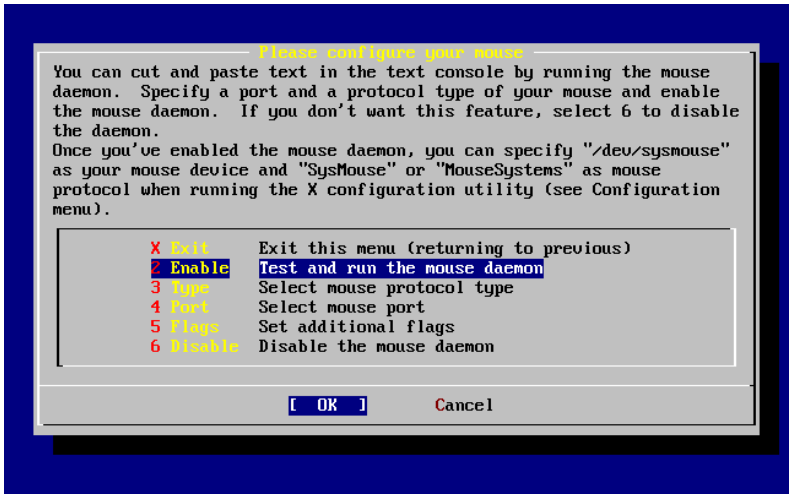
Use the arrow keys to select Port and press **Enter**.

Figure 2-45. Setting the Mouse Port



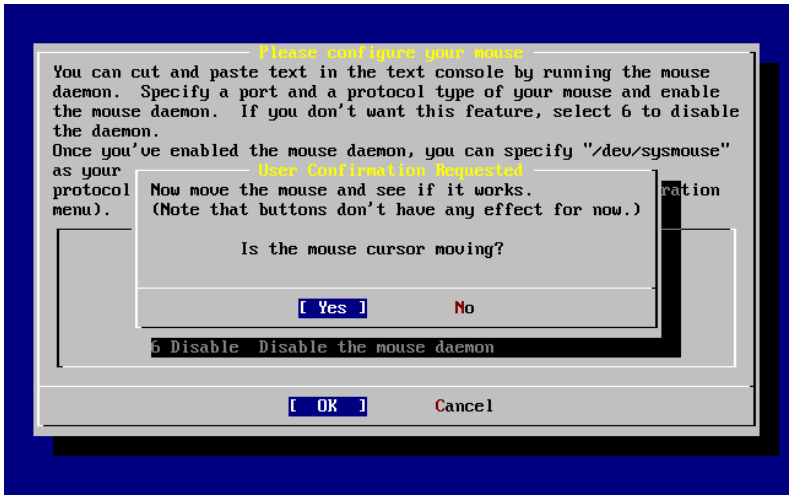
This system had a PS/2 mouse, so the default PS/2 was appropriate. To change the port, use the arrow keys and then press **Enter**.

Figure 2-46. Enable the Mouse Daemon



Last, use the arrow keys to select Enable, and press **Enter** to enable and test the mouse daemon.

Figure 2-47. Test the Mouse Daemon



Move the mouse around the screen and verify the cursor shown responds properly. If it does, select [Yes] and press **Enter**. If not, the mouse has not been configured correctly — select [No] and try using different configuration options.

Select Exit with the arrow keys and press **Enter** to return to continue with the post-installation configuration.

2.10.11 Install Packages

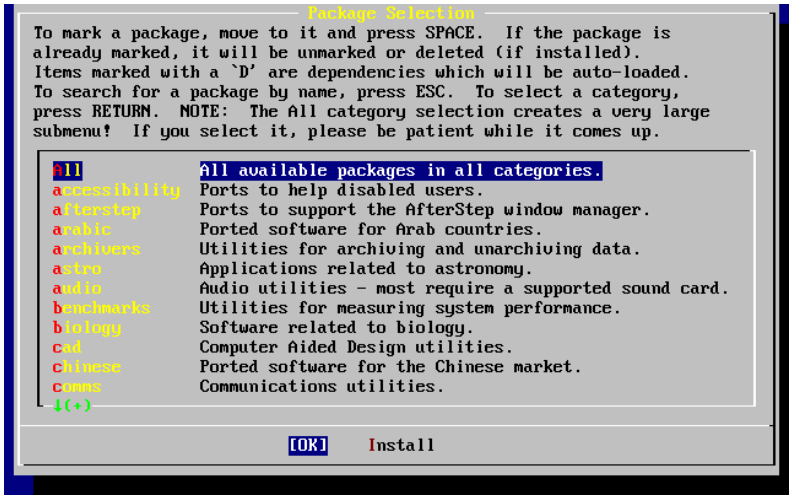
Packages are pre-compiled binaries and are a convenient way to install software.

Installation of one package is shown for purposes of illustration. Additional packages can also be added at this time if desired. After installation `sysinstall` can be used to add additional packages.

```
                    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
```

Selecting [Yes] and pressing **Enter** will be followed by the Package Selection screens:

Figure 2-48. Select Package Category

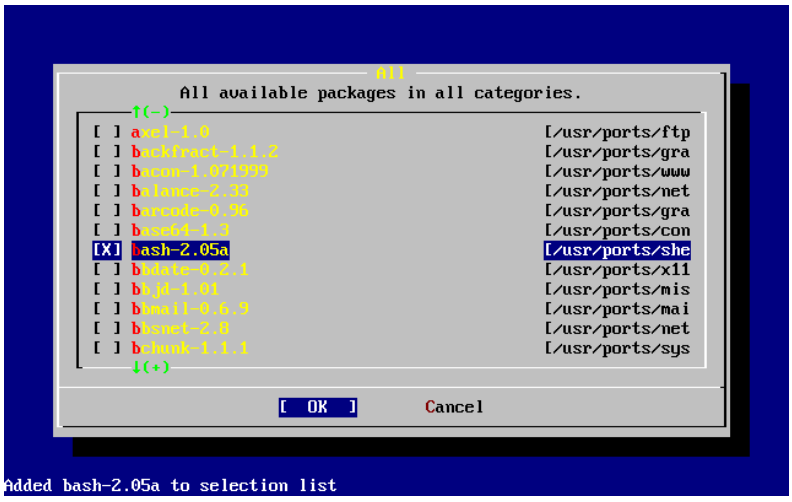


Only packages on the current installation media are available for installation at any given time.

All packages available will be displayed if All is selected or you can select a particular category. Highlight your selection with the arrow keys and press **Enter**.

A menu will display showing all the packages available for the selection made:

Figure 2-49. Select Packages



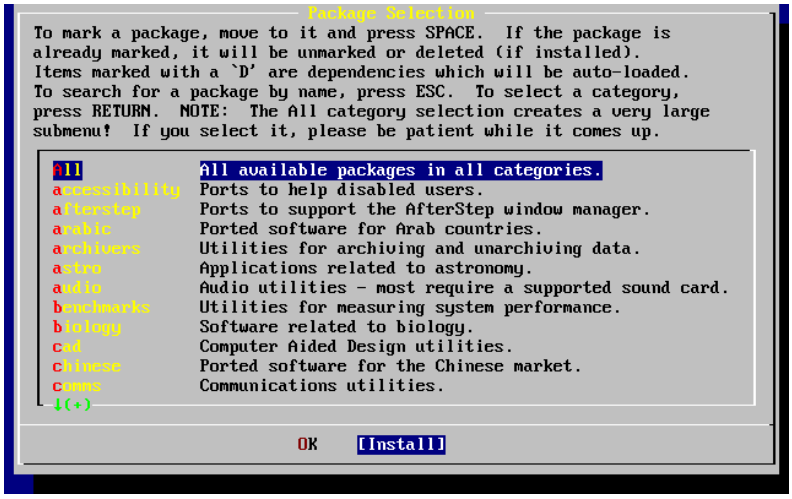
The **bash** shell is shown selected. Select as many as desired by highlighting the package and pressing the **Space** key. A short description of each package will appear in the lower left corner of the screen.

Pressing the **Tab** key will toggle between the last selected package, [OK], and [Cancel].

When you have finished marking the packages for installation, press **Tab** once to toggle to the [OK] and press **Enter** to return to the Package Selection menu.

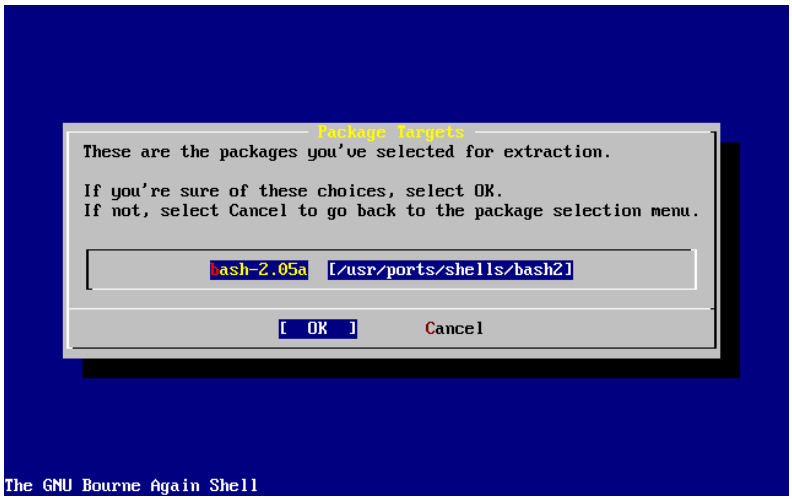
The left and right arrow keys will also toggle between [OK] and [Cancel]. This method can also be used to select [OK] and press **Enter** to return to the Package Selection menu.

Figure 2-50. Install Packages



Use the **Tab** and arrow keys to select [Install] and press **Enter**. You will then need to confirm that you want to install the packages:

Figure 2-51. Confirm Package Installation



Selecting [OK] and pressing **Enter** will start the package installation. Installing messages will appear until completed. Make note if there are any error messages.

The final configuration continues after packages are installed. If you end up not selecting any packages, and wish to return to the final configuration, select Install anyways.

2.10.12 Add Users/Groups

You should add at least one user during the installation so that you can use the system without being logged in as `root`. The root partition is generally small and running applications as `root` can quickly fill it. A bigger danger is noted below:

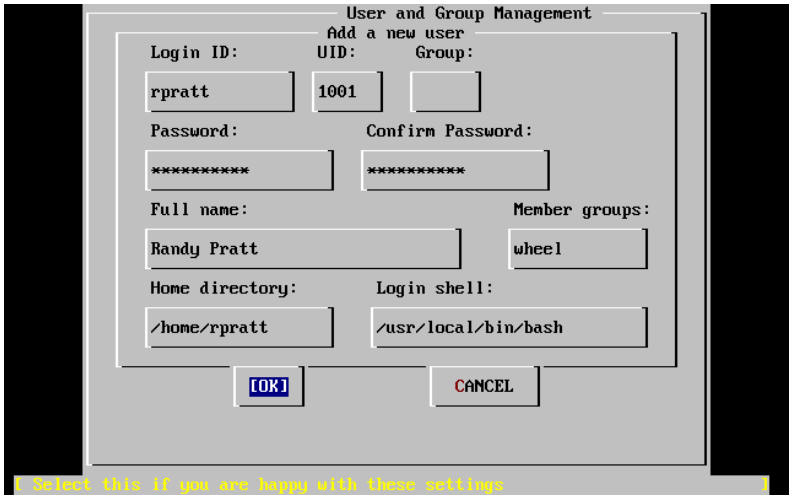
```
                    User Confirmation Requested
Would you like to add any initial user accounts to the system? Adding
at least one account for yourself at this stage is suggested since
working as the "root" user is dangerous (it is easy to do things which
adversely affect the entire system).

                              [ Yes ]   No
```

Select [Yes] and press **Enter** to continue with adding a user.

Figure 2-52. Select User

Select User with the arrow keys and press **Enter**.

Figure 2-53. Add User Information



The following descriptions will appear in the lower part of the screen as the items are selected with **Tab** to assist with entering the required information:

Login ID

The login name of the new user (mandatory).

UID

The numerical ID for this user (leave blank for automatic choice).

Group

The login group name for this user (leave blank for automatic choice).

Password

The password for this user (enter this field with care!).

Full name

The user’s full name (comment).

Member groups

The groups this user belongs to (i.e. gets access rights for).

Home directory

The user’s home directory (leave blank for default).

Login shell

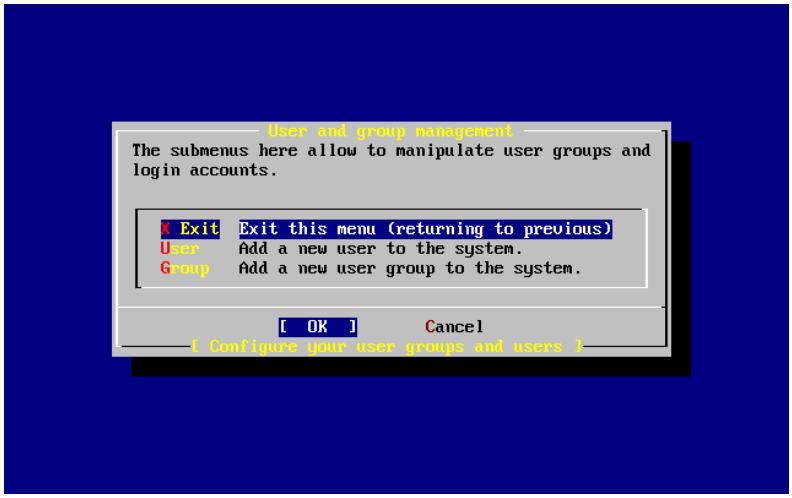
The user’s login shell (leave blank for default, e.g. `/bin/sh`).

The login shell was changed from `/bin/sh` to `/usr/local/bin/bash` to use the **bash** shell that was previously installed as a package. Do not try to use a shell that does not exist or you will not be able to login. The most common shell used in the BSD-world is the C shell, which can be indicated as `/bin/tcsh`.

The user was also added to the `wheel` group to be able to become a superuser with `root` privileges.

When you are satisfied, press [OK] and the User and Group Management menu will redisplay:

Figure 2-54. Exit User and Group Management



Groups can also be added at this time if specific needs are known. Otherwise, this may be accessed through using `sysinstall` after installation is completed.

When you are finished adding users, select Exit with the arrow keys and press **Enter** to continue the installation.

2.10.13 Set the `root` Password

```
                        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 ]
```

Press **Enter** to set the `root` password.

The password will need to be typed in twice correctly. Needless to say, make sure you have a way of finding the password if you forget. Notice that the password you type in is not echoed, nor are asterisks displayed.

```
New password:
Retype new password :
```

The installation will continue after the password is successfully entered.

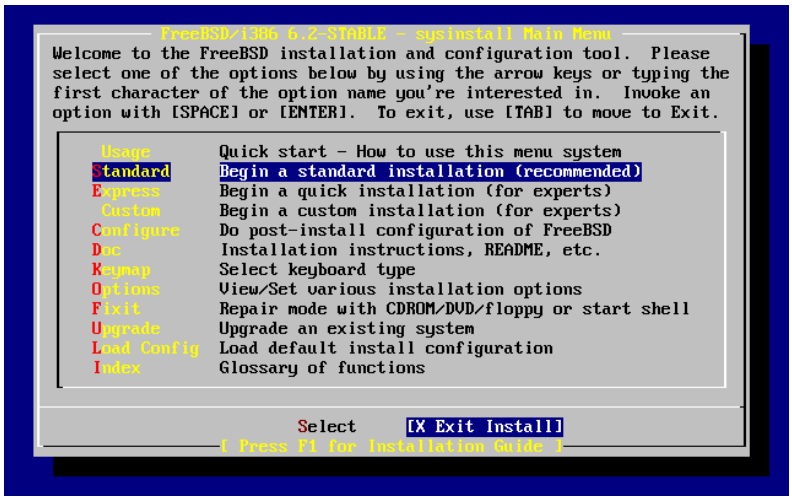
2.10.14 Exiting Install

If you need to configure additional network services or any other configuration, you can do it at this point or after installation with `sysinstall`.

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

                                Yes   [ No ]
```

Select [No] with the arrow keys and press **Enter** to return to the Main Installation Menu.

Figure 2-55. Exit Install



Select [X Exit Install] with the arrow keys and press **Enter**. You will be asked to confirm exiting the installation:

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

                              [ Yes ]   No
```

Select [Yes]. If you are booting from the CDROM drive the following message will remind you to remove the disk:

```
                     Message
Be sure to remove the media from the drive.

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

The CDROM drive is locked until the machine starts to reboot then the disk can be removed from drive (quickly). Press [OK] to reboot.

The system will reboot so watch for any error messages that may appear, see Section 2.10.16 for more details.

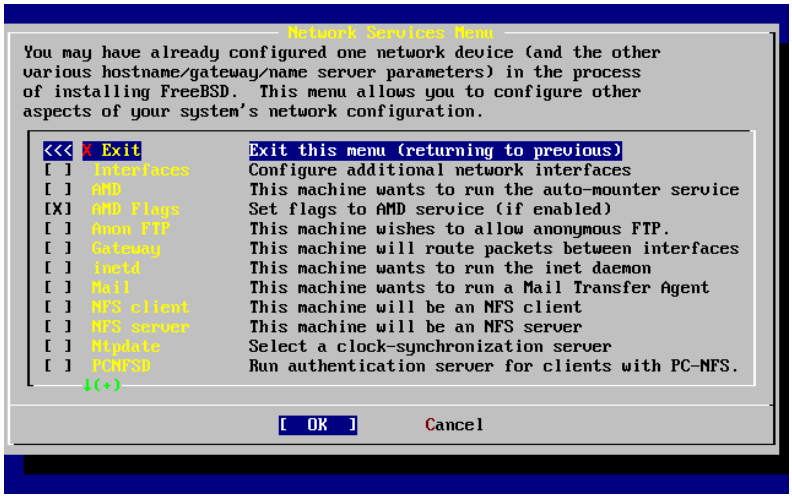
2.10.15 Configure Additional Network Services

Configuring network services can be a daunting task for new users if they lack previous knowledge in this area. Networking, including the Internet, is critical to all modern operating systems including FreeBSD; as a result, it is very useful to have some understanding FreeBSD's extensive networking capabilities. Doing this during the installation will ensure users have some understanding of the various services available to them.

Network services are programs that accept input from anywhere on the network. Every effort is made to make sure these programs will not do anything "harmful". Unfortunately, programmers are not perfect and through time there have been cases where bugs in network services have been exploited by attackers to do bad things. It is important that you only enable the network services you know that you need. If in doubt it is best if you do not enable a network service until you find out that you do need it. You can always enable it later by re-running **sysinstall** or by using the features provided by the `/etc/rc.conf` file.

Selecting the Networking option will display a menu similar to the one below:

Figure 2-56. Network Configuration Upper-level



The first option, Interfaces, was previously covered during the Section 2.10.1, thus this option can safely be ignored.

Selecting the AMD option adds support for the BSD automatic mount utility. This is usually used in conjunction with the NFS protocol (see below) for automatically mounting remote file systems. No special configuration is required here.

Next in line is the AMD Flags option. When selected, a menu will pop up for you to enter specific AMD flags. The menu already contains a set of default options:

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

The `-a` option sets the default mount location which is specified here as `/.amd_mnt`. The `-l` option specifies the default `log` file; however, when `syslogd` is used all log activity will be sent to the system log daemon. The `/host` directory is used to mount an exported file system from a remote host, while `/net` directory is used to mount an exported file system from an IP address. The `/etc/amd.map` file defines the default options for AMD exports.

The Anon FTP option permits anonymous FTP connections. Select this option to make this machine an anonymous FTP server. Be aware of the security risks involved with this option. Another menu will be displayed to explain the

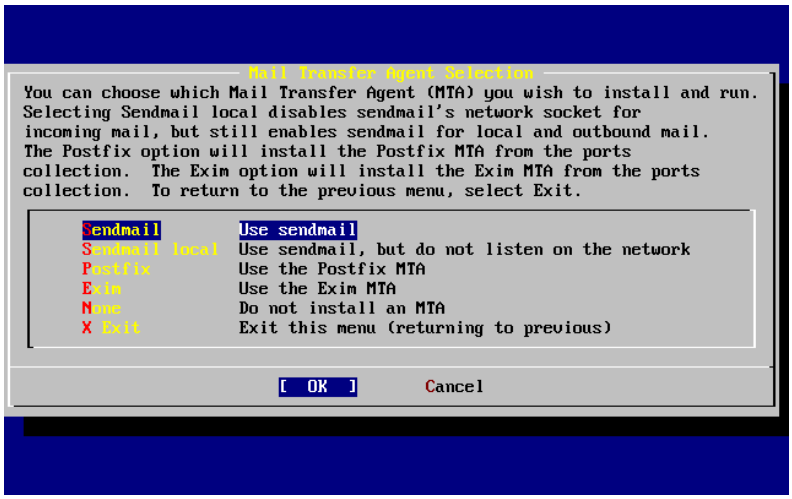
security risks and configuration in depth.

The Gateway configuration menu will set the machine up to be a gateway as explained previously. This can be used to unset the Gateway option if you accidentally selected it during the installation process.

The Inetd option can be used to configure or completely disable the inetd(8) daemon as discussed above.

The Mail option is used to configure the system's default MTA or Mail Transfer Agent. Selecting this option will bring up the following menu:

Figure 2-57. Select a default MTA



Here you are offered a choice as to which MTA to install and set as the default. An MTA is nothing more than a mail server which delivers email to users on the system or the Internet.

Selecting Sendmail will install the popular **sendmail** server which is the FreeBSD default. The Sendmail local option will set **sendmail** to be the default MTA, but disable its ability to receive incoming email from the Internet. The other options here, Postfix and Exim act similar to Sendmail. They both deliver email; however, some users prefer these alternatives to the **sendmail** MTA.

After selecting an MTA, or choosing not to select an MTA, the network configuration menu will appear with the next option being NFS client.

The NFS client option will configure the system to communicate with a server via NFS. An NFS server makes file systems available to other machines on the network via the NFS protocol. If this is a stand-alone machine, this option can remain unselected. The system may require more configuration later; see Section 29.3 for more information about client and server configuration.

Below that option is the NFS server option, permitting you to set the system up as an NFS server. This adds the required information to start up the RPC remote procedure call services. RPC is used to coordinate connections between hosts and programs.

Next in line is the Ntpdate option, which deals with time synchronization. When selected, a menu like the one below shows up:

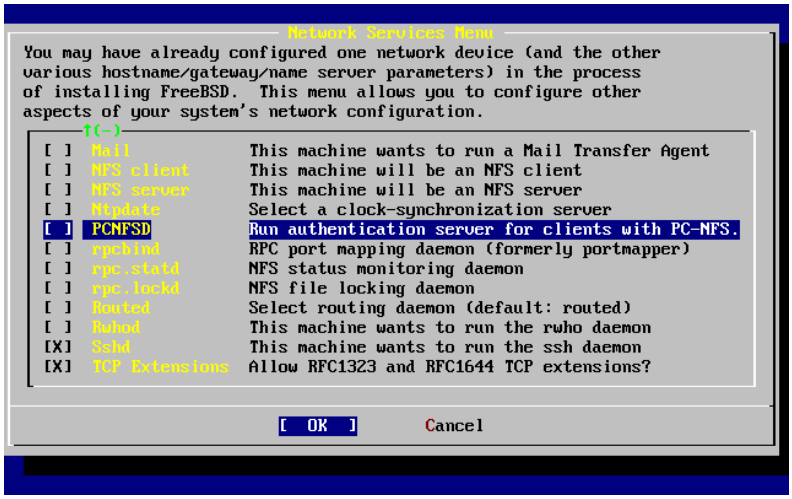
Figure 2-58. Ntpdate Configuration

From this menu, select the server which is the closest to your location. Selecting a close one will make the time synchronization more accurate as a server further from your location may have more connection latency.

The next option is the PCNFSD selection. This option will install the `net/pcnfsd` package from the Ports Collection. This is a useful utility which provides NFS authentication services for systems which are unable to provide their own, such as Microsoft's MS-DOS operating system.

Now you must scroll down a bit to see the other options:

Figure 2-59. Network Configuration Lower-level



The rpcbind(8), rpc.statd(8), and rpc.lockd(8) utilities are all used for Remote Procedure Calls (RPC). The `rpcbind` utility manages communication between NFS servers and clients, and is required for NFS servers to operate correctly. The **rpc.statd** daemon interacts with the **rpc.statd** daemon on other hosts to provide status monitoring. The reported status is usually held in the `/var/db/statd.status` file. The next option listed here is the rpc.lockd option, which, when selected, will provide file locking services. This is usually used with **rpc.statd** to monitor what

hosts are requesting locks and how frequently they request them. While these last two options are marvelous for debugging, they are not required for NFS servers and clients to operate correctly.

As you progress down the list the next item here is Routed, which is the routing daemon. The routed(8) utility manages network routing tables, discovers multicast routers, and supplies a copy of the routing tables to any physically connected host on the network upon request. This is mainly used for machines which act as a gateway for the local network. When selected, a menu will be presented requesting the default location of the utility. The default location is already defined for you and can be selected with the **Enter** key. You will then be presented with yet another menu, this time asking for the flags you wish to pass on to **routed**. The default is `-q` and it should already appear on the screen.

Next in line is the Rwhod option which, when selected, will start the rwhod(8) daemon during system initialization. The `rwhod` utility broadcasts system messages across the network periodically, or collects them when in "consumer" mode. More information can be found in the ruptime(1) and rwho(1) manual pages.

The next to the last option in the list is for the sshd(8) daemon. This is the secure shell server for **OpenSSH** and it is highly recommended over the standard **telnet** and FTP servers. The **sshd** server is used to create a secure connection from one host to another by using encrypted connections.

Finally there is the TCP Extensions option. This enables the TCP Extensions defined in RFC 1323 and RFC 1644. While on many hosts this can speed up connections, it can also cause some connections to be dropped. It is not recommended for servers, but may be beneficial for stand alone machines.

Now that you have configured the network services, you can scroll up to the very top item which is X Exit and continue on to the next configuration item or simply exit **sysinstall** in selecting X Exit twice then [X Exit Install].

2.10.16 FreeBSD Bootup

2.10.16.1 FreeBSD/i386 Bootup

If everything went well, you will see messages scroll off the screen and you will arrive at a login prompt. You can view the content of the messages by pressing **Scroll-Lock** and using **PgUp** and **PgDn**. Pressing **Scroll-Lock** again will return to the prompt.

The entire message may not display (buffer limitation) but it can be viewed from the command line after logging in by typing `dmesg` at the prompt.

Login using the username/password you set during installation (`rpratt`, in this example). Avoid logging in as `root` except when necessary.

Typical boot messages (version information omitted):

```
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
atkbdc0: <keyboard controller (i8042)> at port 0x60-0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq 1 on atkbdc0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbdc0
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/ad0s1a
swapon: adding /dev/ad0s1b as swap device
Automatic boot in progress...
/dev/ad0s1a: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1a: clean, 48752 free (552 frags, 6025 blocks, 0.9% fragmentation)
/dev/ad0s1f: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1f: clean, 128997 free (21 frags, 16122 blocks, 0.0% fragmentation)
/dev/ad0s1g: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1g: clean, 3036299 free (43175 frags, 374073 blocks, 1.3% fragmentation)
/dev/ad0s1e: filesystem CLEAN; SKIPPING CHECKS
/dev/ad0s1e: 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 0xffffff00 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:
```

Generating the RSA and DSA keys may take some time on slower machines. This happens only on the initial boot-up of a new installation. Subsequent boots will be faster.

If the X server has been configured and a Default Desktop chosen, it can be started by typing `startx` at the command line.

2.10.17 FreeBSD Shutdown

It is important to properly shutdown the operating system. Do not just turn off power. First, become a superuser by typing `su` at the command line and entering the `root` password. This will work only if the user is a member of the `wheel` group. Otherwise, login as `root` and use `shutdown -h now`.

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

It is safe to turn off the power after the shutdown command has been issued and the message “Please press any key to reboot” appears. If any key is pressed instead of turning off the power switch, the system will reboot.

You could also use the **Ctrl**+**Alt**+**Del** key combination to reboot the system, however this is not recommended during normal operation.

2.11 Troubleshooting

The following section covers basic installation troubleshooting, such as common problems people have reported. There are also a few questions and answers for people wishing to dual-boot FreeBSD with MS-DOS or Windows.

2.11.1 What to Do If Something Goes Wrong

Due to various limitations of the PC architecture, it is impossible for probing to be 100% reliable, however, there are a few things you can do if it fails.

Check the Hardware Notes (http://www.FreeBSD.org/releases/index.html) document for your version of FreeBSD to make sure your hardware is supported.

If your hardware is supported and you still experience lock-ups or other problems, you will need to build a custom kernel. This will allow you to add in support for devices which are not present in the `GENERIC` kernel. The kernel on the boot disks is configured assuming that most hardware devices are in their factory default configuration in terms of IRQs, IO addresses, and DMA channels. If your hardware has been reconfigured, you will most likely need to edit the kernel configuration and recompile to tell FreeBSD where to find things.

It is also possible that a probe for a device not present will cause a later probe for another device that is present to fail. In that case, the probes for the conflicting driver(s) should be disabled.

> **Note:** Some installation problems can be avoided or alleviated by updating the firmware on various hardware components, most notably the motherboard. The motherboard firmware may also be referred to as BIOS and most of the motherboard or computer manufactures have a website where the upgrades and upgrade information may be located.
>
> Most manufacturers strongly advise against upgrading the motherboard BIOS unless there is a good reason for doing so, which could possibly be a critical update of sorts. The upgrade process *can* go wrong, causing permanent damage to the BIOS chip.

2.11.2 Using MS-DOS® and Windows® File Systems

At this time, FreeBSD does not support file systems compressed with the **Double Space™** application. Therefore the file system will need to be uncompressed before FreeBSD can access the data. This can be done by running the **Compression Agent** located in the Start> Programs > System Tools menu.

FreeBSD can support MS-DOS file systems (sometimes called FAT file systems). The mount_msdosfs(8) command grafts such file systems onto the existing directory hierarchy, allowing the file system's contents to be accessed. The mount_msdosfs(8) program is not usually invoked directly; instead, it is called by the system through a line in `/etc/fstab` or by a call to the mount(8) utility with the appropriate parameters.

A typical line in `/etc/fstab` is:

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

> **Note:** The `/dos` directory must already exist for this to work. For details about the format of `/etc/fstab`, see fstab(5).

A typicall call to mount(8) for a MS-DOS file system looks like:

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

In this example, the MS-DOS file system is located on the first partition of the primary hard disk. Your situation may be different, check the output from the `dmesg`, and `mount` commands. They should produce enough information to give an idea of the partition layout.

> **Note:** FreeBSD may number disk slices (that is, MS-DOS partitions) differently than other operating systems. In particular, extended MS-DOS partitions are usually given higher slice numbers than primary MS-DOS partitions. The fdisk(8) utility can help determine which slices belong to FreeBSD and which belong to other operating systems.

NTFS partitions can also be mounted in a similar manner using the mount_ntfs(8) command.

2.11.3 Troubleshooting Questions and Answers

1. My system hangs while probing hardware during boot, or it behaves strangely during install, or the floppy drive is not probed.

FreeBSD makes extensive use of the system ACPI service on the i386, amd64 and ia64 platforms to aid in system configuration if it is detected during boot. Unfortunately, some bugs still exist in both the ACPI driver and within system motherboards and BIOS. The use of ACPI can be disabled by setting the `hint.acpi.0.disabled` hint in the third stage boot loader:

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

This is reset each time the system is booted, so it is necessary to add `hint.acpi.0.disabled="1"` to the file `/boot/loader.conf`. More information about the boot loader can be found in Section 12.1.

2. I go to boot from the hard disk for the first time after installing FreeBSD, the kernel loads and probes my hardware, but stops with messages like:

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

What is wrong? What can I do?

What is this `bios_drive:interface(unit,partition)kernel_name` thing that is displayed with the boot help?

There is a longstanding problem in the case where the boot disk is not the first disk in the system. The BIOS uses a different numbering scheme to FreeBSD, and working out which numbers correspond to which is difficult to get right.

In the case where the boot disk is not the first disk in the system, FreeBSD can need some help finding it. There are two common situations here, and in both of these cases, you need to tell FreeBSD where the root filesystem is. You do this by specifying the BIOS disk number, the disk type and the FreeBSD disk number for that type.

The first situation is where you have two IDE disks, each configured as the master on their respective IDE busses, and wish to boot FreeBSD from the second disk. The BIOS sees these as disk 0 and disk 1, while FreeBSD sees them as `ad0` and `ad2`.

FreeBSD is on BIOS disk 1, of type `ad` and the FreeBSD disk number is 2, so you would say:

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

Note that if you have a slave on the primary bus, the above is not necessary (and is effectively wrong).

The second situation involves booting from a SCSI disk when you have one or more IDE disks in the system. In this case, the FreeBSD disk number is lower than the BIOS disk number. If you have two IDE disks as well as the SCSI disk, the SCSI disk is BIOS disk 2, type `da` and FreeBSD disk number 0, so you would say:

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

To tell FreeBSD that you want to boot from BIOS disk 2, which is the first SCSI disk in the system. If you only had one IDE disk, you would use `1:` instead.

Once you have determined the correct values to use, you can put the command exactly as you would have typed it in the `/boot.config` file using a standard text editor. Unless instructed otherwise, FreeBSD will use the contents of this file as the default response to the `boot:` prompt.

3. I go to boot from the hard disk for the first time after installing FreeBSD, but the Boot Manager prompt just prints `F?` at the boot menu each time but the boot will not go any further.

The hard disk geometry was set incorrectly in the partition editor when you installed FreeBSD. Go back into the partition editor and specify the actual geometry of your hard disk. You must reinstall FreeBSD again from the beginning with the correct geometry.

If you are failing entirely in figuring out the correct geometry for your machine, here is a tip: Install a small DOS partition at the beginning of the disk and install FreeBSD after that. The install program will see the DOS partition and try to infer the correct geometry from it, which usually works.

The following tip is no longer recommended, but is left here for reference:

> If you are setting up a truly dedicated FreeBSD server or workstation where you do not care for (future) compatibility with DOS, Linux or another operating system, you also have got the option to use the entire disk (`A` in the partition editor), selecting the non-standard option where FreeBSD occupies the entire disk from the very first to the very last sector. This will leave all geometry considerations aside, but is somewhat limiting unless you're never going to run anything other than FreeBSD on a disk.

4. The system finds my ed(4) network card, but I keep getting device timeout errors.

Your card is probably on a different IRQ from what is specified in the `/boot/device.hints` file. The ed(4) driver does not use the "soft" configuration by default (values entered using EZSETUP in DOS), but it will use the software configuration if you specify `-1` in the hints for the interface.

Either move the jumper on the card to a hard configuration setting (altering the kernel settings if necessary), or specify the IRQ as `-1` by setting the hint `hint.ed.0.irq="-1"` This will tell the kernel to use the soft configuration.

Another possibility is that your card is at IRQ 9, which is shared by IRQ 2 and frequently a cause of problems (especially when you have a VGA card using IRQ 2!). You should not use IRQ 2 or 9 if at all possible.

5. When **sysinstall** is used in an X11 terminal, the yellow font is difficult to read against the light gray background. Is there a way to provide higher contrast for this application?

If you already have X11 installed and the default colors chosen by **sysinstall** make text illegible while using xterm(1) or rxvt(1), add the following to your `~/.Xdefaults` to get a darker background gray: `XTerm*color7: #c0c0c0`

2.12 Advanced Installation Guide

This section describes how to install FreeBSD in exceptional cases.

2.12.1 Installing FreeBSD on a System without a Monitor or Keyboard

This type of installation is called a “headless install”, because the machine that you are trying to install FreeBSD on either does not have a monitor attached to it, or does not even have a VGA output. How is this possible you ask? Using a serial console. A serial console is basically using another machine to act as the main display and keyboard for a system. To do this, just follow the steps to create an installation USB memstick, explained in Section 2.3.7 or download the correct installation ISO image see Section 2.13.1.

To modify these media to boot into a serial console, follow these steps (If you want to use a CDROM you can skip the first step):

1. Enabling the Installation USB Stick to Boot into a Serial Console

   If you were to boot into the USB stick that you just made, FreeBSD would boot into its normal install mode. We want FreeBSD to boot into a serial console for our install. To do this, you have to mount the USB disk onto your FreeBSD system using the mount(8) command.

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

   Note: Adapt the device node and the mount point to your situation.

   Now that you have the stick mounted, you must set the USB stick to boot into a serial console. You have to add to the `loader.conf` file of the USB stick file system a line setting the serial console as the system console:

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

   Now that you have your USB stick configured correctly, you must unmount the disk using the umount(8) command:

   ```
   # umount /mnt
   ```

   Now you can unplug the USB stick and jump directly to the third step of this procedure.

2. Enabling the Installation CD to Boot into a Serial Console

   If you were to boot into the CD that you just made from the installation ISO image (see Section 2.13.1), FreeBSD would boot into its normal install mode. We want FreeBSD to boot into a serial console for our install. To do this, you have to extract, modify and regenerate the ISO image before burning it on a CD-R media.

   From the FreeBSD system where is saved the installation ISO image, for example `FreeBSD-8.1-RELEASE-i386-disc1.iso`, use the tar(1) utility to extract all the files:

   ```
   # mkdir /path/to/headless-iso
   # tar -C /path/to/headless-iso -pxvf FreeBSD-8.1-RELEASE-i386-disc1.iso
   ```

   Now you must set the installation media to boot into a serial console. You have to add to the `loader.conf` file from the extracted ISO image a line setting the serial console as the system console:

   ```
   # echo 'console="comconsole"' >> /path/to/headless-iso/boot/loader.conf
   ```

   Then we can create a new ISO image from the modified tree. The mkisofs(8) tool from the `sysutils/cdrtools` port is used:

   ```
   # mkisofs -v -b boot/cdboot -no-emul-boot -r -J -V "Headless_install" \
        -o Headless-FreeBSD-8.1-RELEASE-i386-disc1.iso /path/to/headless-iso
   ```

   Now that you have your ISO image configured correctly, you can burn it on a CD-R with your favorite burning application.

3. Connecting Your Null-modem Cable

   You now need to connect a null-modem cable between the two machines. Just connect the cable to the serial ports of the 2 machines. *A normal serial cable will not work here*, you need a null-modem cable because it has some of the wires inside crossed over.

4. Booting Up for the Install

   It is now time to go ahead and start the install. Plug in the USB memstick on the machine you are doing the headless install on, and power on the machine. If you are using a prepared CDROM, power on the machine and insert the disk to boot on.

5. Connecting to Your Headless Machine

   Now you have to connect to that machine with cu(1):

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

   On FreeBSD 7.X use the following command instead:

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

That's it! You should now be able to control the headless machine through your `cu` session. It will load the kernel and then it will come up with a selection of what kind of terminal to use. Select the FreeBSD color console and proceed with your install!

2.13 Preparing Your Own Installation Media

> **Note:** To prevent repetition, "FreeBSD disc" in this context means a FreeBSD CDROM or DVD that you have purchased or produced yourself.

There may be some situations in which you need to create your own FreeBSD installation media and/or source. This might be physical media, such as a tape, or a source that **sysinstall** can use to retrieve the files, such as a local FTP site, or an MS-DOS partition.

For example:

- You have many machines connected to your local network, and one FreeBSD disc. You want to create a local FTP site using the contents of the FreeBSD disc, and then have your machines use this local FTP site instead of needing to connect to the Internet.
- You have a FreeBSD disc, and FreeBSD does not recognize your CD/DVD drive, but MS-DOS/Windows does. You want to copy the FreeBSD installation files to a DOS partition on the same computer, and then install FreeBSD using those files.
- The computer you want to install on does not have a CD/DVD drive or a network card, but you can connect a "Laplink-style" serial or parallel cable to a computer that does.
- You want to create a tape that can be used to install FreeBSD.

2.13.1 Creating an Installation CDROM

As part of each release, the FreeBSD project makes available at least two CDROM images (“ISO images”) per supported architecture. These images can be written (“burned”) to CDs if you have a CD writer, and then used to install FreeBSD. If you have a CD writer, and bandwidth is cheap, then this is the easiest way to install FreeBSD.

1. Download the Correct ISO Images

   The ISO images for each release can be downloaded from `ftp://ftp.FreeBSD.org/pub/FreeBSD/ISO-IMAGES-`*`arch`*`/`*`version`* or the closest mirror. Substitute *`arch`* and *`version`* as appropriate.

   That directory will normally contain the following images:

   Table 2-4. FreeBSD 7.*X* and 8.*X* ISO Image Names and Meanings

Filename	Contents
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-bootonly.iso`	This CD image allows you to start the installation process by booting from a CD-ROM drive but it does not contain the support for installing FreeBSD from the CD itself. You would need to perform a network based install (e.g. from an FTP server) after booting from this CD.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-dvd1.iso.gz`	This DVD image contains everything necessary to install the base FreeBSD operating system, a collection of pre-built packages, and the documentation. It also supports booting into a “livefs” based rescue mode.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-memstick.img`	This image can be written to an USB memory stick and used to do an install on machines capable of booting off USB drives. It also supports booting into a “livefs” based rescue mode. The documentation packages are provided but no other packages. This image is not available for FreeBSD 7.3 and earlier.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-disc1.iso`	This CD image contains the base FreeBSD operating system and the documentation packages but no other packages.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-disc2.iso`	A CD image with as many third-party packages as would fit on the disc. This image is not available for FreeBSD 8.0 and later.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-disc3.iso`	Another CD image with as many third-party packages as would fit on the disc. This image is not available for FreeBSD 8.0 and later.
`version``-RELEASE-`*`arch`*`-docs.iso`	The FreeBSD documentation.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-livefs.iso`	This CD image contains support for booting into a “livefs” based rescue mode but does not support doing an install from the CD itself.

Note: FreeBSD 7.X releases before FreeBSD 7.3 and FreeBSD 8.X releases before FreeBSD 8.1 used a different naming convention. The names of their ISO images are not prefixed with `FreeBSD-`.

You *must* download one of either the `bootonly` ISO image (if available), or the image of `disc1`. Do not download both of them, since the `disc1` image contains everything that the `bootonly` ISO image contains.

Use the `bootonly` ISO if Internet access is cheap for you. It will let you install FreeBSD, and you can then install third-party packages by downloading them using the ports/packages system (see Chapter 4) as necessary.

Use the image of `dvd1` if you want to install a FreeBSD release and want a reasonable selection of third-party packages on the disc as well.

The additional disc images are useful, but not essential, especially if you have high-speed access to the Internet.

2. Write the CDs

   You must then write the CD images to disc. If you will be doing this on another FreeBSD system then see Section 18.6 for more information (in particular, Section 18.6.3 and Section 18.6.4).

   If you will be doing this on another platform then you will need to use whatever utilities exist to control your CD writer on that platform. The images provided are in the standard ISO format, which many CD writing applications support.

Note: If you are interested in building a customized release of FreeBSD, please see the Release Engineering Article (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/releng).

2.13.2 Creating a Local FTP Site with a FreeBSD Disc

FreeBSD discs are laid out in the same way as the FTP site. This makes it very easy for you to create a local FTP site that can be used by other machines on your network when installing FreeBSD.

1. On the FreeBSD computer that will host the FTP site, ensure that the CDROM is in the drive, and mounted on `/cdrom`.

   ```
   # mount /cdrom
   ```

2. Create an account for anonymous FTP in `/etc/passwd`. Do this by editing `/etc/passwd` using vipw(8) and adding this line:

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

3. Ensure that the FTP service is enabled in `/etc/inetd.conf`.

Anyone with network connectivity to your machine can now chose a media type of FTP and type in **`ftp://`*`your machine`*** after picking “Other” in the FTP sites menu during the install.

Note: If the boot media (floppy disks, usually) for your FTP clients is not precisely the same version as that provided by the local FTP site, then **sysinstall** will not let you complete the installation. If the versions are not similar and you want to override this, you must go into the Options menu and change distribution name to any.

> **Warning:** This approach is OK for a machine that is on your local network, and that is protected by your firewall. Offering up FTP services to other machines over the Internet (and not your local network) exposes your computer to the attention of crackers and other undesirables. We strongly recommend that you follow good security practices if you do this.

2.13.3 Creating Installation Floppies

If you must install from floppy disk (which we suggest you do *not* do), either due to unsupported hardware or simply because you insist on doing things the hard way, you must first prepare some floppies for the installation.

At a minimum, you will need as many 1.44 MB floppies as it takes to hold all the files in the `base` (base distribution) directory. If you are preparing the floppies from DOS, then they *must* be formatted using the MS-DOS `FORMAT` command. If you are using Windows, use Explorer to format the disks (right-click on the `A:` drive, and select "Format").

Do *not* trust factory pre-formatted floppies. Format them again yourself, just to be sure. Many problems reported by our users in the past have resulted from the use of improperly formatted media, which is why we are making a point of it now.

If you are creating the floppies on another FreeBSD machine, a format is still not a bad idea, though you do not need to put a DOS filesystem on each floppy. You can use the `bsdlabel` and `newfs` commands to put a UFS filesystem on them instead, as the following sequence of commands (for a 3.5" 1.44 MB floppy) illustrates:

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

Then you can mount and write to them like any other filesystem.

After you have formatted the floppies, you will need to copy the files to them. The distribution files are split into chunks conveniently sized so that five of them will fit on a conventional 1.44 MB floppy. Go through all your floppies, packing as many files as will fit on each one, until you have all of the distributions you want packed up in this fashion. Each distribution should go into a subdirectory on the floppy, e.g.: `a:\base\base.aa`, `a:\base\base.ab`, and so on.

> **Important:** The `base.inf` file also needs to go on the first floppy of the `base` set since it is read by the installation program in order to figure out how many additional pieces to look for when fetching and concatenating the distribution.

Once you come to the Media screen during the install process, select Floppy and you will be prompted for the rest.

2.13.4 Installing from an MS-DOS Partition

To prepare for an installation from an MS-DOS partition, copy the files from the distribution into a directory called `freebsd` in the root directory of the partition. For example, `c:\freebsd`. The directory structure of the CDROM or FTP site must be partially reproduced within this directory, so we suggest using the DOS `xcopy` command if you are copying it from a CD. For example, to prepare for a minimal installation of FreeBSD:

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

Assuming that `C:` is where you have free space and `E:` is where your CDROM is mounted.

If you do not have a CDROM drive, you can download the distribution from ftp.FreeBSD.org (ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/8.2-RELEASE/). Each distribution is in its own directory; for example, the *base* distribution can be found in the 8.2/base/ (ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/8.2-RELEASE/base/) directory.

For as many distributions you wish to install from an MS-DOS partition (and you have the free space for), install each one under `c:\freebsd` — the `BIN` distribution is the only one required for a minimum installation.

2.13.5 Creating an Installation Tape

Installing from tape is probably the easiest method, short of an online FTP install or CDROM install. The installation program expects the files to be simply tarred onto the tape. After getting all of the distribution files you are interested in, simply tar them onto the tape:

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

When you perform the installation, you should make sure that you leave enough room in some temporary directory (which you will be allowed to choose) to accommodate the *full* contents of the tape you have created. Due to the non-random access nature of tapes, this method of installation requires quite a bit of temporary storage.

> **Note:** When starting the installation, the tape must be in the drive *before* booting from the boot floppy. The installation probe may otherwise fail to find it.

2.13.6 Before Installing over a Network

There are three types of network installations available. Ethernet (a standard Ethernet controller), Serial port (PPP), or Parallel port (PLIP (laplink cable)).

For the fastest possible network installation, an Ethernet adapter is always a good choice! FreeBSD supports most common PC Ethernet cards; a table of supported cards (and their required settings) is provided in the Hardware Notes for each release of FreeBSD. If you are using one of the supported PCMCIA Ethernet cards, also be sure that it is plugged in *before* the laptop is powered on! FreeBSD does not, unfortunately, currently support hot insertion of PCMCIA cards during installation.

You will also need to know your IP address on the network, the netmask value for your address class, and the name of your machine. If you are installing over a PPP connection and do not have a static IP, fear not, the IP address can be dynamically assigned by your ISP. Your system administrator can tell you which values to use for your particular network setup. If you will be referring to other hosts by name rather than IP address, you will also need a name server and possibly the address of a gateway (if you are using PPP, it is your provider's IP address) to use in talking to it. If you want to install by FTP via a HTTP proxy, you will also need the proxy's address. If you do not know the

answers to all or most of these questions, then you should really probably talk to your system administrator or ISP *before* trying this type of installation.

If you are using a modem, then PPP is almost certainly your only choice. Make sure that you have your service provider's information handy as you will need to know it fairly early in the installation process.

If you use PAP or CHAP to connect your ISP (in other words, if you can connect to the ISP in Windows without using a script), then all you will need to do is type in `dial` at the **ppp** prompt. Otherwise, you will need to know how to dial your ISP using the "AT commands" specific to your modem, as the PPP dialer provides only a very simple terminal emulator. Please refer to the user-ppp handbook and FAQ (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/ppp.html) entries for further information. If you have problems, logging can be directed to the screen using the command `set log local ...`.

If a hard-wired connection to another FreeBSD machine is available, you might also consider installing over a "laplink" parallel port cable. The data rate over the parallel port is much higher than what is typically possible over a serial line (up to 50 kbytes/sec), thus resulting in a quicker installation.

2.13.6.1 Before Installing via NFS

The NFS installation is fairly straight-forward. Simply copy the FreeBSD distribution files you want onto an NFS server and then point the NFS media selection at it.

If this server supports only "privileged port" (as is generally the default for Sun workstations), you will need to set the option `NFS Secure` in the Options menu before installation can proceed.

If you have a poor quality Ethernet card which suffers from very slow transfer rates, you may also wish to toggle the `NFS Slow` flag.

In order for NFS installation to work, the server must support subdir mounts, for example, if your FreeBSD 8.2 distribution directory lives on: `ziggy:/usr/archive/stuff/FreeBSD`, then `ziggy` will have to allow the direct mounting of `/usr/archive/stuff/FreeBSD`, not just `/usr` or `/usr/archive/stuff`.

In FreeBSD's `/etc/exports` file, this is controlled by the `-alldirs` options. Other NFS servers may have different conventions. If you are getting `permission denied` messages from the server, then it is likely that you do not have this enabled properly.

Chapter 3
UNIX Basics

3.1 Synopsis

The following chapter will cover the basic commands and functionality of the FreeBSD operating system. Much of this material is relevant for any UNIX-like operating system. Feel free to skim over this chapter if you are familiar with the material. If you are new to FreeBSD, then you will definitely want to read through this chapter carefully.

After reading this chapter, you will know:

- How to use the “virtual consoles” of FreeBSD.
- How UNIX file permissions work along with understanding file flags in FreeBSD.
- The default FreeBSD file system layout.
- The FreeBSD disk organization.
- How to mount and unmount file systems.
- What processes, daemons, and signals are.
- What a shell is, and how to change your default login environment.
- How to use basic text editors.
- What devices and device nodes are.
- What binary format is used under FreeBSD.
- How to read manual pages for more information.

3.2 Virtual Consoles and Terminals

FreeBSD can be used in various ways. One of them is typing commands to a text terminal. A lot of the flexibility and power of a UNIX operating system is readily available at your hands when using FreeBSD this way. This section describes what “terminals” and “consoles” are, and how you can use them in FreeBSD.

3.2.1 The Console

If you have not configured FreeBSD to automatically start a graphical environment during startup, the system will present you with a login prompt after it boots, right after the startup scripts finish running. You will see something similar to:

```
Additional ABI support:.
Local package initialization:.
Additional TCP options:.
```

```
Fri Sep 20 13:01:06 EEST 2002

FreeBSD/i386 (pc3.example.org) (ttyv0)

login:
```

The messages might be a bit different on your system, but you will see something similar. The last two lines are what we are interested in right now. The second last line reads:

```
FreeBSD/i386 (pc3.example.org) (ttyv0)
```

This line contains some bits of information about the system you have just booted. You are looking at a "FreeBSD" console, running on an Intel or compatible processor of the x86 architecture[1]. The name of this machine (every UNIX machine has a name) is `pc3.example.org`, and you are now looking at its system console—the `ttyv0` terminal.

Finally, the last line is always:

```
login:
```

This is the part where you are supposed to type in your "username" to log into FreeBSD. The next section describes how you can do this.

3.2.2 Logging into FreeBSD

FreeBSD is a multiuser, multiprocessing system. This is the formal description that is usually given to a system that can be used by many different people, who simultaneously run a lot of programs on a single machine.

Every multiuser system needs some way to distinguish one "user" from the rest. In FreeBSD (and all the UNIX-like operating systems), this is accomplished by requiring that every user must "log into" the system before being able to run programs. Every user has a unique name (the "username") and a personal, secret key (the "password"). FreeBSD will ask for these two before allowing a user to run any programs.

Right after FreeBSD boots and finishes running its startup scripts[2], it will present you with a prompt and ask for a valid username:

```
login:
```

For the sake of this example, let us assume that your username is `john`. Type `john` at this prompt and press **Enter**. You should then be presented with a prompt to enter a "password":

```
login: john
Password:
```

Type in `john`'s password now, and press **Enter**. The password is *not echoed!* You need not worry about this right now. Suffice it to say that it is done for security reasons.

If you have typed your password correctly, you should by now be logged into FreeBSD and ready to try out all the available commands.

You should see the MOTD or message of the day followed by a command prompt (a `#`, `$`, or `%` character). This indicates you have successfully logged into FreeBSD.

3.2.3 Multiple Consoles

Running UNIX commands in one console is fine, but FreeBSD can run many programs at once. Having one console where commands can be typed would be a bit of a waste when an operating system like FreeBSD can run dozens of programs at the same time. This is where "virtual consoles" can be very helpful.

FreeBSD can be configured to present you with many different virtual consoles. You can switch from one of them to any other virtual console by pressing a couple of keys on your keyboard. Each console has its own different output channel, and FreeBSD takes care of properly redirecting keyboard input and monitor output as you switch from one virtual console to the next.

Special key combinations have been reserved by FreeBSD for switching consoles[3]. You can use **Alt-F1**, **Alt-F2**, through **Alt-F8** to switch to a different virtual console in FreeBSD.

As you are switching from one console to the next, FreeBSD takes care of saving and restoring the screen output. The result is an "illusion" of having multiple "virtual" screens and keyboards that you can use to type commands for FreeBSD to run. The programs that you launch on one virtual console do not stop running when that console is not visible. They continue running when you have switched to a different virtual console.

3.2.4 The `/etc/ttys` File

The default configuration of FreeBSD will start up with eight virtual consoles. This is not a hardwired setting though, and you can easily customize your installation to boot with more or fewer virtual consoles. The number and settings of the virtual consoles are configured in the `/etc/ttys` file.

You can use the `/etc/ttys` file to configure the virtual consoles of FreeBSD. Each uncommented line in this file (lines that do not start with a # character) contains settings for a single terminal or virtual console. The default version of this file that ships with FreeBSD configures nine virtual consoles, and enables eight of them. They are the lines that start with `ttyv`:

```
# name  getty                           type    status          comments
#
ttyv0   "/usr/libexec/getty Pc"         cons25  on  secure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv2   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv3   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv4   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv5   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv6   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv7   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   off secure
```

For a detailed description of every column in this file and all the options you can use to set things up for the virtual consoles, consult the ttys(5) manual page.

3.2.5 Single User Mode Console

A detailed description of what "single user mode" is can be found in Section 12.6.2. It is worth noting that there is only one console when you are running FreeBSD in single user mode. There are no virtual consoles available. The

settings of the single user mode console can also be found in the `/etc/ttys` file. Look for the line that starts with `console`:

```
# 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 secure
```

> **Note:** As the comments above the `console` line indicate, you can edit this line and change `secure` to `insecure`. If you do that, when FreeBSD boots into single user mode, it will still ask for the `root` password.
>
> *Be careful when changing this to `insecure`.* If you ever forget the `root` password, booting into single user mode is a bit involved. It is still possible, but it might be a bit hard for someone who is not very comfortable with the FreeBSD booting process and the programs involved.

3.2.6 Changing Console Video Modes

The FreeBSD console default video mode may be adjusted to 1024x768, 1280x1024, or any other size supported by your graphics chip and monitor. To use a different video mode, you first must recompile your kernel and include two additional options:

```
options VESA
options SC_PIXEL_MODE
```

Once the kernel has been recompiled with these two options, you can then determine what video modes are supported by your hardware by using the vidcontrol(1) utility. To get a list of supported video modes issue the following:

```
# vidcontrol -i mode
```

The output of this command is a list of video modes that are supported by your hardware. You can then choose to use a new video mode by passing it to vidcontrol(1) in a `root` console:

```
# vidcontrol MODE_279
```

If the new video mode is acceptable, it can be permanently set on boot by setting it in the `/etc/rc.conf` file:

```
allscreens_flags="MODE_279"
```

3.3 Permissions

FreeBSD, being a direct descendant of BSD UNIX, is based on several key UNIX concepts. The first and most pronounced is that FreeBSD is a multi-user operating system. The system can handle several users all working simultaneously on completely unrelated tasks. The system is responsible for properly sharing and managing requests for hardware devices, peripherals, memory, and CPU time fairly to each user.

Because the system is capable of supporting multiple users, everything the system manages has a set of permissions governing who can read, write, and execute the resource. These permissions are stored as three octets broken into three pieces, one for the owner of the file, one for the group that the file belongs to, and one for everyone else. This numerical representation works like this:

Value	Permission	Directory Listing
0	No read, no write, no execute	`---`
1	No read, no write, execute	`--x`
2	No read, write, no execute	`-w-`
3	No read, write, execute	`-wx`
4	Read, no write, no execute	`r--`
5	Read, no write, execute	`r-x`
6	Read, write, no execute	`rw-`
7	Read, write, execute	`rwx`

You can use the `-l` command line argument to ls(1) to view a long directory listing that includes a column with information about a file's permissions for the owner, group, and everyone else. For example, a `ls -l` in an arbitrary directory may show:

```
% ls -l
total 530
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 myfile
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 otherfile
-rw-r--r--  1 root  wheel    7680 Sep  5 12:31 email.txt
...
```

Here is how the first column of `ls -l` is broken up:

```
-rw-r--r--
```

The first (leftmost) character tells if this file is a regular file, a directory, a special character device, a socket, or any other special pseudo-file device. In this case, the `-` indicates a regular file. The next three characters, `rw-` in this example, give the permissions for the owner of the file. The next three characters, `r--`, give the permissions for the group that the file belongs to. The final three characters, `r--`, give the permissions for the rest of the world. A dash means that the permission is turned off. In the case of this file, the permissions are set so the owner can read and write to the file, the group can read the file, and the rest of the world can only read the file. According to the table above, the permissions for this file would be `644`, where each digit represents the three parts of the file's permission.

This is all well and good, but how does the system control permissions on devices? FreeBSD actually treats most hardware devices as a file that programs can open, read, and write data to just like any other file. These special device files are stored on the `/dev` directory.

Directories are also treated as files. They have read, write, and execute permissions. The executable bit for a directory has a slightly different meaning than that of files. When a directory is marked executable, it means it can be traversed into, that is, it is possible to "cd" (change directory) into it. This also means that within the directory it is possible to access files whose names are known (subject, of course, to the permissions on the files themselves).

In particular, in order to perform a directory listing, read permission must be set on the directory, whilst to delete a file that one knows the name of, it is necessary to have write *and* execute permissions to the directory containing the file.

There are more permission bits, but they are primarily used in special circumstances such as setuid binaries and sticky directories. If you want more information on file permissions and how to set them, be sure to look at the chmod(1) manual page.

3.3.1 Symbolic Permissions

Symbolic permissions, sometimes referred to as symbolic expressions, use characters in place of octal values to assign permissions to files or directories. Symbolic expressions use the syntax of (who) (action) (permissions), where the following values are available:

Option	Letter	Represents
(who)	u	User
(who)	g	Group owner
(who)	o	Other
(who)	a	All (“world”)
(action)	+	Adding permissions
(action)	-	Removing permissions
(action)	=	Explicitly set permissions
(permissions)	r	Read
(permissions)	w	Write
(permissions)	x	Execute
(permissions)	t	Sticky bit
(permissions)	s	Set UID or GID

These values are used with the chmod(1) command just like before, but with letters. For an example, you could use the following command to block other users from accessing *`FILE`*:

```
% chmod go= FILE
```

A comma separated list can be provided when more than one set of changes to a file must be made. For example the following command will remove the group and “world” write permission on *`FILE`*, then it adds the execute permissions for everyone:

```
% chmod go-w,a+x FILE
```

3.3.2 FreeBSD File Flags

In addition to file permissions discussed previously, FreeBSD supports the use of “file flags.” These flags add an additional level of security and control over files, but not directories.

These file flags add an additional level of control over files, helping to ensure that in some cases not even the `root` can remove or alter files.

File flags are altered by using the chflags(1) utility, using a simple interface. For example, to enable the system undeletable flag on the file `file1`, issue the following command:

```
# chflags sunlink file1
```

And to disable the system undeletable flag, simply issue the previous command with “no” in front of the `sunlink`. Observe:

```
# chflags nosunlink file1
```

To view the flags of this file, use the ls(1) command with the `-lo` flags:

```
# ls -lo file1
```

The output should look like the following:

```
-rw-r--r--  1 trhodes  trhodes  sunlnk 0 Mar  1 05:54 file1
```

Several flags may only added or removed to files by the `root` user. In other cases, the file owner may set these flags. It is recommended that administrators read over the chflags(1) and chflags(2) manual pages for more information.

3.3.3 The setuid, setgid, and sticky Permissions

Other than the permissions already discussed, there are three other specific settings that all administrators should know about. They are the `setuid`, `setgid` and `sticky` permissions.

These settings are important for some UNIX operations as they provide functionality not normally granted to normal users. To understand them, the difference between the real user ID and effective user ID must also be noted.

The real user ID is the UID who owns or starts the process. The effective UID is the user ID the process runs as. As an example, the passwd(1) utility runs with the real user ID as the user changing their password; however, to manipulate the password database, it runs as the effective ID of the `root` user. This is what allows normal users to change their passwords without seeing a `Permission Denied` error.

> **Note:** The `nosuid` mount(8) option will cause these binaries to silently fail. That is, they will fail to execute without ever alerting the user. That option is also not completely reliable as a `nosuid` wrapper may be able to circumvent it; according to the mount(8) manual page.

The setuid permission may be set by prefixing a permission set with the number four (4) as shown in the following example:

```
# chmod 4755 suidexample.sh
```

The permissions on the *`suidexample.sh`* file should now look like the following:

```
-rwsr-xr-x   1 trhodes  trhodes    63 Aug 29 06:36 suidexample.sh
```

It should be noticeable from this example that an `s` is now part of the permission set designated for the file owner, replacing the executable bit. This allows utilities which need elevated permissions, such as `passwd`.

To view this in real time, open two terminals. On one, start the `passwd` process as a normal user. While it waits for a new password, check the process table and look at the user information of the `passwd` command.

In terminal A:

```
Changing local password for trhodes
Old Password:
```

In terminal B:

```
# 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
```

As stated above, the `passwd` is run by a normal user, but is using the effective UID of `root`.

The `setgid` permission performs the same function as the `setuid` permission; except that it alters the group settings. When an application or utility is ran with this setting, it will be granted the permissions based on the group that owns the file, not the user who started the process.

To set the `setgid` permission on a file, provide the `chmod` command with a leading two (2) as in the following example:

```
# chmod 2755 sgidexample.sh
```

The new setting may be viewed as before, notice the `s` is now in the field designated for the group permission settings:

```
-rwxr-sr-x   1 trhodes  trhodes    44 Aug 31 01:49 sgidexample.sh
```

> **Note:** In these examples, even though the shell script in question is an executable file, it will not run with a different EUID or effective user ID. This is because shell scripts may not access the setuid(2) system calls.

The first two special permission bits we discussed (the `setuid` and `setgid` permission bits) may lower system security, by allowing for elevated permissions. There is a third special permission bit that can strengthen the security of a system: the `sticky bit`.

The `sticky bit`, when set on a directory, allows file deletion only by the file owner. This permission set is useful to prevent file deletion in public directories, such as `/tmp`, by users who do not own the file. To utilize this permission, prefix the permission with a one (1). For example:

```
# chmod 1777 /tmp
```

Now, it is possible to see the effect by using the `ls` command:

```
# ls -al / | grep tmp

drwxrwxrwt  10 root  wheel         512 Aug 31 01:49 tmp
```

The `sticky bit` permission is distinguishable from the `t` at the very end of the set.

3.4 Directory Structure

The FreeBSD directory hierarchy is fundamental to obtaining an overall understanding of the system. The most important concept to grasp is that of the root directory, “/”. This directory is the first one mounted at boot time and it

contains the base system necessary to prepare the operating system for multi-user operation. The root directory also contains mount points for other file systems that are mounted during the transition to multi-user operation.

A mount point is a directory where additional file systems can be grafted onto a parent file system (usually the root file system). This is further described in Section 3.5. Standard mount points include `/usr`, `/var`, `/tmp`, `/mnt`, and `/cdrom`. These directories are usually referenced to entries in the file `/etc/fstab`. `/etc/fstab` is a table of various file systems and mount points for reference by the system. Most of the file systems in `/etc/fstab` are mounted automatically at boot time from the script rc(8) unless they contain the `noauto` option. Details can be found in Section 3.6.1.

A complete description of the file system hierarchy is available in hier(7). For now, a brief overview of the most common directories will suffice.

Directory	Description
`/`	Root directory of the file system.
`/bin/`	User utilities fundamental to both single-user and multi-user environments.
`/boot/`	Programs and configuration files used during operating system bootstrap.
`/boot/defaults/`	Default bootstrapping configuration files; see loader.conf(5).
`/dev/`	Device nodes; see intro(4).
`/etc/`	System configuration files and scripts.
`/etc/defaults/`	Default system configuration files; see rc(8).
`/etc/mail/`	Configuration files for mail transport agents such as sendmail(8).
`/etc/namedb/`	`named` configuration files; see named(8).
`/etc/periodic/`	Scripts that are run daily, weekly, and monthly, via cron(8); see periodic(8).
`/etc/ppp/`	`ppp` configuration files; see ppp(8).
`/mnt/`	Empty directory commonly used by system administrators as a temporary mount point.
`/proc/`	Process file system; see procfs(5), mount_procfs(8).
`/rescue/`	Statically linked programs for emergency recovery; see rescue(8).
`/root/`	Home directory for the `root` account.
`/sbin/`	System programs and administration utilities fundamental to both single-user and multi-user environments.
`/tmp/`	Temporary files. The contents of `/tmp` are usually NOT preserved across a system reboot. A memory-based file system is often mounted at `/tmp`. This can be automated using the tmpmfs-related variables of rc.conf(5) (or with an entry in `/etc/fstab`; see mdmfs(8)).
`/usr/`	The majority of user utilities and applications.
`/usr/bin/`	Common utilities, programming tools, and applications.

Directory	Description
/usr/include/	Standard C include files.
/usr/lib/	Archive libraries.
/usr/libdata/	Miscellaneous utility data files.
/usr/libexec/	System daemons & system utilities (executed by other programs).
/usr/local/	Local executables, libraries, etc. Also used as the default destination for the FreeBSD ports framework. Within /usr/local, the general layout sketched out by hier(7) for /usr should be used. Exceptions are the man directory, which is directly under /usr/local rather than under /usr/local/share, and the ports documentation is in share/doc/*port*.
/usr/obj/	Architecture-specific target tree produced by building the /usr/src tree.
/usr/ports/	The FreeBSD Ports Collection (optional).
/usr/sbin/	System daemons & system utilities (executed by users).
/usr/share/	Architecture-independent files.
/usr/src/	BSD and/or local source files.
/usr/X11R6/	X11R6 distribution executables, libraries, etc (optional).
/var/	Multi-purpose log, temporary, transient, and spool files. A memory-based file system is sometimes mounted at /var. This can be automated using the varmfs-related variables of rc.conf(5) (or with an entry in /etc/fstab; see mdmfs(8)).
/var/log/	Miscellaneous system log files.
/var/mail/	User mailbox files.
/var/spool/	Miscellaneous printer and mail system spooling directories.
/var/tmp/	Temporary files. The files are usually preserved across a system reboot, unless /var is a memory-based file system.
/var/yp/	NIS maps.

3.5 Disk Organization

The smallest unit of organization that FreeBSD uses to find files is the filename. Filenames are case-sensitive, which means that readme.txt and README.TXT are two separate files. FreeBSD does not use the extension (.txt) of a file to determine whether the file is a program, or a document, or some other form of data.

Files are stored in directories. A directory may contain no files, or it may contain many hundreds of files. A directory can also contain other directories, allowing you to build up a hierarchy of directories within one another. This makes it much easier to organize your data.

Files and directories are referenced by giving the file or directory name, followed by a forward slash, `/`, followed by any other directory names that are necessary. If you have directory `foo`, which contains directory `bar`, which contains the file `readme.txt`, then the full name, or *path* to the file is `foo/bar/readme.txt`.

Directories and files are stored in a file system. Each file system contains exactly one directory at the very top level, called the *root directory* for that file system. This root directory can then contain other directories.

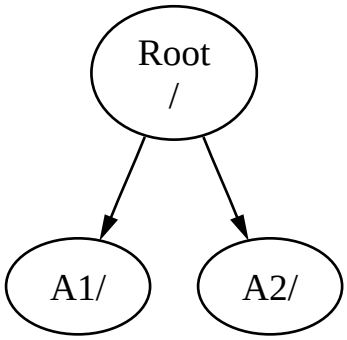
So far this is probably similar to any other operating system you may have used. There are a few differences; for example, MS-DOS uses \ to separate file and directory names, while Mac OS® uses `:`.

FreeBSD does not use drive letters, or other drive names in the path. You would not write `c:/foo/bar/readme.txt` on FreeBSD.

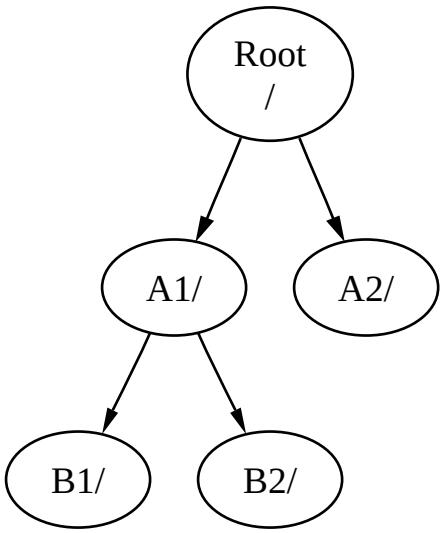
Instead, one file system is designated the *root file system*. The root file system's root directory is referred to as `/`. Every other file system is then *mounted* under the root file system. No matter how many disks you have on your FreeBSD system, every directory appears to be part of the same disk.

Suppose you have three file systems, called `A`, `B`, and `C`. Each file system has one root directory, which contains two other directories, called `A1`, `A2` (and likewise `B1`, `B2` and `C1`, `C2`).

Call `A` the root file system. If you used the `ls` command to view the contents of this directory you would see two subdirectories, `A1` and `A2`. The directory tree looks like this:

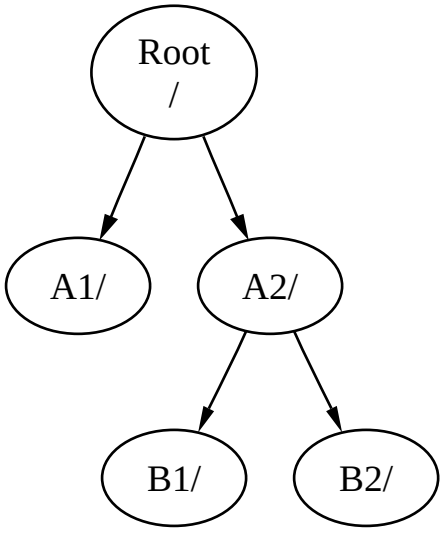


A file system must be mounted on to a directory in another file system. So now suppose that you mount file system `B` on to the directory `A1`. The root directory of `B` replaces `A1`, and the directories in `B` appear accordingly:



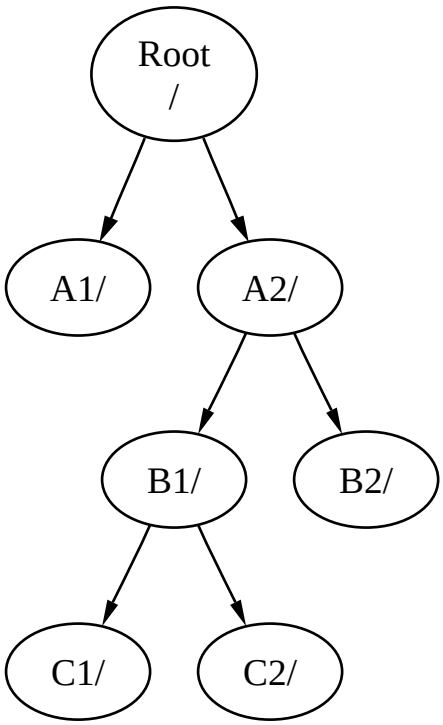
Any files that are in the `B1` or `B2` directories can be reached with the path `/A1/B1` or `/A1/B2` as necessary. Any files that were in `/A1` have been temporarily hidden. They will reappear if `B` is *unmounted* from A.

If `B` had been mounted on `A2` then the diagram would look like this:

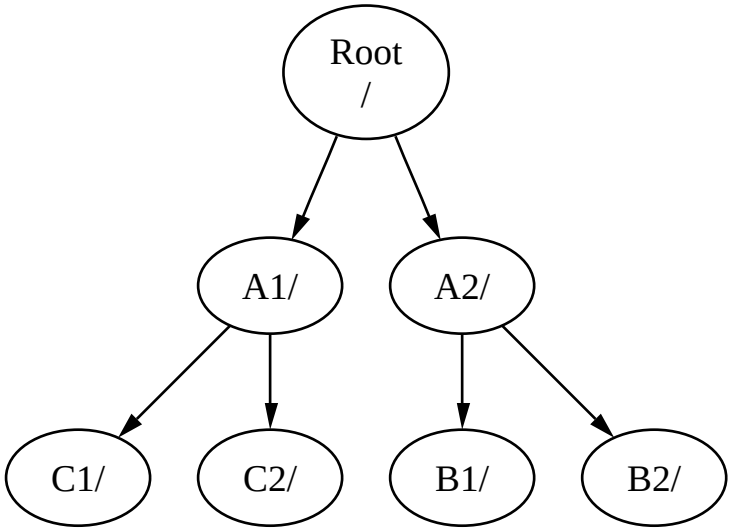


and the paths would be `/A2/B1` and `/A2/B2` respectively.

File systems can be mounted on top of one another. Continuing the last example, the `C` file system could be mounted on top of the `B1` directory in the `B` file system, leading to this arrangement:



Or `C` could be mounted directly on to the `A` file system, under the `A1` directory:



If you are familiar with MS-DOS, this is similar, although not identical, to the `join` command.

This is not normally something you need to concern yourself with. Typically you create file systems when installing FreeBSD and decide where to mount them, and then never change them unless you add a new disk.

It is entirely possible to have one large root file system, and not need to create any others. There are some drawbacks to this approach, and one advantage.

Benefits of Multiple File Systems

- Different file systems can have different *mount options*. For example, with careful planning, the root file system can be mounted read-only, making it impossible for you to inadvertently delete or edit a critical file. Separating user-writable file systems, such as `/home`, from other file systems also allows them to be mounted *nosuid*; this option prevents the *suid*/*guid* bits on executables stored on the file system from taking effect, possibly improving security.
- FreeBSD automatically optimizes the layout of files on a file system, depending on how the file system is being used. So a file system that contains many small files that are written frequently will have a different optimization to one that contains fewer, larger files. By having one big file system this optimization breaks down.
- FreeBSD's file systems are very robust should you lose power. However, a power loss at a critical point could still damage the structure of the file system. By splitting your data over multiple file systems it is more likely that the system will still come up, making it easier for you to restore from backup as necessary.

Benefit of a Single File System

- File systems are a fixed size. If you create a file system when you install FreeBSD and give it a specific size, you may later discover that you need to make the partition bigger. This is not easily accomplished without backing up, recreating the file system with the new size, and then restoring the backed up data.

> **Important:** FreeBSD features the growfs(8) command, which makes it possible to increase the size of file system on the fly, removing this limitation.

File systems are contained in partitions. This does not have the same meaning as the common usage of the term partition (for example, MS-DOS partition), because of FreeBSD's UNIX heritage. Each partition is identified by a letter from `a` through to `h`. Each partition can contain only one file system, which means that file systems are often described by either their typical mount point in the file system hierarchy, or the letter of the partition they are contained in.

FreeBSD also uses disk space for *swap space*. Swap space provides FreeBSD with *virtual memory*. This allows your computer to behave as though it has much more memory than it actually does. When FreeBSD runs out of memory it moves some of the data that is not currently being used to the swap space, and moves it back in (moving something else out) when it needs it.

Some partitions have certain conventions associated with them.

Partition	Convention
`a`	Normally contains the root file system
`b`	Normally contains swap space
`c`	Normally the same size as the enclosing slice. This allows utilities that need to work on the entire slice (for example, a bad block scanner) to work on the `c` partition. You would not normally create a file system on this partition.
`d`	Partition `d` used to have a special meaning associated with it, although that is now gone and `d` may work as any normal partition.

Each partition-that-contains-a-file-system is stored in what FreeBSD calls a *slice*. Slice is FreeBSD's term for what the common call partitions, and again, this is because of FreeBSD's UNIX background. Slices are numbered, starting

at 1, through to 4.

Slice numbers follow the device name, prefixed with an `s`, starting at 1. So "da0*s1*" is the first slice on the first SCSI drive. There can only be four physical slices on a disk, but you can have logical slices inside physical slices of the appropriate type. These extended slices are numbered starting at 5, so "ad0*s5*" is the first extended slice on the first IDE disk. These devices are used by file systems that expect to occupy a slice.

Slices, "dangerously dedicated" physical drives, and other drives contain *partitions*, which are represented as letters from `a` to `h`. This letter is appended to the device name, so "da0*a*" is the a partition on the first da drive, which is "dangerously dedicated". "ad1s3*e*" is the fifth partition in the third slice of the second IDE disk drive.

Finally, each disk on the system is identified. A disk name starts with a code that indicates the type of disk, and then a number, indicating which disk it is. Unlike slices, disk numbering starts at 0. Common codes that you will see are listed in Table 3-1.

When referring to a partition FreeBSD requires that you also name the slice and disk that contains the partition, and when referring to a slice you must also refer to the disk name. Thus, you refer to a partition by listing the disk name, `s`, the slice number, and then the partition letter. Examples are shown in Example 3-1.

Example 3-2 shows a conceptual model of the disk layout that should help make things clearer.

In order to install FreeBSD you must first configure the disk slices, then create partitions within the slice you will use for FreeBSD, and then create a file system (or swap space) in each partition, and decide where that file system will be mounted.

Table 3-1. Disk Device Codes

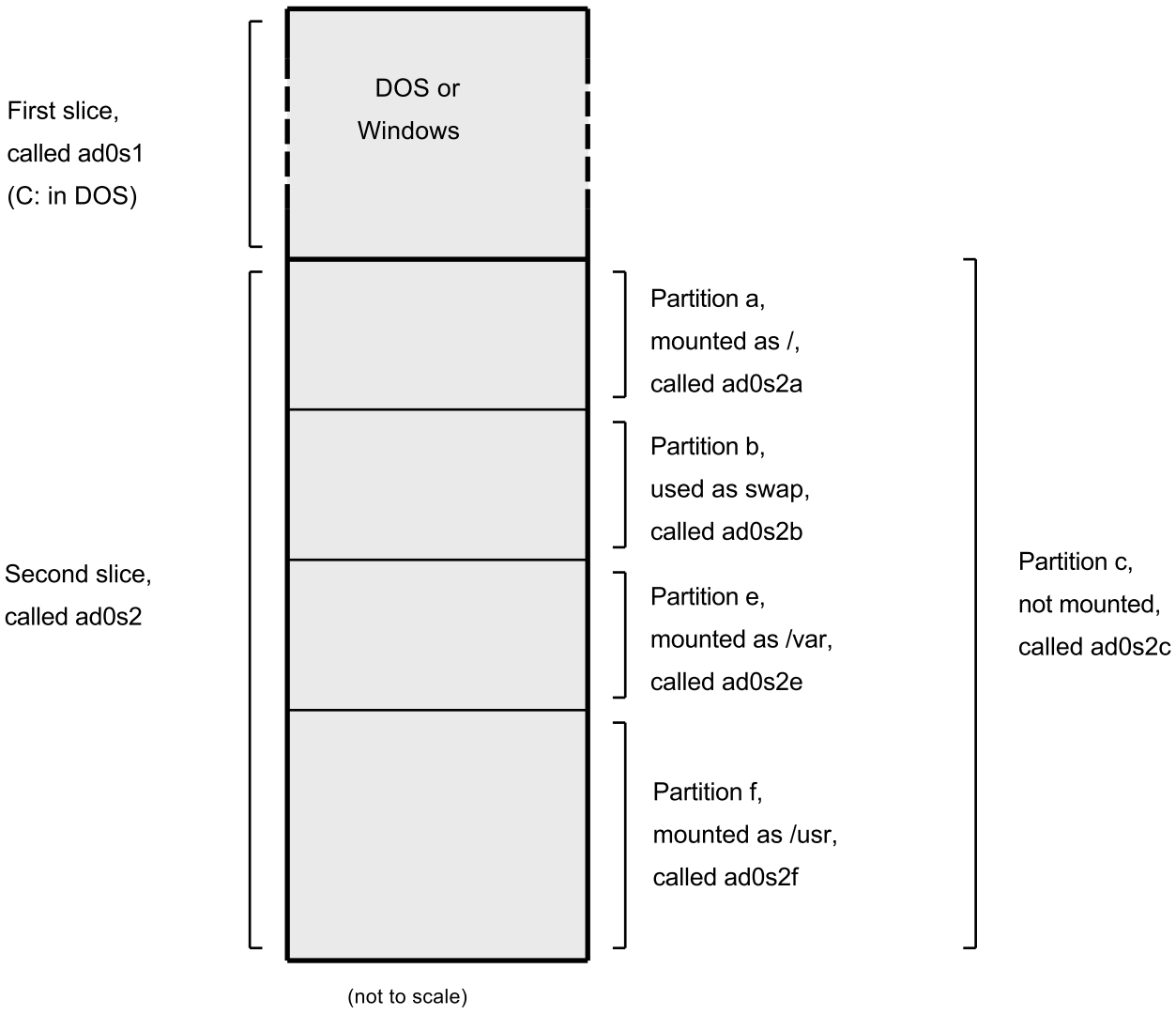
Code	Meaning
`ad`	ATAPI (IDE) disk
`da`	SCSI direct access disk
`acd`	ATAPI (IDE) CDROM
`cd`	SCSI CDROM
`fd`	Floppy disk

Example 3-1. Sample Disk, Slice, and Partition Names

Name	Meaning
`ad0s1a`	The first partition (`a`) on the first slice (`s1`) on the first IDE disk (`ad0`).
`da1s2e`	The fifth partition (`e`) on the second slice (`s2`) on the second SCSI disk (`da1`).

Example 3-2. Conceptual Model of a Disk

This diagram shows FreeBSD's view of the first IDE disk attached to the system. Assume that the disk is 4 GB in size, and contains two 2 GB slices (MS-DOS partitions). The first slice contains a MS-DOS disk, `C:`, and the second slice contains a FreeBSD installation. This example FreeBSD installation has three data partitions, and a swap partition.



3.6 Mounting and Unmounting File Systems

The file system is best visualized as a tree, rooted, as it were, at `/`. `/dev`, `/usr`, and the other directories in the root directory are branches, which may have their own branches, such as `/usr/local`, and so on.

There are various reasons to house some of these directories on separate file systems. `/var` contains the directories `log/`, `spool/`, and various types of temporary files, and as such, may get filled up. Filling up the root file system is not a good idea, so splitting `/var` from `/` is often favorable.

Another common reason to contain certain directory trees on other file systems is if they are to be housed on separate physical disks, or are separate virtual disks, such as Network File System mounts, or CDROM drives.

3.6.1 The `fstab` File

During the boot process, file systems listed in `/etc/fstab` are automatically mounted (unless they are listed with the `noauto` option).

The `/etc/fstab` file contains a list of lines of the following format:

```
device        /mount-point  fstype      options       dumpfreq      passno
```

`device`

A device name (which should exist), as explained in Section 18.2.

`mount-point`

A directory (which should exist), on which to mount the file system.

`fstype`

The file system type to pass to mount(8). The default FreeBSD file system is `ufs`.

`options`

Either `rw` for read-write file systems, or `ro` for read-only file systems, followed by any other options that may be needed. A common option is `noauto` for file systems not normally mounted during the boot sequence. Other options are listed in the mount(8) manual page.

`dumpfreq`

This is used by dump(8) to determine which file systems require dumping. If the field is missing, a value of zero is assumed.

`passno`

This determines the order in which file systems should be checked. File systems that should be skipped should have their `passno` set to zero. The root file system (which needs to be checked before everything else) should have its `passno` set to one, and other file systems' `passno` should be set to values greater than one. If more than one file systems have the same `passno` then fsck(8) will attempt to check file systems in parallel if possible.

Consult the fstab(5) manual page for more information on the format of the `/etc/fstab` file and the options it contains.

3.6.2 The `mount` Command

The mount(8) command is what is ultimately used to mount file systems.

In its most basic form, you use:

```
# mount device mountpoint
```

There are plenty of options, as mentioned in the mount(8) manual page, but the most common are:

Mount Options

`-a`

Mount all the file systems listed in `/etc/fstab`. Except those marked as “noauto”, excluded by the `-t` flag, or those that are already mounted.

`-d`

Do everything except for the actual mount system call. This option is useful in conjunction with the `-v` flag to determine what mount(8) is actually trying to do.

`-f`

Force the mount of an unclean file system (dangerous), or forces the revocation of write access when downgrading a file system’s mount status from read-write to read-only.

`-r`

Mount the file system read-only. This is identical to using the `ro` argument to the `-o` option.

`-t` *`fstype`*

Mount the given file system as the given file system type, or mount only file systems of the given type, if given the `-a` option.

“ufs” is the default file system type.

`-u`

Update mount options on the file system.

`-v`

Be verbose.

`-w`

Mount the file system read-write.

The `-o` option takes a comma-separated list of the options, including the following:

noexec

Do not allow execution of binaries on this file system. This is also a useful security option.

nosuid

Do not interpret setuid or setgid flags on the file system. This is also a useful security option.

3.6.3 The `umount` Command

The umount(8) command takes, as a parameter, one of a mountpoint, a device name, or the `-a` or `-A` option.

All forms take `-f` to force unmounting, and `-v` for verbosity. Be warned that `-f` is not generally a good idea. Forcibly unmounting file systems might crash the computer or damage data on the file system.

`-a` and `-A` are used to unmount all mounted file systems, possibly modified by the file system types listed after `-t`. `-A`, however, does not attempt to unmount the root file system.

3.7 Processes

FreeBSD is a multi-tasking operating system. This means that it seems as though more than one program is running at once. Each program running at any one time is called a *process*. Every command you run will start at least one new process, and there are a number of system processes that run all the time, keeping the system functional.

Each process is uniquely identified by a number called a *process ID*, or *PID*, and, like files, each process also has one owner and group. The owner and group information is used to determine what files and devices the process can open, using the file permissions discussed earlier. Most processes also have a parent process. The parent process is the process that started them. For example, if you are typing commands to the shell then the shell is a process, and any commands you run are also processes. Each process you run in this way will have your shell as its parent process. The exception to this is a special process called init(8). `init` is always the first process, so its PID is always 1. `init` is started automatically by the kernel when FreeBSD starts.

Two commands are particularly useful to see the processes on the system, ps(1) and top(1). The `ps` command is used to show a static list of the currently running processes, and can show their PID, how much memory they are using, the command line they were started with, and so on. The `top` command displays all the running processes, and updates the display every few seconds, so that you can interactively see what your computer is doing.

By default, `ps` only shows you the commands that are running and are owned by you. For example:

```
% ps
  PID  TT  STAT      TIME COMMAND
  298  p0  Ss     0:01.10 tcsh
 7078  p0  S      2:40.88 xemacs mdoc.xsl (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
```

As you can see in this example, the output from ps(1) is organized into a number of columns. `PID` is the process ID discussed earlier. PIDs are assigned starting from 1, go up to 99999, and wrap around back to the beginning when

you run out (a PID is not reassigned if it is already in use). The TT column shows the tty the program is running on, and can safely be ignored for the moment. STAT shows the program's state, and again, can be safely ignored. TIME is the amount of time the program has been running on the CPU—this is usually not the elapsed time since you started the program, as most programs spend a lot of time waiting for things to happen before they need to spend time on the CPU. Finally, COMMAND is the command line that was used to run the program.

ps(1) supports a number of different options to change the information that is displayed. One of the most useful sets is auxww. a displays information about all the running processes, not just your own. u displays the username of the process' owner, as well as memory usage. x displays information about daemon processes, and ww causes ps(1) to display the full command line for each process, rather than truncating it once it gets too long to fit on the screen.

The output from top(1) is similar. A sample session looks like this:

```
% 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
...
```

The output is split into two sections. The header (the first five lines) shows the PID of the last process to run, the system load averages (which are a measure of how busy the system is), the system uptime (time since the last reboot) and the current time. The other figures in the header relate to how many processes are running (47 in this case), how much memory and swap space has been taken up, and how much time the system is spending in different CPU states.

Below that are a series of columns containing similar information to the output from ps(1). As before you can see the PID, the username, the amount of CPU time taken, and the command that was run. top(1) also defaults to showing you the amount of memory space taken by the process. This is split into two columns, one for total size, and one for resident size—total size is how much memory the application has needed, and the resident size is how much it is actually using at the moment. In this example you can see that **Netscape®** has required almost 30 MB of RAM, but is currently only using 9 MB.

top(1) automatically updates this display every two seconds; this can be changed with the s option.

3.8 Daemons, Signals, and Killing Processes

When you run an editor it is easy to control the editor, tell it to load files, and so on. You can do this because the editor provides facilities to do so, and because the editor is attached to a *terminal*. Some programs are not designed to be run with continuous user input, and so they disconnect from the terminal at the first opportunity. For example, a web server spends all day responding to web requests, it normally does not need any input from you. Programs that transport email from site to site are another example of this class of application.

We call these programs *daemons*. Daemons were characters in Greek mythology: neither good or evil, they were little attendant spirits that, by and large, did useful things for mankind, much like the web servers and mail servers of today do useful things. This is why the BSD mascot has, for a long time, been the cheerful-looking daemon with sneakers and a pitchfork.

There is a convention to name programs that normally run as daemons with a trailing "d". **BIND** is the Berkeley Internet Name Domain, but the actual program that executes is called `named`; the **Apache** web server program is called `httpd`; the line printer spooling daemon is `lpd` and so on. This is a convention, not a hard and fast rule; for example, the main mail daemon for the **Sendmail** application is called `sendmail`, and not `maild`, as you might imagine.

Sometimes you will need to communicate with a daemon process. One way to do so is to send it (or any other running process), what is known as a *signal*. There are a number of different signals that you can send—some of them have a specific meaning, others are interpreted by the application, and the application's documentation will tell you how that application interprets signals. You can only send a signal to a process that you own. If you send a signal to someone else's process with kill(1) or kill(2), permission will be denied. The exception to this is the `root` user, who can send signals to everyone's processes.

FreeBSD will also send applications signals in some cases. If an application is badly written, and tries to access memory that it is not supposed to, FreeBSD sends the process the *Segmentation Violation* signal (`SIGSEGV`). If an application has used the alarm(3) system call to be alerted after a period of time has elapsed then it will be sent the Alarm signal (`SIGALRM`), and so on.

Two signals can be used to stop a process, `SIGTERM` and `SIGKILL`. `SIGTERM` is the polite way to kill a process; the process can *catch* the signal, realize that you want it to shut down, close any log files it may have open, and generally finish whatever it is doing at the time before shutting down. In some cases a process may even ignore `SIGTERM` if it is in the middle of some task that can not be interrupted.

`SIGKILL` can not be ignored by a process. This is the "I do not care what you are doing, stop right now" signal. If you send `SIGKILL` to a process then FreeBSD will stop that process there and then[4].

The other signals you might want to use are `SIGHUP`, `SIGUSR1`, and `SIGUSR2`. These are general purpose signals, and different applications will do different things when they are sent.

Suppose that you have changed your web server's configuration file—you would like to tell the web server to re-read its configuration. You could stop and restart `httpd`, but this would result in a brief outage period on your web server, which may be undesirable. Most daemons are written to respond to the `SIGHUP` signal by re-reading their configuration file. So instead of killing and restarting `httpd` you would send it the `SIGHUP` signal. Because there is no standard way to respond to these signals, different daemons will have different behavior, so be sure and read the documentation for the daemon in question.

Signals are sent using the kill(1) command, as this example shows.

Sending a Signal to a Process

This example shows how to send a signal to inetd(8). The `inetd` configuration file is `/etc/inetd.conf`, and `inetd` will re-read this configuration file when it is sent `SIGHUP`.

1. Find the process ID of the process you want to send the signal to. Do this using ps(1) and grep(1). The grep(1) command is used to search through output, looking for the string you specify. This command is run as a normal user, and inetd(8) is run as `root`, so the `ax` options must be given to ps(1).

```
% ps -ax | grep inetd
  198  ??  IWs    0:00.00 inetd -wW
```

So the inetd(8) PID is 198. In some cases the `grep inetd` command might also appear in this output. This is because of the way ps(1) has to find the list of running processes.

2. Use kill(1) to send the signal. Because inetd(8) is being run by `root` you must use su(1) to become `root` first.

```
% su
Password:
# /bin/kill -s HUP 198
```

In common with most UNIX commands, kill(1) will not print any output if it is successful. If you send a signal to a process that you do not own then you will see `kill: PID: Operation not permitted`. If you mistype the PID you will either send the signal to the wrong process, which could be bad, or, if you are lucky, you will have sent the signal to a PID that is not currently in use, and you will see `kill: PID: No such process`.

> **Why Use `/bin/kill`?:** Many shells provide the `kill` command as a built in command; that is, the shell will send the signal directly, rather than running `/bin/kill`. This can be very useful, but different shells have a different syntax for specifying the name of the signal to send. Rather than try to learn all of them, it can be simpler just to use the `/bin/kill ...` command directly.

Sending other signals is very similar, just substitute `TERM` or `KILL` in the command line as necessary.

> **Important:** Killing random process on the system can be a bad idea. In particular, init(8), process ID `1`, is very special. Running `/bin/kill -s KILL 1` is a quick way to shutdown your system. *Always* double check the arguments you run kill(1) with *before* you press **Return**.

3.9 Shells

In FreeBSD, a lot of everyday work is done in a command line interface called a shell. A shell's main job is to take commands from the input channel and execute them. A lot of shells also have built in functions to help with everyday tasks such as file management, file globbing, command line editing, command macros, and environment variables. FreeBSD comes with a set of shells, such as `sh`, the Bourne Shell, and `tcsh`, the improved C-shell. Many other shells are available from the FreeBSD Ports Collection, such as `zsh` and `bash`.

Which shell do you use? It is really a matter of taste. If you are a C programmer you might feel more comfortable with a C-like shell such as `tcsh`. If you have come from Linux or are new to a UNIX command line interface you might try `bash`. The point is that each shell has unique properties that may or may not work with your preferred working environment, and that you have a choice of what shell to use.

One common feature in a shell is filename completion. Given the typing of the first few letters of a command or filename, you can usually have the shell automatically complete the rest of the command or filename by hitting the **Tab** key on the keyboard. Here is an example. Suppose you have two files called `foobar` and `foo.bar`. You want to delete `foo.bar`. So what you would type on the keyboard is: `rm fo[Tab].[Tab]`.

The shell would print out `rm foo[BEEP].bar`.

The [BEEP] is the console bell, which is the shell telling me it was unable to totally complete the filename because there is more than one match. Both `foobar` and `foo.bar` start with `fo`, but it was able to complete to `foo`. If you type in `.`, then hit **Tab** again, the shell would be able to fill in the rest of the filename for you.

Another feature of the shell is the use of environment variables. Environment variables are a variable/key pair stored in the shell's environment space. This space can be read by any program invoked by the shell, and thus contains a lot of program configuration. Here is a list of common environment variables and what they mean:

Variable	Description
`USER`	Current logged in user's name.
`PATH`	Colon-separated list of directories to search for binaries.
`DISPLAY`	Network name of the X11 display to connect to, if available.
`SHELL`	The current shell.
`TERM`	The name of the user's type of terminal. Used to determine the capabilities of the terminal.
`TERMCAP`	Database entry of the terminal escape codes to perform various terminal functions.
`OSTYPE`	Type of operating system. e.g., FreeBSD.
`MACHTYPE`	The CPU architecture that the system is running on.
`EDITOR`	The user's preferred text editor.
`PAGER`	The user's preferred text pager.
`MANPATH`	Colon-separated list of directories to search for manual pages.

Setting an environment variable differs somewhat from shell to shell. For example, in the C-Style shells such as `tcsh` and `csh`, you would use `setenv` to set environment variables. Under Bourne shells such as `sh` and `bash`, you would use `export` to set your current environment variables. For example, to set or modify the `EDITOR` environment variable, under `csh` or `tcsh` a command like this would set `EDITOR` to `/usr/local/bin/emacs`:

```
% setenv EDITOR /usr/local/bin/emacs
```

Under Bourne shells:

```
% export EDITOR="/usr/local/bin/emacs"
```

You can also make most shells expand the environment variable by placing a `$` character in front of it on the command line. For example, `echo $TERM` would print out whatever `$TERM` is set to, because the shell expands `$TERM` and passes it on to `echo`.

Shells treat a lot of special characters, called meta-characters as special representations of data. The most common one is the `*` character, which represents any number of characters in a filename. These special meta-characters can be used to do filename globbing. For example, typing in `echo *` is almost the same as typing in `ls` because the shell takes all the files that match `*` and puts them on the command line for `echo` to see.

To prevent the shell from interpreting these special characters, they can be escaped from the shell by putting a backslash (`\`) character in front of them. `echo $TERM` prints whatever your terminal is set to. `echo \$TERM` prints `$TERM` as is.

3.9.1 Changing Your Shell

The easiest way to change your shell is to use the `chsh` command. Running `chsh` will place you into the editor that is in your `EDITOR` environment variable; if it is not set, you will be placed in `vi`. Change the "Shell:" line accordingly.

You can also give `chsh` the `-s` option; this will set your shell for you, without requiring you to enter an editor. For example, if you wanted to change your shell to `bash`, the following should do the trick:

```
% chsh -s /usr/local/bin/bash
```

> **Note:** The shell that you wish to use *must* be present in the `/etc/shells` file. If you have installed a shell from the ports collection, then this should have been done for you already. If you installed the shell by hand, you must do this.
>
> For example, if you installed `bash` by hand and placed it into `/usr/local/bin`, you would want to:
>
> ```
> # echo "/usr/local/bin/bash" >> /etc/shells
> ```
>
> Then rerun `chsh`.

3.10 Text Editors

A lot of configuration in FreeBSD is done by editing text files. Because of this, it would be a good idea to become familiar with a text editor. FreeBSD comes with a few as part of the base system, and many more are available in the Ports Collection.

The easiest and simplest editor to learn is an editor called **ee**, which stands for easy editor. To start **ee**, one would type at the command line `ee` *`filename`* where *`filename`* is the name of the file to be edited. For example, to edit `/etc/rc.conf`, type in `ee /etc/rc.conf`. Once inside of `ee`, all of the commands for manipulating the editor's functions are listed at the top of the display. The caret `^` character represents the **Ctrl** key on the keyboard, so `^e` expands to the key combination **Ctrl**+**e**. To leave **ee**, hit the **Esc** key, then choose leave editor. The editor will prompt you to save any changes if the file has been modified.

FreeBSD also comes with more powerful text editors such as **vi** as part of the base system, while other editors, like **Emacs** and **vim**, are part of the FreeBSD Ports Collection (`editors/emacs` and `editors/vim`). These editors offer much more functionality and power at the expense of being a little more complicated to learn. However if you plan on doing a lot of text editing, learning a more powerful editor such as **vim** or **Emacs** will save you much more time in the long run.

Many applications which modify files or require typed input will automatically open a text editor. To alter the default editor used, set the `EDITOR` environment variable. See shells section for more details.

3.11 Devices and Device Nodes

A device is a term used mostly for hardware-related activities in a system, including disks, printers, graphics cards, and keyboards. When FreeBSD boots, the majority of what FreeBSD displays are devices being detected. You can look through the boot messages again by viewing `/var/run/dmesg.boot`.

For example, `acd0` is the first IDE CDROM drive, while `kbd0` represents the keyboard.

Most of these devices in a UNIX operating system must be accessed through special files called device nodes, which are located in the `/dev` directory.

3.11.1 Creating Device Nodes

When adding a new device to your system, or compiling in support for additional devices, new device nodes must be created.

3.11.1.1 DEVFS (DEVice File System)

The device file system, or DEVFS, provides access to kernel's device namespace in the global file system namespace. Instead of having to create and modify device nodes, DEVFS maintains this particular file system for you.

See the devfs(5) manual page for more information.

3.12 Binary Formats

To understand why FreeBSD uses the elf(5) format, you must first know a little about the three currently "dominant" executable formats for UNIX:

- a.out(5)

  The oldest and "classic" UNIX object format. It uses a short and compact header with a magic number at the beginning that is often used to characterize the format (see a.out(5) for more details). It contains three loaded segments: .text, .data, and .bss plus a symbol table and a string table.

- COFF

  The SVR3 object format. The header now comprises a section table, so you can have more than just .text, .data, and .bss sections.

- elf(5)

  The successor to COFF, featuring multiple sections and 32-bit or 64-bit possible values. One major drawback: ELF was also designed with the assumption that there would be only one ABI per system architecture. That assumption is actually quite incorrect, and not even in the commercial SYSV world (which has at least three ABIs: SVR4, Solaris, SCO) does it hold true.

  FreeBSD tries to work around this problem somewhat by providing a utility for *branding* a known ELF executable with information about the ABI it is compliant with. See the manual page for brandelf(1) for more information.

FreeBSD comes from the "classic" camp and used the a.out(5) format, a technology tried and proven through many generations of BSD releases, until the beginning of the 3.X branch. Though it was possible to build and run native ELF binaries (and kernels) on a FreeBSD system for some time before that, FreeBSD initially resisted the "push" to switch to ELF as the default format. Why? Well, when the Linux camp made their painful transition to ELF, it was not so much to flee the `a.out` executable format as it was their inflexible jump-table based shared library mechanism, which made the construction of shared libraries very difficult for vendors and developers alike. Since the ELF tools available offered a solution to the shared library problem and were generally seen as "the way forward"

anyway, the migration cost was accepted as necessary and the transition made. FreeBSD's shared library mechanism is based more closely on Sun's SunOS™ style shared library mechanism and, as such, is very easy to use.

So, why are there so many different formats?

Back in the dim, dark past, there was simple hardware. This simple hardware supported a simple, small system. `a.out` was completely adequate for the job of representing binaries on this simple system (a PDP-11). As people ported UNIX from this simple system, they retained the `a.out` format because it was sufficient for the early ports of UNIX to architectures like the Motorola 68k, VAXen, etc.

Then some bright hardware engineer decided that if he could force software to do some sleazy tricks, then he would be able to shave a few gates off the design and allow his CPU core to run faster. While it was made to work with this new kind of hardware (known these days as RISC), `a.out` was ill-suited for this hardware, so many formats were developed to get to a better performance from this hardware than the limited, simple `a.out` format could offer. Things like COFF, ECOFF, and a few obscure others were invented and their limitations explored before things seemed to settle on ELF.

In addition, program sizes were getting huge and disks (and physical memory) were still relatively small so the concept of a shared library was born. The VM system also became more sophisticated. While each one of these advancements was done using the `a.out` format, its usefulness was stretched more and more with each new feature. In addition, people wanted to dynamically load things at run time, or to junk parts of their program after the init code had run to save in core memory and swap space. Languages became more sophisticated and people wanted code called before main automatically. Lots of hacks were done to the `a.out` format to allow all of these things to happen, and they basically worked for a time. In time, `a.out` was not up to handling all these problems without an ever increasing overhead in code and complexity. While ELF solved many of these problems, it would be painful to switch from the system that basically worked. So ELF had to wait until it was more painful to remain with `a.out` than it was to migrate to ELF.

However, as time passed, the build tools that FreeBSD derived their build tools from (the assembler and loader especially) evolved in two parallel trees. The FreeBSD tree added shared libraries and fixed some bugs. The GNU folks that originally wrote these programs rewrote them and added simpler support for building cross compilers, plugging in different formats at will, and so on. Since many people wanted to build cross compilers targeting FreeBSD, they were out of luck since the older sources that FreeBSD had for **as** and **ld** were not up to the task. The new GNU tools chain (**binutils**) does support cross compiling, ELF, shared libraries, C++ extensions, etc. In addition, many vendors are releasing ELF binaries, and it is a good thing for FreeBSD to run them.

ELF is more expressive than `a.out` and allows more extensibility in the base system. The ELF tools are better maintained, and offer cross compilation support, which is important to many people. ELF may be a little slower than `a.out`, but trying to measure it can be difficult. There are also numerous details that are different between the two in how they map pages, handle init code, etc. None of these are very important, but they are differences. In time support for `a.out` will be moved out of the `GENERIC` kernel, and eventually removed from the kernel once the need to run legacy `a.out` programs is past.

3.13 For More Information

3.13.1 Manual Pages

The most comprehensive documentation on FreeBSD is in the form of manual pages. Nearly every program on the system comes with a short reference manual explaining the basic operation and various arguments. These manuals can be viewed with the `man` command. Use of the `man` command is simple:

```
% man command
```

`command` is the name of the command you wish to learn about. For example, to learn more about `ls` command type:

```
% man ls
```

The online manual is divided up into numbered sections:

1. User commands.
2. System calls and error numbers.
3. Functions in the C libraries.
4. Device drivers.
5. File formats.
6. Games and other diversions.
7. Miscellaneous information.
8. System maintenance and operation commands.
9. Kernel developers.

In some cases, the same topic may appear in more than one section of the online manual. For example, there is a `chmod` user command and a `chmod()` system call. In this case, you can tell the `man` command which one you want by specifying the section:

```
% man 1 chmod
```

This will display the manual page for the user command `chmod`. References to a particular section of the online manual are traditionally placed in parenthesis in written documentation, so chmod(1) refers to the `chmod` user command and chmod(2) refers to the system call.

This is fine if you know the name of the command and simply wish to know how to use it, but what if you cannot recall the command name? You can use `man` to search for keywords in the command descriptions by using the `-k` switch:

```
% man -k mail
```

With this command you will be presented with a list of commands that have the keyword “mail” in their descriptions. This is actually functionally equivalent to using the `apropos` command.

So, you are looking at all those fancy commands in `/usr/bin` but do not have the faintest idea what most of them actually do? Simply do:

```
% cd /usr/bin
% man -f *
```

or

```
% cd /usr/bin
% whatis *
```

which does the same thing.

3.13.2 GNU Info Files

FreeBSD includes many applications and utilities produced by the Free Software Foundation (FSF). In addition to manual pages, these programs come with more extensive hypertext documents called `info` files which can be viewed with the `info` command or, if you installed **emacs**, the info mode of **emacs**.

To use the info(1) command, simply type:

```
% info
```

For a brief introduction, type `h`. For a quick command reference, type `?`.

Notes

1. This is what `i386` means. Note that even if you are not running FreeBSD on an Intel 386 CPU, this is going to be `i386`. It is not the type of your processor, but the processor "architecture" that is shown here.
2. Startup scripts are programs that are run automatically by FreeBSD when booting. Their main function is to set things up for everything else to run, and start any services that you have configured to run in the background doing useful things.
3. A fairly technical and accurate description of all the details of the FreeBSD console and keyboard drivers can be found in the manual pages of syscons(4), atkbd(4), vidcontrol(1) and kbdcontrol(1). We will not expand on the details here, but the interested reader can always consult the manual pages for a more detailed and thorough explanation of how things work.
4. Not quite true—there are a few things that can not be interrupted. For example, if the process is trying to read from a file that is on another computer on the network, and the other computer has gone away for some reason (been turned off, or the network has a fault), then the process is said to be "uninterruptible". Eventually the process will time out, typically after two minutes. As soon as this time out occurs the process will be killed.

Chapter 4
Installing Applications: Packages and Ports

4.1 Synopsis

FreeBSD is bundled with a rich collection of system tools as part of the base system. However, there is only so much one can do before needing to install an additional third-party application to get real work done. FreeBSD provides two complementary technologies for installing third-party software on your system: the FreeBSD Ports Collection (for installing from source), and packages (for installing from pre-built binaries). Either method may be used to install the newest version of your favorite applications from local media or straight off the network.

After reading this chapter, you will know:

- How to install third-party binary software packages.
- How to build third-party software from source by using the ports collection.
- How to remove previously installed packages or ports.
- How to override the default values that the ports collection uses.
- How to find the appropriate software package.
- How to upgrade your applications.

4.2 Overview of Software Installation

If you have used a UNIX system before you will know that the typical procedure for installing third-party software goes something like this:

1. Download the software, which might be distributed in source code format, or as a binary.
2. Unpack the software from its distribution format (typically a tarball compressed with compress(1), gzip(1), or bzip2(1)).
3. Locate the documentation (perhaps an `INSTALL` or `README` file, or some files in a `doc/` subdirectory) and read up on how to install the software.
4. If the software was distributed in source format, compile it. This may involve editing a `Makefile`, or running a `configure` script, and other work.
5. Test and install the software.

And that is only if everything goes well. If you are installing a software package that was not deliberately ported to FreeBSD you may even have to go in and edit the code to make it work properly.

Should you want to, you can continue to install software the "traditional" way with FreeBSD. However, FreeBSD provides two technologies which can save you a lot of effort: packages and ports. At the time of writing, over 20,000 third-party applications have been made available in this way.

For any given application, the FreeBSD package for that application is a single file which you must download. The package contains pre-compiled copies of all the commands for the application, as well as any configuration files or documentation. A downloaded package file can be manipulated with FreeBSD package management commands, such as pkg_add(1), pkg_delete(1), pkg_info(1), and so on. Installing a new application can be carried out with a single command.

A FreeBSD port for an application is a collection of files designed to automate the process of compiling an application from source code.

Remember that there are a number of steps you would normally carry out if you compiled a program yourself (downloading, unpacking, patching, compiling, installing). The files that make up a port contain all the necessary information to allow the system to do this for you. You run a handful of simple commands and the source code for the application is automatically downloaded, extracted, patched, compiled, and installed for you.

In fact, the ports system can also be used to generate packages which can later be manipulated with `pkg_add` and the other package management commands that will be introduced shortly.

Both packages and ports understand *dependencies*. Suppose you want to install an application that depends on a specific library being installed. Both the application and the library have been made available as FreeBSD ports and packages. If you use the `pkg_add` command or the ports system to add the application, both will notice that the library has not been installed, and automatically install the library first.

Given that the two technologies are quite similar, you might be wondering why FreeBSD bothers with both. Packages and ports both have their own strengths, and which one you use will depend on your own preference.

Package Benefits

- A compressed package tarball is typically smaller than the compressed tarball containing the source code for the application.
- Packages do not require any additional compilation. For large applications, such as **Mozilla**, **KDE**, or **GNOME** this can be important, particularly if you are on a slow system.
- Packages do not require any understanding of the process involved in compiling software on FreeBSD.

Ports Benefits

- Packages are normally compiled with conservative options, because they have to run on the maximum number of systems. By installing from the port, you can tweak the compilation options to (for example) generate code that is specific to a Pentium 4 or Athlon processor.
- Some applications have compile-time options relating to what they can and cannot do. For example, **Apache** can be configured with a wide variety of different built-in options. By building from the port you do not have to accept the default options, and can set them yourself.

  In some cases, multiple packages will exist for the same application to specify certain settings. For example, **Ghostscript** is available as a `ghostscript` package and a `ghostscript-nox11` package, depending on whether or not you have installed an X11 server. This sort of rough tweaking is possible with packages, but rapidly becomes impossible if an application has more than one or two different compile-time options.

- The licensing conditions of some software distributions forbid binary distribution. They must be distributed as source code.
- Some people do not trust binary distributions. At least with source code, you can (in theory) read through it and look for potential problems yourself.
- If you have local patches, you will need the source in order to apply them.
- Some people like having code around, so they can read it if they get bored, hack it, borrow from it (license permitting, of course), and so on.

To keep track of updated ports, subscribe to the FreeBSD ports mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports) and the FreeBSD ports bugs mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs).

> **Warning:** Before installing any application, you should check http://vuxml.freebsd.org/ for security issues related to your application.
>
> You can also install `ports-mgmt/portaudit` which will automatically check all installed applications for known vulnerabilities; a check will be also performed before any port build. Meanwhile, you can use the command `portaudit -F -a` after you have installed some packages.

The remainder of this chapter will explain how to use packages and ports to install and manage third-party software on FreeBSD.

4.3 Finding Your Application

Before you can install any applications you need to know what you want, and what the application is called.

FreeBSD's list of available applications is growing all the time. Fortunately, there are a number of ways to find what you want:

- The FreeBSD web site maintains an up-to-date searchable list of all the available applications, at http://www.FreeBSD.org/ports/ (http://www.FreeBSD.org/ports/index.html). The ports are divided into categories, and you may either search for an application by name (if you know it), or see all the applications available in a category.
- Dan Langille maintains FreshPorts, at http://www.FreshPorts.org/. FreshPorts tracks changes to the applications in the ports tree as they happen, allows you to "watch" one or more ports, and can send you email when they are updated.
- If you do not know the name of the application you want, try using a site like FreshMeat (http://www.freshmeat.net/) to find an application, then check back at the FreeBSD site to see if the application has been ported yet.
- If you know the exact name of the port, but just need to find out which category it is in, you can use the whereis(1) command. Simply type `whereis` *`file`*, where *`file`* is the program you want to install. If it is found on your system, you will be told where it is, as follows:

  ```
  # whereis lsof
  lsof: /usr/ports/sysutils/lsof
  ```

  This tells us that `lsof` (a system utility) can be found in the `/usr/ports/sysutils/lsof` directory.

- Additionally, you can use a simple echo(1) statement to find where a port exists in the ports tree. For example:

```
# echo /usr/ports/*/*lsof*
/usr/ports/sysutils/lsof
```

  Note that this will return any matched files downloaded into the `/usr/ports/distfiles` directory.

- Yet another way to find a particular port is by using the Ports Collection's built-in search mechanism. To use the search feature, you will need to be in the `/usr/ports` directory. Once in that directory, run `make search name=`*`program-name`* where *`program-name`* is the name of the program you want to find. For example, if you were looking for `lsof`:

```
# 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:
```

  The part of the output you want to pay particular attention to is the "Path:" line, since that tells you where to find the port. The other information provided is not needed in order to install the port, so it will not be covered here.

  For more in-depth searching you can also use `make search key=`*`string`* where *`string`* is some text to search for. This searches port names, comments, descriptions and dependencies and can be used to find ports which relate to a particular subject if you do not know the name of the program you are looking for.

  In both of these cases, the search string is case-insensitive. Searching for "LSOF" will yield the same results as searching for "lsof".

4.4 Using the Packages System

There are several different tools used to manage packages on FreeBSD:

- The `sysinstall` utility can be invoked on a running system to install, delete, and list available and installed packages. For more information, see Section 2.10.11.
- The package management command line tools, which are the subject of the rest of this section.

4.4.1 Installing a Package

You can use the pkg_add(1) utility to install a FreeBSD software package from a local file or from a server on the network.

Example 4-1. Downloading a Package Manually and Installing It Locally

```
# 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
```

If you do not have a source of local packages (such as a FreeBSD CD-ROM set) then it will probably be easier to use the `-r` option to pkg_add(1). This will cause the utility to automatically determine the correct object format and release and then fetch and install the package from an FTP site.

```
# pkg_add -r lsof
```

The example above would download the correct package and add it without any further user intervention. If you want to specify an alternative FreeBSD Packages Mirror, instead of the main distribution site, you have to set the `PACKAGESITE` environment variable accordingly, to override the default settings. pkg_add(1) uses fetch(3) to download the files, which honors various environment variables, including `FTP_PASSIVE_MODE`, `FTP_PROXY`, and `FTP_PASSWORD`. You may need to set one or more of these if you are behind a firewall, or need to use an FTP/HTTP proxy. See fetch(3) for the complete list. Note that in the example above `lsof` is used instead of `lsof-4.56.4`. When the remote fetching feature is used, the version number of the package must be removed. pkg_add(1) will automatically fetch the latest version of the application.

> **Note:** pkg_add(1) will download the latest version of your application if you are using FreeBSD-CURRENT or FreeBSD-STABLE. If you run a -RELEASE version, it will grab the version of the package that was built with your release. It is possible to change this behavior by overriding `PACKAGESITE`. For example, if you run a FreeBSD 8.1-RELEASE system, by default pkg_add(1) will try to fetch packages from `ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-8.1-release/Latest/`. If you want to force pkg_add(1) to download FreeBSD 8-STABLE packages, set `PACKAGESITE` to `ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-8-stable/Latest/`.

Package files are distributed in `.tgz` and `.tbz` formats. You can find them at ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/, or on the FreeBSD CD-ROM distribution. Every CD on the FreeBSD 4-CD set (and the PowerPak, etc.) contains packages in the `/packages` directory. The layout of the packages is similar to that of the `/usr/ports` tree. Each category has its own directory, and every package can be found within the `All` directory.

The directory structure of the package system matches the ports layout; they work with each other to form the entire package/port system.

4.4.2 Managing Packages

pkg_info(1) is a utility that lists and describes the various packages installed.

```
# 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
...
```

pkg_version(1) is a utility that summarizes the versions of all installed packages. It compares the package version to the current version found in the ports tree.

```
# pkg_version
cvsup                       =
docbook                     =
...
```

The symbols in the second column indicate the relative age of the installed version and the version available in the local ports tree.

Symbol	Meaning
=	The version of the installed package matches the one found in the local ports tree.
<	The installed version is older than the one available in the ports tree.
>	The installed version is newer than the one found in the local ports tree. (The local ports tree is probably out of date.)
?	The installed package cannot be found in the ports index. (This can happen, for instance, if an installed port is removed from the Ports Collection or renamed.)
*	There are multiple versions of the package.
!	The installed package exists in the index but for some reason, `pkg_version` was unable to compare the version number of the installed package with the corresponding entry in the index.

4.4.3 Deleting a Package

To remove a previously installed software package, use the pkg_delete(1) utility.

```
# pkg_delete xchat-1.7.1
```

Note that pkg_delete(1) requires the full package name and number; the above command would not work if *`xchat`* was given instead of *`xchat-1.7.1`*. It is, however, easy to use pkg_version(1) to find the version of the installed package. You could instead simply use a wildcard:

```
# pkg_delete xchat\*
```

in this case, all packages whose names start with `xchat` will be deleted.

4.4.4 Miscellaneous

All package information is stored within the `/var/db/pkg` directory. The installed file list and descriptions of each package can be found within files in this directory.

4.5 Using the Ports Collection

The following sections provide basic instructions on using the Ports Collection to install or remove programs from your system. The detailed description of available `make` targets and environment variables is available in ports(7).

4.5.1 Obtaining the Ports Collection

Before you can install ports, you must first obtain the Ports Collection—which is essentially a set of `Makefiles`, patches, and description files placed in `/usr/ports`.

When installing your FreeBSD system, **sysinstall** asked if you would like to install the Ports Collection. If you chose no, you can follow these instructions to obtain the ports collection:

CVSup Method

This is a quick method for getting and keeping your copy of the Ports Collection up to date using **CVSup** protocol. If you want to learn more about **CVSup**, see Using CVSup.

> **Note:** The implementation of **CVSup** protocol included with the FreeBSD system is called **csup**.

Make sure `/usr/ports` is empty before you run **csup** for the first time! If you already have the Ports Collection present, obtained from another source, **csup** will not prune removed patch files.

1. Run `csup`:

   ```
   # csup -L 2 -h cvsup.FreeBSD.org /usr/share/examples/cvsup/ports-supfile
   ```

   Change *`cvsup.FreeBSD.org`* to a **CVSup** server near you. See CVSup Mirrors (Section A.6.7) for a complete listing of mirror sites.

   > **Note:** One may want to use his own `ports-supfile`, for example to avoid the need of passing the **CVSup** server on the command line.

   1. In this case, as `root`, copy `/usr/share/examples/cvsup/ports-supfile` to a new location, such as `/root` or your home directory.
   2. Edit `ports-supfile`.
   3. Change *`CHANGE_THIS.FreeBSD.org`* to a **CVSup** server near you. See CVSup Mirrors (Section A.6.7) for a complete listing of mirror sites.
   4. And now to run `csup`, use the following:

```
# csup -L 2 /root/ports-supfile
```

2. Running the csup(1) command later will download and apply all the recent changes to your Ports Collection, except actually rebuilding the ports for your own system.

Portsnap Method

Portsnap is an alternative system for distributing the Ports Collection. Please refer to Using Portsnap for a detailed description of all **Portsnap** features.

1. Download a compressed snapshot of the Ports Collection into `/var/db/portsnap`. You can disconnect from the Internet after this step, if you wish.

   ```
   # portsnap fetch
   ```

2. If you are running **Portsnap** for the first time, extract the snapshot into `/usr/ports`:

   ```
   # portsnap extract
   ```

   If you already have a populated `/usr/ports` and you are just updating, run the following command instead:

   ```
   # portsnap update
   ```

Sysinstall Method

This method involves using **sysinstall** to install the Ports Collection from the installation media. Note that the old copy of Ports Collection from the date of the release will be installed. If you have Internet access, you should always use one of the methods mentioned above.

1. As `root`, run `sysinstall` as shown below:

   ```
   # sysinstall
   ```

2. Scroll down and select Configure, press **Enter**.
3. Scroll down and select Distributions, press **Enter**.
4. Scroll down to ports, press **Space**.
5. Scroll up to Exit, press **Enter**.
6. Select your desired installation media, such as CDROM, FTP, and so on.
7. Scroll up to Exit and press **Enter**.
8. Press **X** to exit **sysinstall**.

4.5.2 Installing Ports

The first thing that should be explained when it comes to the Ports Collection is what is actually meant by a “skeleton”. In a nutshell, a port skeleton is a minimal set of files that tell your FreeBSD system how to cleanly compile and install a program. Each port skeleton includes:

- A `Makefile`. The `Makefile` contains various statements that specify how the application should be compiled and where it should be installed on your system.
- A `distinfo` file. This file contains information about the files that must be downloaded to build the port, and their checksums (using sha256(1)), to verify that files have not been corrupted during the download.
- A `files` directory. This directory contains patches to make the program compile and install on your FreeBSD system. Patches are basically small files that specify changes to particular files. They are in plain text format, and basically say "Remove line 10" or "Change line 26 to this ...". Patches are also known as "diffs" because they are generated by the diff(1) program.

  This directory may also contain other files used to build the port.
- A `pkg-descr` file. This is a more detailed, often multiple-line, description of the program.
- A `pkg-plist` file. This is a list of all the files that will be installed by the port. It also tells the ports system what files to remove upon deinstallation.

Some ports have other files, such as `pkg-message`. The ports system uses these files to handle special situations. If you want more details on these files, and on ports in general, check out the FreeBSD Porter's Handbook (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/porters-handbook/index.html).

The port includes instructions on how to build source code, but does not include the actual source code. You can get the source code from a CD-ROM or from the Internet. Source code is distributed in whatever manner the software author desires. Frequently this is a tarred and gzipped file, but it might be compressed with some other tool or even uncompressed. The program source code, whatever form it comes in, is called a "distfile". The two methods for installing a FreeBSD port are described below.

> **Note:** You must be logged in as `root` to install ports.

> **Warning:** Before installing any port, you should be sure to have an up-to-date Ports Collection and you should check http://vuxml.freebsd.org/ for security issues related to your port.
>
> A security vulnerabilities check can be automatically done by **portaudit** before any new application installation. This tool can be found in the Ports Collection (`ports-mgmt/portaudit`). Consider running `portaudit -F` before installing a new port, to fetch the current vulnerabilities database. A security audit and an update of the database will be performed during the daily security system check. For more information read the portaudit(1) and periodic(8) manual pages.

The Ports Collection makes an assumption that you have a working Internet connection. If you do not, you will need to put a copy of the distfile into `/usr/ports/distfiles` manually.

To begin, change to the directory for the port you want to install:

```
# cd /usr/ports/sysutils/lsof
```

Once inside the `lsof` directory, you will see the port skeleton. The next step is to compile, or "build", the port. This is done by simply typing `make` at the prompt. Once you have done so, you should see something like this:

```
# 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
...
[extraction output snipped]
...
>> 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
...
[configure output snipped]
...
===>  Building for lsof-4.57
...
[compilation output snipped]
...
#
```

Notice that once the compile is complete you are returned to your prompt. The next step is to install the port. In order to install it, you simply need to tack one word onto the `make` command, and that word is `install`:

```
# make install
===>  Installing for lsof-4.57
...
[installation output snipped]
...
===>   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.
#
```

Once you are returned to your prompt, you should be able to run the application you just installed. Since `lsof` is a program that runs with increased privileges, a security warning is shown. During the building and installation of ports, you should take heed of any other warnings that may appear.

It is a good idea to delete the working subdirectory, which contains all the temporary files used during compilation. Not only does it consume valuable disk space, but it would also cause problems later when upgrading to the newer version of the port.

```
# make clean
===>  Cleaning for lsof-4.57
#
```

> **Note:** You can save two extra steps by just running `make install clean` instead of `make`, `make install` and `make clean` as three separate steps.

> **Note:** Some shells keep a cache of the commands that are available in the directories listed in the `PATH` environment variable, to speed up lookup operations for the executable file of these commands. If you are using

> one of these shells, you might have to use the `rehash` command after installing a port, before the newly installed commands can be used. This command will work for shells like `tcsh`. Use the `hash -r` command for shells like `sh`. Look at the documentation for your shell for more information.

Some third-party DVD-ROM products such as the FreeBSD Toolkit from the FreeBSD Mall (http://www.freebsdmall.com/) contain distfiles. They can be used with the Ports Collection. Mount the DVD-ROM on `/cdrom`. If you use a different mount point, set `CD_MOUNTPTS` make variable. The needed distfiles will be automatically used if they are present on the disk.

> **Note:** Please be aware that the licenses of a few ports do not allow for inclusion on the CD-ROM. This could be because a registration form needs to be filled out before downloading or redistribution is not allowed, or for another reason. If you wish to install a port not included on the CD-ROM, you will need to be online in order to do so.

The ports system uses fetch(1) to download the files, which honors various environment variables, including `FTP_PASSIVE_MODE`, `FTP_PROXY`, and `FTP_PASSWORD`. You may need to set one or more of these if you are behind a firewall, or need to use an FTP/HTTP proxy. See fetch(3) for the complete list.

For users which cannot be connected all the time, the `make fetch` option is provided. Just run this command at the top level directory (`/usr/ports`) and the required files will be downloaded for you. This command will also work in the lower level categories, for example: `/usr/ports/net`. Note that if a port depends on libraries or other ports this will *not* fetch the distfiles of those ports too. Replace `fetch` with `fetch-recursive` if you want to fetch all the dependencies of a port too.

> **Note:** You can build all the ports in a category or as a whole by running `make` in the top level directory, just like the aforementioned `make fetch` method. This is dangerous, however, as some ports cannot co-exist. In other cases, some ports can install two different files with the same filename.

In some rare cases, users may need to acquire the tarballs from a site other than the `MASTER_SITES` (the location where files are downloaded from). You can override the `MASTER_SITES` option with the following command:

```
# cd /usr/ports/directory
# make MASTER_SITE_OVERRIDE= \
ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/ fetch
```

In this example we change the `MASTER_SITES` option to `ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/`.

> **Note:** Some ports allow (or even require) you to provide build options which can enable/disable parts of the application which are unneeded, certain security options, and other customizations. A few which come to mind are `www/mozilla`, `security/gpgme`, and `mail/sylpheed-claws`. A message will be displayed when options such as these are available.

4.5.2.1 Overriding the Default Ports Directories

Sometimes it is useful (or mandatory) to use a different working and target directory. The `WRKDIRPREFIX` and `PREFIX` variables can override the default directories. For example:

```
# make WRKDIRPREFIX=/usr/home/example/ports install
```

will compile the port in `/usr/home/example/ports` and install everything under `/usr/local`.

```
# make PREFIX=/usr/home/example/local install
```

will compile it in `/usr/ports` and install it in `/usr/home/example/local`.

And of course,

```
# make WRKDIRPREFIX=../ports PREFIX=../local install
```

will combine the two (it is too long to completely write on this page, but it should give you the general idea).

Alternatively, these variables can also be set as part of your environment. Read the manual page for your shell for instructions on doing so.

4.5.2.2 Dealing with `imake`

Some ports that use `imake` (a part of the X Window System) do not work well with `PREFIX`, and will insist on installing under `/usr/X11R6`. Similarly, some Perl ports ignore `PREFIX` and install in the Perl tree. Making these ports respect `PREFIX` is a difficult or impossible job.

4.5.2.3 Reconfiguring Ports

When building certain ports, you may be presented with a ncurses-based menu from which you can select certain build options. It is not uncommon for users to wish to revisit this menu to add, remove, or change these options after a port has been built. There are many ways to do this. One option is to go into the directory containing the port and type `make config`, which will simply present the menu again with the same options selected. Another option is to use `make showconfig`, which will show you all the configuration options for the port. Yet another option is to execute `make rmconfig` which will remove all selected options and allow you to start over. All of these options, and others, are explained in great detail in the manual page for ports(7).

4.5.3 Removing Installed Ports

Now that you know how to install ports, you are probably wondering how to remove them, just in case you install one and later on decide that you installed the wrong port. We will remove our previous example (which was `lsof` for those of you not paying attention). Ports are being removed exactly the same as the packages (discussed in the Packages section), using the pkg_delete(1) command:

```
# pkg_delete lsof-4.57
```

4.5.4 Upgrading Ports

First, list outdated ports that have a newer version available in the Ports Collection with the pkg_version(1) command:

```
# pkg_version -v
```

4.5.4.1 /usr/ports/UPDATING

Once you have updated your Ports Collection, before attempting a port upgrade, you should check `/usr/ports/UPDATING`. This file describes various issues and additional steps users may encounter and need to perform when updating a port, including such things as file format changes, changes in locations of configuration files, or other such incompatibilities with previous versions.

If `UPDATING` contradicts something you read here, `UPDATING` takes precedence.

4.5.4.2 Upgrading Ports using Portupgrade

The **portupgrade** utility is designed to easily upgrade installed ports. It is available from the `ports-mgmt/portupgrade` port. Install it like any other port, using the `make install clean` command:

```
# cd /usr/ports/ports-mgmt/portupgrade
# make install clean
```

Scan the list of installed ports with the `pkgdb -F` command and fix all the inconsistencies it reports. It is a good idea to do this regularly, before every upgrade.

When you run `portupgrade -a`, **portupgrade** will begin to upgrade all the outdated ports installed on your system. Use the `-i` flag if you want to be asked for confirmation of every individual upgrade.

```
# portupgrade -ai
```

If you want to upgrade only a certain application, not all available ports, use `portupgrade` *`pkgname`*. Include the `-R` flag if **portupgrade** should first upgrade all the ports required by the given application.

```
# portupgrade -R firefox
```

To use packages instead of ports for installation, provide `-P` flag. With this option **portupgrade** searches the local directories listed in `PKG_PATH`, or fetches packages from remote site if it is not found locally. If packages can not be found locally or fetched remotely, **portupgrade** will use ports. To avoid using ports, specify `-PP`.

```
# portupgrade -PP gnome2
```

To just fetch distfiles (or packages, if `-P` is specified) without building or installing anything, use `-F`. For further information see portupgrade(1).

4.5.4.3 Upgrading Ports using Portmanager

Portmanager is another utility for easy upgrading of installed ports. It is available from the `ports-mgmt/portmanager` port:

```
# cd /usr/ports/ports-mgmt/portmanager
```

```
# make install clean
```

All the installed ports can be upgraded using this simple command:

```
# portmanager -u
```

You can add the `-ui` flag to get asked for confirmation of every step **Portmanager** will perform. **Portmanager** can also be used to install new ports on the system. Unlike the usual `make install clean` command, it will upgrade all the dependencies prior to building and installing the selected port.

```
# portmanager x11/gnome2
```

If there are any problems regarding the dependencies for the selected port, you can use **Portmanager** to rebuild all of them in the correct order. Once finished, the problematic port will be rebuilt too.

```
# portmanager graphics/gimp -f
```

For further information see portmanager(1).

4.5.4.4 Upgrading Ports using Portmaster

Portmaster is another utility for upgrading installed ports. **Portmaster** was designed make use of the tools found in the "base" system (it does not depend upon other ports) and uses the information in `/var/db/pkg/` to determine which ports to upgrade. It is available from the `ports-mgmt/portmaster` port:

```
# cd /usr/ports/ports-mgmt/portmaster
# make install clean
```

Portmaster groups ports into four categories:

- Root ports (no dependencies, not depended on)
- Trunk ports (no dependencies, are depended on)
- Branch ports (have dependencies, are depended on)
- Leaf ports (have dependencies, not depended on)

You can list all the installed ports and search for updates using the `-L` option:

```
# 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
```

All the installed ports can be upgraded using this simple command:

```
# portmaster -a
```

> **Note:** By default, **Portmaster** will make a backup package before deleting the existing port. If the installation of the new version is successful, **Portmaster** will delete the backup. Using the `-b` will instruct **Portmaster** not to automatically delete the backup. Adding the `-i` option will start **Portmaster** in interactive mode, prompting you before upgrading each port.

If you encounter errors during the upgrade process, you can use the `-f` option to upgrade/rebuild all ports:

```
# portmaster -af
```

You can also use **Portmaster** to install new ports on the system, upgrading all dependencies before building and installing the new port:

```
# portmaster shells/bash
```

Please see portmaster(8) for more information.

4.5.5 Ports and Disk Space

Using the Ports Collection will use up disk space over time. After building and installing software from the ports, you should always remember to clean up the temporary `work` directories using the `make clean` command. You can sweep the whole Ports Collection with the following command:

```
# portsclean -C
```

You will accumulate a lot of old source distribution files in the `distfiles` directory over time. You can remove them by hand, or you can use the following command to delete all the distfiles that are no longer referenced by any ports:

```
# portsclean -D
```

Or to remove all distfiles not referenced by any port currently installed on your system:

```
# portsclean -DD
```

> **Note:** The `portsclean` utility is part of the **portupgrade** suite.

Do not forget to remove the installed ports once you no longer need them. A nice tool to help automate this task is available from the `ports-mgmt/pkg_cutleaves` port.

4.6 Post-installation Activities

After installing a new application you will normally want to read any documentation it may have included, edit any configuration files that are required, ensure that the application starts at boot time (if it is a daemon), and so on.

The exact steps you need to take to configure each application will obviously be different. However, if you have just installed a new application and are wondering "What now?" these tips might help:

- Use pkg_info(1) to find out which files were installed, and where. For example, if you have just installed FooPackage version 1.0.0, then this command

  ```
  # pkg_info -L foopackage-1.0.0 | less
  ```

  will show all the files installed by the package. Pay special attention to files in `man/` directories, which will be manual pages, `etc/` directories, which will be configuration files, and `doc/`, which will be more comprehensive documentation.

  If you are not sure which version of the application was just installed, a command like this

  ```
  # pkg_info | grep -i foopackage
  ```

  will find all the installed packages that have *`foopackage`* in the package name. Replace *`foopackage`* in your command line as necessary.
- Once you have identified where the application's manual pages have been installed, review them using man(1). Similarly, look over the sample configuration files, and any additional documentation that may have been provided.
- If the application has a web site, check it for additional documentation, frequently asked questions, and so forth. If you are not sure of the web site address it may be listed in the output from

  ```
  # pkg_info foopackage-1.0.0
  ```

  A `WWW:` line, if present, should provide a URL for the application's web site.
- Ports that should start at boot (such as Internet servers) will usually install a sample script in `/usr/local/etc/rc.d`. You should review this script for correctness and edit or rename it if needed. See Starting Services for more information.

4.7 Dealing with Broken Ports

If you come across a port that does not work for you, there are a few things you can do, including:

1. Find out if there is a fix pending for the port in the Problem Report database (http://www.FreeBSD.org/support.html#gnats). If so, you may be able to use the proposed fix.
2. Ask the maintainer of the port for help. Type `make maintainer` or read the `Makefile` to find the maintainer's email address. Remember to include the name and version of the port (send the `$FreeBSD:` line from the `Makefile`) and the output leading up to the error when you email the maintainer.

> **Note:** Some ports are not maintained by an individual but instead by a mailing list (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/mailing-list-faq/article.html). Many, but not all, of these addresses look like <freebsd-listname@FreeBSD.org>. Please take this into account when phrasing your questions.
>
> In particular, ports shown as maintained by <ports@FreeBSD.org> are actually not maintained by anyone. Fixes and support, if any, come from the general community who subscribe to that mailing list. More volunteers are always needed!

If you do not get a response, you can use send-pr(1) to submit a bug report (see Writing FreeBSD Problem Reports (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/problem-reports/article.html)).

3. Fix it! The Porter's Handbook (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/porters-handbook/index.html) includes detailed information on the "Ports" infrastructure so that you can fix the occasional broken port or even submit your own!
4. Grab the package from an FTP site near you. The "master" package collection is on `ftp.FreeBSD.org` in the packages directory (ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/), but be sure to check your local mirror (http://mirrorlist.FreeBSD.org/) *first*! These are more likely to work than trying to compile from source and are a lot faster as well. Use the pkg_add(1) program to install the package on your system.

Chapter 5
The X Window System

5.1 Synopsis

FreeBSD uses X11 to provide users with a powerful graphical user interface. X11 is a freely available version of the X Window System that is implemented in **Xorg** (and other software packages not discussed here). The default and official flavor of X11 in FreeBSD is **Xorg**, the X11 server developed by the X.Org Foundation under a license very similar to the one used by FreeBSD. Commercial X servers for FreeBSD are also available.

For more information on the video hardware that X11 supports, check the Xorg (http://www.x.org/) web site.

After reading this chapter, you will know:

- The various components of the X Window System, and how they interoperate.
- How to install and configure X11.
- How to install and use different window managers.
- How to use TrueType® fonts in X11.
- How to set up your system for graphical logins (**XDM**).

Before reading this chapter, you should:

- Know how to install additional third-party software (Chapter 4).

5.2 Understanding X

Using X for the first time can be somewhat of a shock to someone familiar with other graphical environments, such as Microsoft Windows or Mac OS.

While it is not necessary to understand all of the details of various X components and how they interact, some basic knowledge makes it possible to take advantage of X's strengths.

5.2.1 Why X?

X is not the first window system written for UNIX, but it is the most popular of them. X's original development team had worked on another window system prior to writing X. That system's name was "W" (for "Window"). X was just the next letter in the Roman alphabet.

X can be called "X", "X Window System", "X11", and a number of other terms. You may find that using the term "X Windows" to describe X11 can be offensive to some people; for a bit more insight on this, see X(7).

5.2.2 The X Client/Server Model

X was designed from the beginning to be network-centric, and adopts a "client-server" model.

In the X model, the "X server" runs on the computer that has the keyboard, monitor, and mouse attached. The server's responsibility includes tasks such as managing the display, handling input from the keyboard and mouse, and other input or output devices (i.e. a "tablet" can be used as an input device, and a video projector may be an alternative output device). Each X application (such as **XTerm**, or **Netscape**) is a "client". A client sends messages to the server such as "Please draw a window at these coordinates", and the server sends back messages such as "The user just clicked on the OK button".

In a home or small office environment, the X server and the X clients commonly run on the same computer. However, it is perfectly possible to run the X server on a less powerful desktop computer, and run X applications (the clients) on, say, the powerful and expensive machine that serves the office. In this scenario the communication between the X client and server takes place over the network.

This confuses some people, because the X terminology is exactly backward to what they expect. They expect the "X server" to be the big powerful machine down the hall, and the "X client" to be the machine on their desk.

It is important to remember that the X server is the machine with the monitor and keyboard, and the X clients are the programs that display the windows.

There is nothing in the protocol that forces the client and server machines to be running the same operating system, or even to be running on the same type of computer. It is certainly possible to run an X server on Microsoft Windows or Apple's Mac OS, and there are various free and commercial applications available that do exactly that.

5.2.3 The Window Manager

The X design philosophy is much like the UNIX design philosophy, "tools, not policy". This means that X does not try to dictate how a task is to be accomplished. Instead, tools are provided to the user, and it is the user's responsibility to decide how to use those tools.

This philosophy extends to X not dictating what windows should look like on screen, how to move them around with the mouse, what keystrokes should be used to move between windows (i.e., **Alt**+**Tab**, in the case of Microsoft Windows), what the title bars on each window should look like, whether or not they have close buttons on them, and so on.

Instead, X delegates this responsibility to an application called a "Window Manager". There are dozens of window managers available for X: **AfterStep**, **Blackbox**, **ctwm**, **Enlightenment**, **fvwm**, **Sawfish**, **twm**, **Window Maker**, and more. Each of these window managers provides a different look and feel; some of them support "virtual desktops"; some of them allow customized keystrokes to manage the desktop; some have a "Start" button or similar device; some are "themeable", allowing a complete change of look-and-feel by applying a new theme. These window managers, and many more, are available in the `x11-wm` category of the Ports Collection.

In addition, the **KDE** and **GNOME** desktop environments both have their own window managers which integrate with the desktop.

Each window manager also has a different configuration mechanism; some expect configuration file written by hand, others feature GUI tools for most of the configuration tasks; at least one (**Sawfish**) has a configuration file written in a dialect of the Lisp language.

Focus Policy: Another feature the window manager is responsible for is the mouse “focus policy”. Every windowing system needs some means of choosing a window to be actively receiving keystrokes, and should visibly indicate which window is active as well.

A familiar focus policy is called “click-to-focus”. This is the model utilized by Microsoft Windows, in which a window becomes active upon receiving a mouse click.

X does not support any particular focus policy. Instead, the window manager controls which window has the focus at any one time. Different window managers will support different focus methods. All of them support click to focus, and the majority of them support several others.

The most popular focus policies are:

focus-follows-mouse

: The window that is under the mouse pointer is the window that has the focus. This may not necessarily be the window that is on top of all the other windows. The focus is changed by pointing at another window, there is no need to click in it as well.

sloppy-focus

: This policy is a small extension to focus-follows-mouse. With focus-follows-mouse, if the mouse is moved over the root window (or background) then no window has the focus, and keystrokes are simply lost. With sloppy-focus, focus is only changed when the cursor enters a new window, and not when exiting the current window.

click-to-focus

: The active window is selected by mouse click. The window may then be “raised”, and appear in front of all other windows. All keystrokes will now be directed to this window, even if the cursor is moved to another window.

Many window managers support other policies, as well as variations on these. Be sure to consult the documentation for the window manager itself.

5.2.4 Widgets

The X approach of providing tools and not policy extends to the widgets seen on screen in each application.

“Widget” is a term for all the items in the user interface that can be clicked or manipulated in some way; buttons, check boxes, radio buttons, icons, lists, and so on. Microsoft Windows calls these “controls”.

Microsoft Windows and Apple’s Mac OS both have a very rigid widget policy. Application developers are supposed to ensure that their applications share a common look and feel. With X, it was not considered sensible to mandate a particular graphical style, or set of widgets to adhere to.

As a result, do not expect X applications to have a common look and feel. There are several popular widget sets and variations, including the original Athena widget set from MIT, **Motif®** (on which the widget set in Microsoft Windows was modeled, all bevelled edges and three shades of grey), **OpenLook**, and others.

Most newer X applications today will use a modern-looking widget set, either Qt, used by **KDE**, or GTK+, used by the **GNOME** project. In this respect, there is some convergence in look-and-feel of the UNIX desktop, which certainly makes things easier for the novice user.

5.3 Installing X11

Xorg is the default X11 implementation for FreeBSD. **Xorg** is the X server of the open source X Window System implementation released by the X.Org Foundation. **Xorg** is based on the code of **XFree86 4.4RC2** and X11R6.6. The version of **Xorg** currently available in the FreeBSD Ports Collection is 7.5.

To build and install **Xorg** from the Ports Collection:

```
# cd /usr/ports/x11/xorg
# make install clean
```

> **Note:** To build **Xorg** in its entirety, be sure to have at least 4 GB of free space available.

Alternatively, X11 can be installed directly from packages. Binary packages to use with pkg_add(1) tool are also available for X11. When the remote fetching feature of pkg_add(1) is used, the version number of the package must be removed. pkg_add(1) will automatically fetch the latest version of the application.

So to fetch and install the package of **Xorg**, simply type:

```
# pkg_add -r xorg
```

> **Note:** The examples above will install the complete X11 distribution including the servers, clients, fonts etc. Separate packages and ports of X11 are also available.
>
> To install a minimal X11 distribution you can alternatively install `x11/xorg-minimal`.

The rest of this chapter will explain how to configure X11, and how to set up a productive desktop environment.

5.4 X11 Configuration

5.4.1 Before Starting

Before configuration of X11 the following information about the target system is needed:

- Monitor specifications
- Video Adapter chipset
- Video Adapter memory

The specifications for the monitor are used by X11 to determine the resolution and refresh rate to run at. These specifications can usually be obtained from the documentation that came with the monitor or from the manufacturer’s website. There are two ranges of numbers that are needed, the horizontal scan rate and the vertical synchronization rate.

The video adapter’s chipset defines what driver module X11 uses to talk to the graphics hardware. With most chipsets, this can be automatically determined, but it is still useful to know in case the automatic detection does not work correctly.

Video memory on the graphic adapter determines the resolution and color depth which the system can run at. This is important to know so the user knows the limitations of the system.

5.4.2 Configuring X11

As of version 7.3, **Xorg** can often work without any configuration file by simply typing at prompt:

```
% startx
```

Starting with version 7.4, **Xorg** can use HAL to autodetect keyboards and mice. The `sysutils/hal` and `devel/dbus` ports are installed as dependencies of `x11/xorg`, but must be enabled by the following entries in the `/etc/rc.conf` file:

```
hald_enable="YES"
dbus_enable="YES"
```

These services should be started (either manually or by rebooting) before further **Xorg** configuration is attempted.

The automatic configuration may fail to work with some hardware, or may not set things up quite as desired. In these cases, manual configuration will be necessary.

> **Note:** Desktop environments like **GNOME**, **KDE** or **Xfce** have tools allowing the user to easily set the screen parameters such as the resolution. So if the default configuration is not acceptable and you planned to install a desktop environment then just continue with the installation of the desktop environment and use the appropriate screen settings tool.

Configuration of X11 is a multi-step process. The first step is to build an initial configuration file. As the super user, simply run:

```
# Xorg -configure
```

This will generate an X11 configuration skeleton file in the `/root` directory called `xorg.conf.new` (whether you su(1) or do a direct login affects the inherited supervisor `$HOME` directory variable). The X11 program will attempt to probe the graphics hardware on the system and write a configuration file to load the proper drivers for the detected hardware on the target system.

The next step is to test the existing configuration to verify that **Xorg** can work with the graphics hardware on the target system. In **Xorg** versions up to 7.3, type:

```
# Xorg -config xorg.conf.new
```

Starting with **Xorg** 7.4 and above, this test produces a black screen which may make it difficult to diagnose whether X11 is working properly. The older behavior is still available by using the `retro` option:

```
# Xorg -config xorg.conf.new -retro
```

If a black and grey grid and an X mouse cursor appear, the configuration was successful. To exit the test, switch to the virtual console used to start it by pressing **Ctrl**+**Alt**+**F*n*** (**F1** for the first virtual console) and press **Ctrl**+**C**.

Note: In **Xorg** versions up to 7.3, the **Ctrl**+**Alt**+**Backspace** key combination could be used to break out of **Xorg**. To enable it in version 7.4 and later, you can either type the following command from any X terminal emulator:

```
% setxkbmap -option terminate:ctrl_alt_bksp
```

or create a keyboard configuration file for **hald** called `x11-input.fdi` and saved in the `/usr/local/etc/hal/fdi/policy` directory. This file should contain the following lines:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<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>
```

You will have to reboot your machine to force **hald** to read this file.

The following line will also have to be added to `xorg.conf.new`, in the `ServerLayout` or `ServerFlags` section:

```
Option "DontZap" "off"
```

If the mouse does not work, you will need to first configure it before proceeding. See Section 2.10.10 in the FreeBSD install chapter. Additionally, starting with version 7.4, the `InputDevice` sections in `xorg.conf` are ignored in favor of the autodetected devices. To restore the old behavior, add the following line to the `ServerLayout` or `ServerFlags` section of this file:

```
Option "AutoAddDevices" "false"
```

Input devices may then be configured as in previous versions, along with any other options needed (e.g. keyboard layout switching).

Note: As previously explained since version 7.4, by default, the **hald** daemon will automatically detect your keyboard. There are chances that your keyboard layout or model will not be correct, desktop environments like **GNOME**, **KDE** or **Xfce** provide tools to configure the keyboard. However, it is possible to set the keyboard properties directly either with the help of the setxkbmap(1) utility or with a **hald**'s configuration rule.

For example if one wants to use a PC 102 keys keyboard coming with a french layout, we have to create a keyboard configuration file for **hald** called `x11-input.fdi` and saved in the `/usr/local/etc/hal/fdi/policy` directory. This file should contain the following lines:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<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>
```

If this file already exists, just copy and add to your file the lines regarding the keyboard configuration.

You will have to reboot your machine to force **hald** to read this file.

It is possible to do the same configuration from an X terminal or a script with this command line:

```
% setxkbmap -model pc102 -layout fr
```

The `/usr/local/share/X11/xkb/rules/base.lst` file lists the various keyboard, layouts and options available.

Next, tune the `xorg.conf.new` configuration file to taste. Open the file in a text editor such as emacs(1) or ee(1). First, add the frequencies for the target system's monitor. These are usually expressed as a horizontal and vertical synchronization rate. These values are added to the `xorg.conf.new` file under the `"Monitor"` section:

```
Section "Monitor"
        Identifier   "Monitor0"
        VendorName   "Monitor Vendor"
        ModelName    "Monitor Model"
        HorizSync    30-107
        VertRefresh  48-120
EndSection
```

The `HorizSync` and `VertRefresh` keywords may be missing in the configuration file. If they are, they need to be added, with the correct horizontal synchronization rate placed after the `HorizSync` keyword and the vertical synchronization rate after the `VertRefresh` keyword. In the example above the target monitor's rates were entered.

X allows DPMS (Energy Star) features to be used with capable monitors. The xset(1) program controls the time-outs and can force standby, suspend, or off modes. If you wish to enable DPMS features for your monitor, you must add the following line to the monitor section:

```
        Option       "DPMS"
```

While the `xorg.conf.new` configuration file is still open in an editor, select the default resolution and color depth desired. This is defined in the `"Screen"` section:

```
Section "Screen"
        Identifier "Screen0"
        Device     "Card0"
        Monitor    "Monitor0"
        DefaultDepth 24
        SubSection "Display"
                Viewport  0 0
                Depth     24
                Modes     "1024x768"
        EndSubSection
EndSection
```

The `DefaultDepth` keyword describes the color depth to run at by default. This can be overridden with the `-depth` command line switch to Xorg(1). The `Modes` keyword describes the resolution to run at for the given color depth. Note that only VESA standard modes are supported as defined by the target system's graphics hardware. In the example above, the default color depth is twenty-four bits per pixel. At this color depth, the accepted resolution is 1024 by 768 pixels.

Finally, write the configuration file and test it using the test mode given above.

> **Note:** One of the tools available to assist you during troubleshooting process are the X11 log files, which contain information on each device that the X11 server attaches to. **Xorg** log file names are in the format of `/var/log/Xorg.0.log`. The exact name of the log can vary from `Xorg.0.log` to `Xorg.8.log` and so forth.

If all is well, the configuration file needs to be installed in a common location where Xorg(1) can find it. This is typically `/etc/X11/xorg.conf` or `/usr/local/etc/X11/xorg.conf`.

```
# cp xorg.conf.new /etc/X11/xorg.conf
```

The X11 configuration process is now complete. **Xorg** may be now started with the startx(1) utility. The X11 server may also be started with the use of xdm(1).

5.4.3 Advanced Configuration Topics

5.4.3.1 Configuration with Intel® i810 Graphics Chipsets

Configuration with Intel i810 integrated chipsets requires the `agpgart` AGP programming interface for X11 to drive the card. See the agp(4) driver manual page for more information.

This will allow configuration of the hardware as any other graphics board. Note on systems without the agp(4) driver compiled in the kernel, trying to load the module with kldload(8) will not work. This driver has to be in the kernel at boot time through being compiled in or using `/boot/loader.conf`.

5.4.3.2 Adding a Widescreen Flatpanel to the Mix

This section assumes a bit of advanced configuration knowledge. If attempts to use the standard configuration tools above have not resulted in a working configuration, there is information enough in the log files to be of use in getting the setup working. Use of a text editor will be necessary.

Current widescreen (WSXGA, WSXGA+, WUXGA, WXGA, WXGA+, et.al.) formats support 16:10 and 10:9 formats or aspect ratios that can be problematic. Examples of some common screen resolutions for 16:10 aspect ratios are:

- 2560x1600
- 1920x1200
- 1680x1050
- 1440x900
- 1280x800

At some point, it will be as easy as adding one of these resolutions as a possible `Mode` in the `Section "Screen"` as such:

```
Section "Screen"
Identifier "Screen0"
Device     "Card0"
Monitor    "Monitor0"
DefaultDepth 24
```

```
SubSection "Display"
 Viewport  0 0
 Depth     24
 Modes     "1680x1050"
EndSubSection
EndSection
```

Xorg is smart enough to pull the resolution information from the widescreen via I2C/DDC information so it knows what the monitor can handle as far as frequencies and resolutions.

If those `ModeLines` do not exist in the drivers, one might need to give **Xorg** a little hint. Using `/var/log/Xorg.0.log` one can extract enough information to manually create a `ModeLine` that will work. Simply look for information resembling this:

```
(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
```

This information is called EDID information. Creating a `ModeLine` from this is just a matter of putting the numbers in the correct order:

```
ModeLine <name> <clock> <4 horiz. timings> <4 vert. timings>
```

So that the `ModeLine` in `Section "Monitor"` for this example would look like this:

```
Section "Monitor"
Identifier      "Monitor1"
VendorName      "Bigname"
ModelName       "BestModel"
ModeLine        "1680x1050" 146.2 1680 1784 1960 2240 1050 1053 1059 1089
Option          "DPMS"
EndSection
```

Now having completed these simple editing steps, X should start on your new widescreen monitor.

5.5 Using Fonts in X11

5.5.1 Type1 Fonts

The default fonts that ship with X11 are less than ideal for typical desktop publishing applications. Large presentation fonts show up jagged and unprofessional looking, and small fonts in **Netscape** are almost completely unintelligible. However, there are several free, high quality Type1 (PostScript®) fonts available which can be readily used with X11. For instance, the URW font collection (`x11-fonts/urwfonts`) includes high quality versions of standard type1 fonts (Times Roman®, Helvetica®, Palatino® and others). The Freefonts collection (`x11-fonts/freefonts`) includes many more fonts, but most of them are intended for use in graphics software such as the **Gimp**, and are not complete enough to serve as screen fonts. In addition, X11 can be configured to use

TrueType fonts with a minimum of effort. For more details on this, see the X(7) manual page or the section on TrueType fonts.

To install the above Type1 font collections from the Ports Collection, run the following commands:

```
# cd /usr/ports/x11-fonts/urwfonts
# make install clean
```

And likewise with the freefont or other collections. To have the X server detect these fonts, add an appropriate line to the X server configuration file (`/etc/X11/xorg.conf`), which reads:

```
FontPath "/usr/local/lib/X11/fonts/URW/"
```

Alternatively, at the command line in the X session run:

```
% xset fp+ /usr/local/lib/X11/fonts/URW
% xset fp rehash
```

This will work but will be lost when the X session is closed, unless it is added to the startup file (`~/.xinitrc` for a normal `startx` session, or `~/.xsession` when logging in through a graphical login manager like **XDM**). A third way is to use the new `/usr/local/etc/fonts/local.conf` file: see the section on anti-aliasing.

5.5.2 TrueType® Fonts

Xorg has built in support for rendering TrueType fonts. There are two different modules that can enable this functionality. The freetype module is used in this example because it is more consistent with the other font rendering back-ends. To enable the freetype module just add the following line to the `"Module"` section of the `/etc/X11/xorg.conf` file.

```
Load  "freetype"
```

Now make a directory for the TrueType fonts (for example, `/usr/local/lib/X11/fonts/TrueType`) and copy all of the TrueType fonts into this directory. Keep in mind that TrueType fonts cannot be directly taken from a Macintosh®; they must be in UNIX/MS-DOS/Windows format for use by X11. Once the files have been copied into this directory, use **ttmkfdir** to create a `fonts.dir` file, so that the X font renderer knows that these new files have been installed. `ttmkfdir` is available from the FreeBSD Ports Collection as `x11-fonts/ttmkfdir`.

```
# cd /usr/local/lib/X11/fonts/TrueType
# ttmkfdir -o fonts.dir
```

Now add the TrueType directory to the font path. This is just the same as described above for Type1 fonts, that is, use

```
% xset fp+ /usr/local/lib/X11/fonts/TrueType
% xset fp rehash
```

or add a `FontPath` line to the `xorg.conf` file.

That's it. Now **Netscape**, **Gimp**, **StarOffice™**, and all of the other X applications should now recognize the installed TrueType fonts. Extremely small fonts (as with text in a high resolution display on a web page) and extremely large fonts (within **StarOffice**) will look much better now.

5.5.3 Anti-Aliased Fonts

All fonts in X11 that are found in `/usr/local/lib/X11/fonts/` and `~/.fonts/` are automatically made available for anti-aliasing to Xft-aware applications. Most recent applications are Xft-aware, including **KDE**, **GNOME**, and **Firefox**.

In order to control which fonts are anti-aliased, or to configure anti-aliasing properties, create (or edit, if it already exists) the file `/usr/local/etc/fonts/local.conf`. Several advanced features of the Xft font system can be tuned using this file; this section describes only some simple possibilities. For more details, please see fonts-conf(5).

This file must be in XML format. Pay careful attention to case, and make sure all tags are properly closed. The file begins with the usual XML header followed by a DOCTYPE definition, and then the <fontconfig> tag:

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<fontconfig>
```

As previously stated, all fonts in `/usr/local/lib/X11/fonts/` as well as `~/.fonts/` are already made available to Xft-aware applications. If you wish to add another directory outside of these two directory trees, add a line similar to the following to `/usr/local/etc/fonts/local.conf`:

```
<dir>/path/to/my/fonts</dir>
```

After adding new fonts, and especially new font directories, you should run the following command to rebuild the font caches:

```
# fc-cache -f
```

Anti-aliasing makes borders slightly fuzzy, which makes very small text more readable and removes “staircases” from large text, but can cause eyestrain if applied to normal text. To exclude font sizes smaller than 14 point from anti-aliasing, include these lines:

```
<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>
```

Spacing for some monospaced fonts may also be inappropriate with anti-aliasing. This seems to be an issue with **KDE**, in particular. One possible fix for this is to force the spacing for such fonts to be 100. Add the following lines:

```
<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>
```

(this aliases the other common names for fixed fonts as "mono"), and then add:

```
<match target="pattern" name="family">
    <test qual="any" name="family">
        <string>mono</string>
    </test>
    <edit name="spacing" mode="assign">
        <int>100</int>
    </edit>
</match>
```

Certain fonts, such as Helvetica, may have a problem when anti-aliased. Usually this manifests itself as a font that seems cut in half vertically. At worst, it may cause applications to crash. To avoid this, consider adding the following to `local.conf`:

```
<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>
```

Once you have finished editing `local.conf` make sure you end the file with the `</fontconfig>` tag. Not doing this will cause your changes to be ignored.

Finally, users can add their own settings via their personal `.fonts.conf` files. To do this, each user should simply create a `~/.fonts.conf`. This file must also be in XML format.

One last point: with an LCD screen, sub-pixel sampling may be desired. This basically treats the (horizontally separated) red, green and blue components separately to improve the horizontal resolution; the results can be dramatic. To enable this, add the line somewhere in the `local.conf` file:

```
<match target="font">
    <test qual="all" name="rgba">
        <const>unknown</const>
    </test>
```

```
        <edit name="rgba" mode="assign">
            <const>rgb</const>
        </edit>
    </match>
```

Note: Depending on the sort of display, `rgb` may need to be changed to `bgr`, `vrgb` or `vbgr`: experiment and see which works best.

5.6 The X Display Manager

5.6.1 Overview

The X Display Manager (**XDM**) is an optional part of the X Window System that is used for login session management. This is useful for several types of situations, including minimal “X Terminals”, desktops, and large network display servers. Since the X Window System is network and protocol independent, there are a wide variety of possible configurations for running X clients and servers on different machines connected by a network. **XDM** provides a graphical interface for choosing which display server to connect to, and entering authorization information such as a login and password combination.

Think of **XDM** as providing the same functionality to the user as the getty(8) utility (see Section 26.3.2 for details). That is, it performs system logins to the display being connected to and then runs a session manager on behalf of the user (usually an X window manager). **XDM** then waits for this program to exit, signaling that the user is done and should be logged out of the display. At this point, **XDM** can display the login and display chooser screens for the next user to login.

5.6.2 Using XDM

To start using **XDM**, install the `x11/xdm` port (it is not installed by default in recent versions of **Xorg**). The **XDM** daemon program may then be found in `/usr/local/bin/xdm`. This program can be run at any time as `root` and it will start managing the X display on the local machine. If **XDM** is to be run every time the machine boots up, a convenient way to do this is by adding an entry to `/etc/ttys`. For more information about the format and usage of this file, see Section 26.3.2.1. There is a line in the default `/etc/ttys` file for running the **XDM** daemon on a virtual terminal:

```
ttyv8   "/usr/local/bin/xdm -nodaemon"  xterm   off secure
```

By default this entry is disabled; in order to enable it change field 5 from `off` to `on` and restart init(8) using the directions in Section 26.3.2.2. The first field, the name of the terminal this program will manage, is `ttyv8`. This means that **XDM** will start running on the 9th virtual terminal.

5.6.3 Configuring XDM

The **XDM** configuration directory is located in `/usr/local/lib/X11/xdm`. In this directory there are several files used to change the behavior and appearance of **XDM**. Typically these files will be found:

File	Description
`Xaccess`	Client authorization ruleset.
`Xresources`	Default X resource values.
`Xservers`	List of remote and local displays to manage.
`Xsession`	Default session script for logins.
`Xsetup_*`	Script to launch applications before the login interface.
`xdm-config`	Global configuration for all displays running on this machine.
`xdm-errors`	Errors generated by the server program.
`xdm-pid`	The process ID of the currently running XDM.

Also in this directory are a few scripts and programs used to set up the desktop when **XDM** is running. The purpose of each of these files will be briefly described. The exact syntax and usage of all of these files is described in xdm(1).

The default configuration is a simple rectangular login window with the hostname of the machine displayed at the top in a large font and “Login:” and “Password:” prompts below. This is a good starting point for changing the look and feel of **XDM** screens.

5.6.3.1 Xaccess

The protocol for connecting to **XDM**-controlled displays is called the X Display Manager Connection Protocol (XDMCP). This file is a ruleset for controlling XDMCP connections from remote machines. It is ignored unless the `xdm-config` is changed to listen for remote connections. By default, it does not allow any clients to connect.

5.6.3.2 Xresources

This is an application-defaults file for the display chooser and login screens. In it, the appearance of the login program can be modified. The format is identical to the app-defaults file described in the X11 documentation.

5.6.3.3 Xservers

This is a list of the remote displays the chooser should provide as choices.

5.6.3.4 Xsession

This is the default session script for **XDM** to run after a user has logged in. Normally each user will have a customized session script in `~/.xsession` that overrides this script.

5.6.3.5 Xsetup_*

These will be run automatically before displaying the chooser or login interfaces. There is a script for each display being used, named `Xsetup_` followed by the local display number (for instance `Xsetup_0`). Typically these scripts will run one or two programs in the background such as `xconsole`.

5.6.3.6 xdm-config

This contains settings in the form of app-defaults that are applicable to every display that this installation manages.

5.6.3.7 xdm-errors

This contains the output of the X servers that **XDM** is trying to run. If a display that **XDM** is trying to start hangs for some reason, this is a good place to look for error messages. These messages are also written to the user's `~/.xsession-errors` file on a per-session basis.

5.6.4 Running a Network Display Server

In order for other clients to connect to the display server, you must edit the access control rules and enable the connection listener. By default these are set to conservative values. To make **XDM** listen for connections, first comment out a line in the `xdm-config` file:

```
! 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
```

and then restart **XDM**. Remember that comments in app-defaults files begin with a "!" character, not the usual "#". More strict access controls may be desired — look at the example entries in `Xaccess`, and refer to the xdm(1) manual page for further information.

5.6.5 Replacements for XDM

Several replacements for the default **XDM** program exist. One of them, **kdm** (bundled with **KDE**) is described later in this chapter. The **kdm** display manager offers many visual improvements and cosmetic frills, as well as the functionality to allow users to choose their window manager of choice at login time.

5.7 Desktop Environments

This section describes the different desktop environments available for X on FreeBSD. A "desktop environment" can mean anything ranging from a simple window manager to a complete suite of desktop applications, such as **KDE** or **GNOME**.

5.7.1 GNOME

5.7.1.1 About GNOME

GNOME is a user-friendly desktop environment that enables users to easily use and configure their computers. **GNOME** includes a panel (for starting applications and displaying status), a desktop (where data and applications can be placed), a set of standard desktop tools and applications, and a set of conventions that make it easy for applications to cooperate and be consistent with each other. Users of other operating systems or environments should feel right at home using the powerful graphics-driven environment that **GNOME** provides. More information regarding **GNOME** on FreeBSD can be found on the FreeBSD GNOME Project (http://www.FreeBSD.org/gnome)'s web site. The web site also contains fairly comprehensive FAQs about installing, configuring, and managing **GNOME**.

5.7.1.2 Installing GNOME

The software can be easily installed from a package or the Ports Collection:

To install the **GNOME** package from the network, simply type:

```
# pkg_add -r gnome2
```

To build **GNOME** from source, use the ports tree:

```
# cd /usr/ports/x11/gnome2
# make install clean
```

Once **GNOME** is installed, the X server must be told to start **GNOME** instead of a default window manager.

The easiest way to start **GNOME** is with **GDM**, the GNOME Display Manager. **GDM**, which is installed as a part of the **GNOME** desktop (but is disabled by default), can be enabled by adding `gdm_enable="YES"` to `/etc/rc.conf`. Once you have rebooted, **GDM** will start automatically.

Additionally, to enable all **GNOME** services when **GDM** starts, add `gnome_enable="YES"` to `/etc/rc.conf`.

GNOME may also be started from the command-line by properly configuring a file named `.xinitrc`. If a custom `.xinitrc` is already in place, simply replace the line that starts the current window manager with one that starts **/usr/local/bin/gnome-session** instead. If nothing special has been done to the configuration file, then it is enough simply to type:

```
% echo "/usr/local/bin/gnome-session" > ~/.xinitrc
```

Next, type `startx`, and the **GNOME** desktop environment will be started.

> **Note:** If an older display manager, like **XDM**, is being used, this will not work. Instead, create an executable `.xsession` file with the same command in it. To do this, edit the file and replace the existing window manager command with **/usr/local/bin/gnome-session**:

```
% echo "#!/bin/sh" > ~/.xsession
% echo "/usr/local/bin/gnome-session" >> ~/.xsession
% chmod +x ~/.xsession
```

Yet another option is to configure the display manager to allow choosing the window manager at login time; the section on KDE details explains how to do this for **kdm**, the display manager of **KDE**.

5.7.2 KDE

5.7.2.1 About KDE

KDE is an easy to use contemporary desktop environment. Some of the things that **KDE** brings to the user are:

- A beautiful contemporary desktop
- A desktop exhibiting complete network transparency
- An integrated help system allowing for convenient, consistent access to help on the use of the **KDE** desktop and its applications
- Consistent look and feel of all **KDE** applications
- Standardized menu and toolbars, keybindings, color-schemes, etc.
- Internationalization: **KDE** is available in more than 40 languages
- Centralized, consistent, dialog-driven desktop configuration
- A great number of useful **KDE** applications

KDE comes with a web browser called **Konqueror**, which is a solid competitor to other existing web browsers on UNIX systems. More information on **KDE** can be found on the KDE website (http://www.kde.org/). For FreeBSD specific information and resources on **KDE**, consult the KDE on FreeBSD team (http://freebsd.kde.org/)'s website.

There are two versions of **KDE** available on FreeBSD. Version 3 has been around for a long time, and is very mature. Version 4, the next generation, is also available in the Ports Collection. They can even be installed side by side.

5.7.2.2 Installing KDE

Just as with **GNOME** or any other desktop environment, the software can be easily installed from a package or the Ports Collection:

To install the **KDE3** package from the network, simply type:

```
# pkg_add -r kde
```

To install the **KDE4** package from the network, simply type:

```
# pkg_add -r kde4
```

pkg_add(1) will automatically fetch the latest version of the application.

To build **KDE3** from source, use the ports tree:

```
# cd /usr/ports/x11/kde3
# make install clean
```

To build **KDE4** from source, use the ports tree:

```
# cd /usr/ports/x11/kde4
# make install clean
```

After **KDE** has been installed, the X server must be told to launch this application instead of the default window manager. This is accomplished by editing the `.xinitrc` file:

For **KDE3**:

```
% echo "exec startkde" > ~/.xinitrc
```

For **KDE4**:

```
% echo "exec /usr/local/kde4/bin/startkde" > ~/.xinitrc
```

Now, whenever the X Window System is invoked with `startx`, **KDE** will be the desktop.

If a display manager such as **XDM** is being used, the configuration is slightly different. Edit the `.xsession` file instead. Instructions for **kdm** are described later in this chapter.

5.7.3 More Details on KDE

Now that **KDE** is installed on the system, most things can be discovered through the help pages, or just by pointing and clicking at various menus. Windows or Mac® users will feel quite at home.

The best reference for **KDE** is the on-line documentation. **KDE** comes with its own web browser, **Konqueror**, dozens of useful applications, and extensive documentation. The remainder of this section discusses the technical items that are difficult to learn by random exploration.

5.7.3.1 The KDE Display Manager

An administrator of a multi-user system may wish to have a graphical login screen to welcome users. XDM can be used, as described earlier. However, **KDE** includes an alternative, **kdm**, which is designed to look more attractive and include more login-time options. In particular, users can easily choose (via a menu) which desktop environment (**KDE**, **GNOME**, or something else) to run after logging on.

To enable **kdm**, different files need to be edited depending on the version of **KDE**.

For **KDE3**, the `ttyv8` entry in `/etc/ttys` has to be adapted as follows:

```
ttyv8 "/usr/local/bin/kdm -nodaemon" xterm on secure
```

For **KDE4**, you have to add the following lines to `/etc/rc.conf`:

```
local_startup="${local_startup} /usr/local/kde4/etc/rc.d"
kdm4_enable="YES"
```

5.7.4 Xfce

5.7.4.1 About Xfce

Xfce is a desktop environment based on the GTK+ toolkit used by **GNOME**, but is much more lightweight and meant for those who want a simple, efficient desktop which is nevertheless easy to use and configure. Visually, it looks very much like **CDE**, found on commercial UNIX systems. Some of **Xfce**'s features are:

- A simple, easy-to-handle desktop
- Fully configurable via mouse, with drag and drop, etc.
- Main panel similar to **CDE**, with menus, applets and applications launchers
- Integrated window manager, file manager, sound manager, **GNOME** compliance module, and more
- Themeable (since it uses GTK+)
- Fast, light and efficient: ideal for older/slower machines or machines with memory limitations

More information on **Xfce** can be found on the Xfce website (http://www.xfce.org/).

5.7.4.2 Installing Xfce

A binary package for **Xfce** exists (at the time of writing). To install, simply type:

```
# pkg_add -r xfce4
```

Alternatively, to build from source, use the Ports Collection:

```
# cd /usr/ports/x11-wm/xfce4
# make install clean
```

Now, tell the X server to launch **Xfce** the next time X is started. Simply type this:

```
% echo "/usr/local/bin/startxfce4" > ~/.xinitrc
```

The next time X is started, **Xfce** will be the desktop. As before, if a display manager like **XDM** is being used, create an `.xsession`, as described in the section on GNOME, but with the `/usr/local/bin/startxfce4` command; or, configure the display manager to allow choosing a desktop at login time, as explained in the section on kdm.

II. Common Tasks

Now that the basics have been covered, this part of the FreeBSD Handbook will discuss some frequently used features of FreeBSD. These chapters:

- Introduce you to popular and useful desktop applications: browsers, productivity tools, document viewers, etc.
- Introduce you to a number of multimedia tools available for FreeBSD.
- Explain the process of building a customized FreeBSD kernel, to enable extra functionality on your system.
- Describe the print system in detail, both for desktop and network-connected printer setups.
- Show you how to run Linux applications on your FreeBSD system.

Some of these chapters recommend that you do some prior reading, and this is noted in the synopsis at the beginning of each chapter.

Chapter 6
Desktop Applications

6.1 Synopsis

FreeBSD can run a wide variety of desktop applications, such as browsers and word processors. Most of these are available as packages or can be automatically built from the ports collection. Many new users expect to find these kinds of applications on their desktop. This chapter will show you how to install some popular desktop applications effortlessly, either from their packages or from the Ports Collection.

Note that when installing programs from the ports, they are compiled from source. This can take a very long time, depending on what you are compiling and the processing power of your machine(s). If building from source takes a prohibitively long amount of time for you, you can install most of the programs of the Ports Collection from pre-built packages.

As FreeBSD features Linux binary compatibility, many applications originally developed for Linux are available for your desktop. It is strongly recommended that you read Chapter 10 before installing any of the Linux applications. Many of the ports using the Linux binary compatibility start with "linux-". Remember this when you search for a particular port, for instance with whereis(1). In the following text, it is assumed that you have enabled Linux binary compatibility before installing any of the Linux applications.

Here are the categories covered by this chapter:

- Browsers (such as **Firefox**, **Opera**, **Konqueror**)
- Productivity (such as **KOffice**, **AbiWord**, **The GIMP**, **OpenOffice.org**)
- Document Viewers (such as **Acrobat Reader®**, **gv**, **Xpdf**, **GQview**)
- Finance (such as **GnuCash**, **Gnumeric**, **Abacus**)

Before reading this chapter, you should:

- Know how to install additional third-party software (Chapter 4).
- Know how to install additional Linux software (Chapter 10).

For information on how to get a multimedia environment, read Chapter 7. If you want to set up and use electronic mail, please refer to Chapter 28.

6.2 Browsers

FreeBSD does not come with a particular browser pre-installed. Instead, the www (http://www.FreeBSD.org/ports/www.html) directory of the Ports Collection contains a lot of browsers ready to be installed. If you do not have time to compile everything (this can take a very long time in some cases) many of them are available as packages.

KDE and **GNOME** already provide HTML browsers. Please refer to Section 5.7 for more information on how to set up these complete desktops.

If you are looking for light-weight browsers, you should investigate the Ports Collection for `www/dillo2`, `www/links`, or `www/w3m`.

This section covers these applications:

Application Name	Resources Needed	Installation from Ports	Major Dependencies
Firefox	medium	heavy	**Gtk+**
Opera	light	light	FreeBSD and Linux versions available. The Linux version depends on the Linux Binary Compatibility and **linux-openmotif**.
Konqueror	medium	heavy	**KDE** Libraries

6.2.1 Firefox

Firefox is a modern, free, open-source stable browser that is fully ported to FreeBSD: it features a very standards-compliant HTML display engine, tabbed browsing, popup blocking, extensions, improved security, and more. **Firefox** is based on the **Mozilla** codebase.

Install the package by typing:

```
# pkg_add -r firefox
```

This will install **Firefox** 3.6, if you want to run **Firefox** 3.5, use instead:

```
# pkg_add -r firefox35
```

You can also use the Ports Collection if you prefer to compile from source code:

```
# cd /usr/ports/www/firefox
# make install clean
```

For **Firefox** 3.5, in the previous command replace `firefox` with `firefox35`.

6.2.2 Firefox and Java™ Plugin

> **Note:** In this section and in the next two sections, we assume you have already installed **Firefox**.
>
> Currently, the Java™ plugin does not work with **Firefox** 3.6.

The FreeBSD Foundation has a license with Sun Microsystems to distribute FreeBSD binaries for the Java Runtime Environment (JRE™) and Java Development Kit (JDK™). Binary packages for FreeBSD are available on the FreeBSD Foundation (http://www.freebsdfoundation.org/downloads/java.shtml) web site.

To add Java support to **Firefox**, you first have to install the `java/javavmwrapper` port. Then, download the **Diablo JRE** package from http://www.freebsdfoundation.org/downloads/java.shtml, and install it with pkg_add(1).

> **Note:** The above site does not provide binary packages for FreeBSD 8.*X*. It is however possible to use the packages for FreeBSD 7.*X* on an 8.*X* system. Simply install the `misc/compat7x` port before installing the package.
>
> Alternatively, **Diablo JRE** (as well as **Diablo JDK**) may be installed using the Ports Collection (the relevant ports are `java/diablo-jre16` and `java/diablo-jdk16`). Installing from the Ports Collection requires the source files (distfiles) to be downloaded manually due to licensing issues. Specific download instructions are provided when the `make install` command is invoked.

Start your browser, enter `about:plugins` in the location bar and press **Enter**. A page listing the installed plugins will be displayed; the **Java** plugin should be listed there now. If it is not, each user will have to run the following command:

```
% ln -s /usr/local/diablo-jre1.6.0/plugin/i386/ns7/libjavaplugin_oji.so \
  $HOME/.mozilla/plugins/
```

or, if you installed the **Diablo JDK** package:

```
% ln -s /usr/local/diablo-jdk1.6.0/jre/plugin/i386/ns7/libjavaplugin_oji.so \
  $HOME/.mozilla/plugins/
```

Then relaunch your browser.

> **Note:** The commands above assume you are running the i386 architecture, amd64 packages are also available.

6.2.3 Firefox and Macromedia® Flash™ Plugin

Macromedia® Flash™ plugin is not available for FreeBSD. However, a software layer (wrapper) for running the Linux version of the plugin exists. This wrapper also supports Adobe® Acrobat® plugin, RealPlayer® plugin and more.

According to the version of FreeBSD you run various steps are required:

1. Under FreeBSD 7.X

   Install the `www/nspluginwrapper` port. This port requires `emulators/linux_base-fc4` which is a large port.

   The next step is to install the `www/linux-flashplugin9` port. This will install Flash 9.X, this version is known to run correctly under FreeBSD 7.X.

   > **Note:** On FreeBSD versions older than FreeBSD 7.1-RELEASE you have to install `www/linux-flashplugin7` and skip the linprocfs(5) part below.

2. Under FreeBSD 8.X

   Install the `www/nspluginwrapper` port. This port requires `emulators/linux_base-f10` which is a large port.

   The next step is to install the `www/linux-f10-flashplugin10` port. This will install Flash 10.X, this version is known to run correctly under FreeBSD 8.X.

   This version will require the following link to be created:

   ```
   # ln -s /usr/local/lib/npapi/linux-f10-flashplugin/libflashplayer.so \
     /usr/local/lib/browser_plugins/
   ```

Once the right Flash port, according to the FreeBSD version you run, is installed, the plugin must be installed by each user with `nspluginwrapper`:

```
% nspluginwrapper -v -a -i
```

The Linux process file system, linprocfs(5) has to be mounted on `/usr/compat/linux/proc`, if one wants to play Flash animations. This can be done via the following command:

```
# mount -t linprocfs linproc /usr/compat/linux/proc
```

This point can be automated at boot time with the addition of the matching line in `/etc/fstab`:

```
linproc /usr/compat/linux/proc linprocfs rw 0 0
```

Then, start your browser, enter `about:plugins` in the location bar and press **Enter**. A list should appear with all the currently available plugins.

6.2.4 Firefox and Swfdec Flash Plugin

Swfdec is the library for decoding and rendering Flash animations. And Swfdec-Mozilla is a plugin for **Firefox** browsers that uses the Swfdec library for playing SWF files. It is still in heavy development.

If you cannot or do not want to compile it, just install the package from the network:

```
# pkg_add -r swfdec-plugin
```

If the package is not available, you can compile and install it from the Ports Collection:

```
# cd /usr/ports/www/swfdec-plugin
# make install clean
```

Then, restart your browser for this plugin taking effect.

6.2.5 Opera

Opera is a full-featured and standards-compliant browser. It also comes with a built-in mail and news reader, an IRC client, an RSS/Atom feeds reader and much more. Despite this, **Opera** is relatively lightweight and very fast. It comes in two flavors: a “native” FreeBSD version and a version that runs under Linux emulation.

To browse the Web with the FreeBSD version of **Opera**, install the package:

```
# pkg_add -r opera
```

Some FTP sites do not have all the packages, but **Opera** can still be obtained through the Ports Collection by typing:

```
# cd /usr/ports/www/opera
# make install clean
```

To install the Linux version of **Opera**, substitute `linux-opera` in place of `opera` in the examples above. The Linux version is useful in situations requiring the use of plug-ins that are only available for Linux, such as **Adobe Acrobat Reader**. In all other respects, the FreeBSD and Linux versions should be functionally identical.

6.2.6 Konqueror

Konqueror is part of **KDE** but it can also be used outside of **KDE** by installing `x11/kdebase3`. **Konqueror** is much more than a browser, it is also a file manager and a multimedia viewer.

There is also a set of plugins available for **Konqueror**, available in `misc/konq-plugins`.

Konqueror also supports **Flash**; a "How To" guide for getting **Flash** support on **Konqueror** is available at http://freebsd.kde.org/howtos/konqueror-flash.php.

6.3 Productivity

When it comes to productivity, new users often look for a good office suite or a friendly word processor. While some desktop environments like **KDE** already provide an office suite, there is no default productivity package. FreeBSD can provide all that is needed, regardless of your desktop environment.

This section covers these applications:

Application Name	Resources Needed	Installation from Ports	Major Dependencies
KOffice	light	heavy	**KDE**
AbiWord	light	light	**Gtk+** or **GNOME**
The Gimp	light	heavy	**Gtk+**
OpenOffice.org	heavy	huge	**JDK**, **Mozilla**

6.3.1 KOffice

The KDE community has provided its desktop environment with an office suite which can be used outside **KDE**. It includes the four standard components that can be found in other office suites. **KWord** is the word processor, **KSpread** is the spreadsheet program, **KPresenter** manages slide presentations, and **Kontour** lets you draw graphical documents.

Before installing the latest **KOffice**, make sure you have an up-to-date version of **KDE**.

To install **KOffice** as a package, issue the following command:

```
# pkg_add -r koffice
```

If the package is not available, you can use the ports collection. For instance, to install **KOffice** for **KDE3**, do:

```
# cd /usr/ports/editors/koffice-kde3
# make install clean
```

6.3.2 AbiWord

AbiWord is a free word processing program similar in look and feel to **Microsoft Word**. It is suitable for typing papers, letters, reports, memos, and so forth. It is very fast, contains many features, and is very user-friendly.

AbiWord can import or export many file formats, including some proprietary ones like Microsoft's `.doc`.

AbiWord is available as a package. You can install it by:

```
# pkg_add -r abiword
```

If the package is not available, it can be compiled from the Ports Collection. The Ports Collection should be more up to date. It can be done as follows:

```
# cd /usr/ports/editors/abiword
# make install clean
```

6.3.3 The GIMP

For image authoring or picture retouching, **The GIMP** is a very sophisticated image manipulation program. It can be used as a simple paint program or as a quality photo retouching suite. It supports a large number of plug-ins and features a scripting interface. **The GIMP** can read and write a wide range of file formats. It supports interfaces with scanners and tablets.

You can install the package by issuing this command:

```
# pkg_add -r gimp
```

If your FTP site does not have this package, you can use the Ports Collection. The graphics (http://www.FreeBSD.org/ports/graphics.html) directory of the Ports Collection also contains **The Gimp Manual**. Here is how to get them installed:

```
# cd /usr/ports/graphics/gimp
# make install clean
# cd /usr/ports/graphics/gimp-manual-pdf
# make install clean
```

> **Note:** The graphics (http://www.FreeBSD.org/ports/graphics.html) directory of the Ports Collection holds the development version of **The GIMP** in `graphics/gimp-devel`. An HTML version of **The Gimp Manual** is available from `graphics/gimp-manual-html`.

6.3.4 OpenOffice.org

OpenOffice.org includes all of the mandatory applications in a complete office productivity suite: a word processor, a spreadsheet, a presentation manager, and a drawing program. Its user interface is very similar to other office suites, and it can import and export in various popular file formats. It is available in a number of different languages — internationalization has been extended to interfaces, spell checkers, and dictionaries.

The word processor of **OpenOffice.org** uses a native XML file format for increased portability and flexibility. The spreadsheet program features a macro language and it can be interfaced with external databases. **OpenOffice.org** is already stable and runs natively on Windows, Solaris™, Linux, FreeBSD, and Mac OS X. More information about **OpenOffice.org** can be found on the OpenOffice.org web site (http://www.openoffice.org/). For FreeBSD specific information, and to directly download packages, use the FreeBSD OpenOffice.org Porting Team (http://porting.openoffice.org/freebsd/)'s web site.

To install **OpenOffice.org**, do:

```
# pkg_add -r openoffice.org
```

> **Note:** When running a -RELEASE version of FreeBSD, this should work. Otherwise, you should look on the FreeBSD **OpenOffice.org** Porting Team's web site to download and install the appropriate package using pkg_add(1). Both the current release and development version are available for download at this location.

Once the package is installed, you just have to type the following command to run **OpenOffice.org**:

```
% openoffice.org
```

> **Note:** During the first launch, you will be asked some questions and a `.openoffice.org` folder will be created in your home directory.

If the **OpenOffice.org** packages are not available, you still have the option to compile the port. However, you must bear in mind that it requires a lot of disk space and a fairly long time to compile.

```
# cd /usr/ports/editors/openoffice.org-3
# make install clean
```

> **Note:** If you want to build a localized version, replace the previous command line with the following:
>
> ```
> # make LOCALIZED_LANG=your_language install clean
> ```
>
> You have to replace *your_language* with the correct language ISO-code. A list of supported language codes is available in the `files/Makefile.localized` file, located in the port directory.

Once this is done, **OpenOffice.org** can be launched with the command:

```
% openoffice.org
```

6.4 Document Viewers

Some new document formats have gained popularity since the advent of UNIX; the standard viewers they require may not be available in the base system. We will see how to install such viewers in this section.

This section covers these applications:

Application Name	Resources Needed	Installation from Ports	Major Dependencies
Acrobat Reader	light	light	Linux Binary Compatibility
gv	light	light	**Xaw3d**
Xpdf	light	light	**FreeType**
GQview	light	light	**Gtk+** or **GNOME**

6.4.1 Acrobat Reader®

Many documents are now distributed as PDF files, which stands for "Portable Document Format". One of the recommended viewers for these types of files is **Acrobat Reader**, released by Adobe for Linux. As FreeBSD can run Linux binaries, it is also available for FreeBSD.

To install **Acrobat Reader 8** from the Ports collection, do:

```
# cd /usr/ports/print/acroread8
# make install clean
```

A package is not available due to licencing restrictions.

6.4.2 gv

gv is a PostScript and PDF viewer. It is originally based on **ghostview** but it has a nicer look thanks to the **Xaw3d** library. It is fast and its interface is clean. **gv** has many features, such as orientation, paper size, scale, and anti-aliasing. Almost any operation can be done with either the keyboard or the mouse.

To install **gv** as a package, do:

```
# pkg_add -r gv
```

If you cannot get the package, you can use the Ports collection:

```
# cd /usr/ports/print/gv
# make install clean
```

6.4.3 Xpdf

If you want a small FreeBSD PDF viewer, **Xpdf** is a light-weight and efficient viewer. It requires very few resources and is very stable. It uses the standard X fonts and does not require **Motif** or any other X toolkit.

To install the **Xpdf** package, issue this command:

```
# pkg_add -r xpdf
```

If the package is not available or you prefer to use the Ports Collection, do:

```
# cd /usr/ports/graphics/xpdf
# make install clean
```

Once the installation is complete, you can launch **Xpdf** and use the right mouse button to activate the menu.

6.4.4 GQview

GQview is an image manager. You can view a file with a single click, launch an external editor, get thumbnail previews, and much more. It also features a slideshow mode and some basic file operations. You can manage image collections and easily find duplicates. **GQview** can do full screen viewing and supports internationalization.

If you want to install the **GQview** package, do:

```
# pkg_add -r gqview
```

If the package is not available or you prefer to use the Ports Collection, do:

```
# cd /usr/ports/graphics/gqview
# make install clean
```

6.5 Finance

If, for any reason, you would like to manage your personal finances on your FreeBSD Desktop, there are some powerful and easy-to-use applications ready to be installed. Some of them are compatible with widespread file formats, such as the formats used by **Quicken®** and **Excel** to store documents.

This section covers these programs:

Application Name	Resources Needed	Installation from Ports	Major Dependencies
GnuCash	light	heavy	**GNOME**
Gnumeric	light	heavy	**GNOME**
Abacus	light	light	**Tcl/Tk**
KMyMoney	light	heavy	**KDE**

6.5.1 GnuCash

GnuCash is part of the **GNOME** effort to provide user-friendly, yet powerful, applications to end-users. With **GnuCash**, you can keep track of your income and expenses, your bank accounts, and your stocks. It features an intuitive interface while remaining very professional.

GnuCash provides a smart register, a hierarchical system of accounts, and many keyboard accelerators and auto-completion methods. It can split a single transaction into several more detailed pieces. **GnuCash** can import and merge **Quicken** QIF files. It also handles most international date and currency formats.

To install **GnuCash** on your system, do:

```
# pkg_add -r gnucash
```

If the package is not available, you can use the ports collection:

```
# cd /usr/ports/finance/gnucash
# make install clean
```

6.5.2 Gnumeric

Gnumeric is a spreadsheet program, part of the **GNOME** desktop environment. It features convenient automatic "guessing" of user input according to the cell format with an autofill system for many sequences. It can import files in a number of popular formats like those of **Excel**, **Lotus 1-2-3**, or **Quattro Pro**. **Gnumeric** supports graphs through the `math/guppi` graphing program. It has a large number of built-in functions and allows all of the usual cell formats such as number, currency, date, time, and much more.

To install **Gnumeric** as a package, do:

```
# pkg_add -r gnumeric
```

If the package is not available, you can use the ports collection by doing:

```
# cd /usr/ports/math/gnumeric
# make install clean
```

6.5.3 Abacus

Abacus is a small and easy to use spreadsheet program. It includes many built-in functions useful in several domains such as statistics, finances, and mathematics. It can import and export the **Excel** file format. **Abacus** can produce PostScript output.

To install **Abacus** as a package, do:

```
# pkg_add -r abacus
```

If the package is not available, you can use the ports collection by doing:

```
# cd /usr/ports/deskutils/abacus
# make install clean
```

6.5.4 KMyMoney

KMyMoney is a personal finance manager built for **KDE**. **KMyMoney** intends to provide and incorporate all the important features found in commercial personal finance manager applications. It also highlights ease-of-use and proper double-entry accounting among its features. **KMyMoney** imports from standard Quicken Interchange Format (QIF) files, tracks investments, handles multiple currencies, and provides a wealth of reports. OFX import capabilities are also available through a separate plugin.

To install **KMyMoney** as a package, do:

```
# pkg_add -r kmymoney2
```

If the package is not available, you can use the Ports Collection by doing:

```
# cd /usr/ports/finance/kmymoney2
# make install clean
```

6.6 Summary

While FreeBSD is popular among ISPs for its performance and stability, it is quite ready for day-to-day use as a desktop. With several thousand applications available as packages (http://www.FreeBSD.org/where.html) or ports (http://www.FreeBSD.org/ports/index.html), you can build a perfect desktop that suits all your needs.

Here is a quick review of all the desktop applications covered in this chapter:

Application Name	Package Name	Ports Name
Opera	opera	www/opera
Firefox	firefox	www/firefox
KOffice	koffice-kde3	editors/koffice-kde3
AbiWord	abiword	editors/abiword
The GIMP	gimp	graphics/gimp
OpenOffice.org	openoffice	editors/openoffice.org-3
Acrobat Reader	acroread	print/acroread8
gv	gv	print/gv
Xpdf	xpdf	graphics/xpdf
GQview	gqview	graphics/gqview
GnuCash	gnucash	finance/gnucash
Gnumeric	gnumeric	math/gnumeric
Abacus	abacus	deskutils/abacus
KMyMoney	kmymoney2	finance/kmymoney2

Chapter 7
Multimedia

7.1 Synopsis

FreeBSD supports a wide variety of sound cards, allowing you to enjoy high fidelity output from your computer. This includes the ability to record and playback audio in the MPEG Audio Layer 3 (MP3), WAV, and Ogg Vorbis formats as well as many other formats. The FreeBSD Ports Collection also contains applications allowing you to edit your recorded audio, add sound effects, and control attached MIDI devices.

With some experimentation, FreeBSD can support playback of video files and DVDs. The number of applications to encode, convert, and playback various video media is more limited than the number of sound applications. For example as of this writing, there are no good re-encoding applications in the FreeBSD Ports Collection that could be used to convert between formats, as there is with `audio/sox`. However, the software landscape in this area is changing rapidly.

This chapter will describe the necessary steps to configure your sound card. The configuration and installation of X11 (Chapter 5) has already taken care of the hardware issues for your video card, though there may be some tweaks to apply for better playback.

After reading this chapter, you will know:

- How to configure your system so that your sound card is recognized.
- Methods to test whether your card is working.
- How to troubleshoot your sound setup.
- How to playback and encode MP3s and other audio.
- How video is supported by the X server.
- Some video player/encoder ports which give good results.
- How to playback DVDs, `.mpg` and `.avi` files.
- How to rip CD and DVD content into files.
- How to configure a TV card.
- How to configure an image scanner.

Before reading this chapter, you should:

- Know how to configure and install a new kernel (Chapter 8).

> **Warning:** Trying to mount audio CDs with the mount(8) command will result in an error, at least, and a *kernel panic*, at worst. These media have specialized encodings which differ from the usual ISO-filesystem.

7.2 Setting Up the Sound Card

7.2.1 Configuring the System

Before you begin, you should know the model of the card you have, the chip it uses, and whether it is a PCI or ISA card. FreeBSD supports a wide variety of both PCI and ISA cards. Check the supported audio devices list of the Hardware Notes (http://www.FreeBSD.org/releases/8.2R/hardware.html) to see if your card is supported. The Hardware Notes will also mention which driver supports your card.

To use your sound device, you will need to load the proper device driver. This may be accomplished in one of two ways. The easiest way is to simply load a kernel module for your sound card with kldload(8) which can either be done from the command line:

```
# kldload snd_emu10k1
```

or by adding the appropriate line to the file `/boot/loader.conf` like this:

```
snd_emu10k1_load="YES"
```

These examples are for a Creative SoundBlaster® Live! sound card. Other available loadable sound modules are listed in `/boot/defaults/loader.conf`. If you are not sure which driver to use, you may try to load the `snd_driver` module:

```
# kldload snd_driver
```

This is a metadriver loading the most common device drivers at once. This speeds up the search for the correct driver. It is also possible to load all sound drivers via the `/boot/loader.conf` facility.

If you wish to find out the driver selected for your soundcard after loading the `snd_driver` metadriver, you may check the `/dev/sndstat` file with the `cat /dev/sndstat` command.

A second method is to statically compile in support for your sound card in your kernel. The section below provides the information you need to add support for your hardware in this manner. For more information about recompiling your kernel, please see Chapter 8.

7.2.1.1 Configuring a Custom Kernel with Sound Support

The first thing to do is add the audio framework driver sound(4) to the kernel; for that you will need to add the following line to the kernel configuration file:

```
device sound
```

Next, you have to add the support for your sound card. Therefore, you need to know which driver supports the card. Check the supported audio devices list of the Hardware Notes (http://www.FreeBSD.org/releases/8.2R/hardware.html), to determine the correct driver for your sound card. For example, a Creative SoundBlaster Live! sound card is supported by the snd_emu10k1(4) driver. To add the support for this card, use the following:

```
device snd_emu10k1
```

Be sure to read the manual page of the driver for the syntax to use. The explicit syntax for the kernel configuration of every supported sound driver can also be found in the `/usr/src/sys/conf/NOTES` file.

Non-PnP ISA sound cards may require you to provide the kernel with information on the card settings (IRQ, I/O port, etc), as is true of all non-PnP ISA cards. This is done via the `/boot/device.hints` file. During the boot process, the loader(8) will read this file and pass the settings to the kernel. For example, an old Creative SoundBlaster 16 ISA non-PnP card will use the snd_sbc(4) driver in conjunction with `snd_sb16`. For this card the following lines must be added to the kernel configuration file:

```
device snd_sbc
device snd_sb16
```

and these to `/boot/device.hints`:

```
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"
```

In this case, the card uses the `0x220` I/O port and the IRQ `5`.

The syntax used in the `/boot/device.hints` file is covered in the sound(4) driver manual page and the manual page for the driver in question.

The settings shown above are the defaults. In some cases, you may need to change the IRQ or the other settings to match your card. See the snd_sbc(4) manual page for more information about this card.

7.2.2 Testing the Sound Card

After rebooting with the modified kernel, or after loading the required module, the sound card should appear in your system message buffer (dmesg(8)) as something like:

```
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>
```

The status of the sound card may be checked via the `/dev/sndstat` file:

```
# 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)
```

The output from your system may vary. If no `pcm` devices are listed, go back and review what was done earlier. Go through your kernel configuration file again and make sure the correct device driver was chosen. Common problems are listed in Section 7.2.2.1.

If all goes well, you should now have a functioning sound card. If your CD-ROM or DVD-ROM drive's audio-out pins are properly connected to your sound card, you can put a CD in the drive and play it with cdcontrol(1):

```
% cdcontrol -f /dev/acd0 play 1
```

Various applications, such as `audio/workman` can provide a friendlier interface. You may want to install an application such as `audio/mpg123` to listen to MP3 audio files.

Another quick way to test the card is sending data to `/dev/dsp`, like this:

```
% cat filename > /dev/dsp
```

where *filename* can be any file. This command line should produce some noise, confirming the sound card is actually working.

Sound card mixer levels can be changed via the mixer(8) command. More details can be found in the mixer(8) manual page.

7.2.2.1 Common Problems

Error	Solution
`sb_dspwr(XX) timed out`	The I/O port is not set correctly.
`bad irq XX`	The IRQ is set incorrectly. Make sure that the set IRQ and the sound IRQ are the same.
`xxx: gus pcm not attached, out of memory`	There is not enough available memory to use the device.
`xxx: can't open /dev/dsp!`	Check with `fstat | grep dsp` if another application is holding the device open. Noteworthy troublemakers are **esound** and **KDE**'s sound support.

7.2.3 Utilizing Multiple Sound Sources

It is often desirable to have multiple sources of sound that are able to play simultaneously, such as when **esound** or **artsd** do not support sharing of the sound device with a certain application.

FreeBSD lets you do this through *Virtual Sound Channels*, which can be enabled with the sysctl(8) facility. Virtual channels allow you to multiplex your sound card's playback by mixing sound in the kernel.

To set the number of virtual channels, there are three sysctl knobs which, if you are the `root` user, can be set like this:

```
# sysctl dev.pcm.0.play.vchans=4
# sysctl dev.pcm.0.rec.vchans=4
# sysctl hw.snd.maxautovchans=4
```

The above example allocates four virtual channels, which is a practical number for everyday use. Both `dev.pcm.0.play.vchans=4` and `dev.pcm.0.rec.vchans=4` are the number of virtual channels `pcm0` has for playback and recording, and are configurable once a device has been attached. `hw.snd.maxautovchans` is the number of virtual channels a new audio device is given when it is attached using kldload(8). Since the `pcm` module can be loaded independently of the hardware drivers, `hw.snd.maxautovchans` can store how many virtual channels any devices which are attached later will be given. Refer to pcm(4) manual page for more information.

> **Note:** You cannot change the number of virtual channels for a device while it is in use. First close any programs using the device, such as music players or sound daemons.

The correct pcm device will automatically be allocated transparently to a program that requests /dev/dsp0.

7.2.4 Setting Default Values for Mixer Channels

The default values for the different mixer channels are hardcoded in the sourcecode of the pcm(4) driver. There are many different applications and daemons that allow you to set values for the mixer that are remembered between invocations, but this is not a clean solution. It is possible to set default mixer values at the driver level — this is accomplished by defining the appropriate values in /boot/device.hints, e.g.:

```
hint.pcm.0.vol="50"
```

This will set the volume channel to a default value of 50 when the pcm(4) module is loaded.

7.3 MP3 Audio

MP3 (MPEG Layer 3 Audio) accomplishes near CD-quality sound, leaving no reason to let your FreeBSD workstation fall short of its offerings.

7.3.1 MP3 Players

By far, the most popular X11 MP3 player is **XMMS** (X Multimedia System). **Winamp** skins can be used with **XMMS** since the GUI is almost identical to that of Nullsoft's **Winamp**. **XMMS** also has native plug-in support.

XMMS can be installed from the multimedia/xmms port or package.

XMMS's interface is intuitive, with a playlist, graphic equalizer, and more. Those familiar with **Winamp** will find **XMMS** simple to use.

The audio/mpg123 port is an alternative, command-line MP3 player.

mpg123 can be run by specifying the sound device and the MP3 file on the command line. Assuming your audio device is /dev/dsp1.0 and you want to play the MP3 file *Foobar-GreatestHits.mp3* you would enter the following:

```
# mpg123 -a /dev/dsp1.0 Foobar-GreatestHits.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 Foobar-GreatestHits.mp3 ...
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
```

7.3.2 Ripping CD Audio Tracks

Before encoding a CD or CD track to MP3, the audio data on the CD must be ripped onto the hard drive. This is done by copying the raw CDDA (CD Digital Audio) data to WAV files.

The `cdda2wav` tool, which is a part of the `sysutils/cdrtools` suite, is used for ripping audio information from CDs and the information associated with them.

With the audio CD in the drive, the following command can be issued (as `root`) to rip an entire CD into individual (per track) WAV files:

```
# cdda2wav -D 0,1,0 -B
```

cdda2wav will support ATAPI (IDE) CDROM drives. To rip from an IDE drive, specify the device name in place of the SCSI unit numbers. For example, to rip track 7 from an IDE drive:

```
# cdda2wav -D /dev/acd0 -t 7
```

The `-D 0,1,0` indicates the SCSI device `0,1,0`, which corresponds to the output of `cdrecord -scanbus`.

To rip individual tracks, make use of the `-t` option as shown:

```
# cdda2wav -D 0,1,0 -t 7
```

This example rips track seven of the audio CDROM. To rip a range of tracks, for example, track one to seven, specify a range:

```
# cdda2wav -D 0,1,0 -t 1+7
```

The utility dd(1) can also be used to extract audio tracks on ATAPI drives, read Section 18.6.5 for more information on that possibility.

7.3.3 Encoding MP3s

Nowadays, the mp3 encoder of choice is **lame**. **Lame** can be found at `audio/lame` in the ports tree.

Using the ripped WAV files, the following command will convert *`audio01.wav`* to *`audio01.mp3`*:

```
# lame -h -b 128 \
--tt "Foo Song Title" \
--ta "FooBar Artist" \
--tl "FooBar Album" \
--ty "2001" \
--tc "Ripped and encoded by Foo" \
--tg "Genre" \
audio01.wav audio01.mp3
```

128 kbits seems to be the standard MP3 bitrate in use. Many enjoy the higher quality 160, or 192. The higher the bitrate, the more disk space the resulting MP3 will consume--but the quality will be higher. The `-h` option turns on the “higher quality but a little slower” mode. The options beginning with `--t` indicate ID3 tags, which usually contain song information, to be embedded within the MP3 file. Additional encoding options can be found by consulting the **lame** man page.

7.3.4 Decoding MP3s

In order to burn an audio CD from MP3s, they must be converted to a non-compressed WAV format. Both **XMMS** and **mpg123** support the output of MP3 to an uncompressed file format.

Writing to Disk in **XMMS**:

1. Launch **XMMS**.
2. Right-click on the window to bring up the **XMMS** menu.
3. Select `Preference` under `Options`.
4. Change the Output Plugin to “Disk Writer Plugin”.
5. Press `Configure`.
6. Enter (or choose browse) a directory to write the uncompressed files to.
7. Load the MP3 file into **XMMS** as usual, with volume at 100% and EQ settings turned off.
8. Press `Play` — **XMMS** will appear as if it is playing the MP3, but no music will be heard. It is actually playing the MP3 to a file.
9. Be sure to set the default Output Plugin back to what it was before in order to listen to MP3s again.

Writing to stdout in **mpg123**:

1. Run `mpg123 -s` *`audio01.mp3`* > *`audio01.pcm`*

XMMS writes a file in the WAV format, while **mpg123** converts the MP3 into raw PCM audio data. Both of these formats can be used with **cdrecord** to create audio CDs. You have to use raw PCM with burncd(8). If you use WAV files, you will notice a small tick sound at the beginning of each track, this sound is the header of the WAV file. You can simply remove the header of a WAV file with the utility **SoX** (it can be installed from the `audio/sox` port or package):

```
% sox -t wav -r 44100 -s -w -c 2 track.wav track.raw
```

Read Section 18.6 for more information on using a CD burner in FreeBSD.

7.4 Video Playback

Video playback is a very new and rapidly developing application area. Be patient. Not everything is going to work as smoothly as it did with sound.

Before you begin, you should know the model of the video card you have and the chip it uses. While **Xorg** supports a wide variety of video cards, fewer give good playback performance. To obtain a list of extensions supported by the X server using your card use the command xdpyinfo(1) while X11 is running.

It is a good idea to have a short MPEG file which can be treated as a test file for evaluating various players and options. Since some DVD players will look for DVD media in `/dev/dvd` by default, or have this device name hardcoded in them, you might find it useful to make symbolic links to the proper devices:

```
# ln -sf /dev/acd0 /dev/dvd
```

```
# ln -sf /dev/acd0 /dev/rdvd
```

Note that due to the nature of devfs(5), manually created links like these will not persist if you reboot your system. In order to create the symbolic links automatically whenever you boot your system, add the following lines to `/etc/devfs.conf`:

```
link acd0 dvd
link acd0 rdvd
```

Additionally, DVD decryption, which requires invoking special DVD-ROM functions, requires write permission on the DVD devices.

To enhance the shared memory X11 interface, it is recommended that the values of some sysctl(8) variables should be increased:

```
kern.ipc.shmmax=67108864
kern.ipc.shmall=32768
```

7.4.1 Determining Video Capabilities

There are several possible ways to display video under X11. What will really work is largely hardware dependent. Each method described below will have varying quality across different hardware. Secondly, the rendering of video in X11 is a topic receiving a lot of attention lately, and with each version of **Xorg**, there may be significant improvement.

A list of common video interfaces:

1. X11: normal X11 output using shared memory.
2. XVideo: an extension to the X11 interface which supports video in any X11 drawable.
3. SDL: the Simple Directmedia Layer.
4. DGA: the Direct Graphics Access.
5. SVGAlib: low level console graphics layer.

7.4.1.1 XVideo

Xorg has an extension called *XVideo* (aka Xvideo, aka Xv, aka xv) which allows video to be directly displayed in drawable objects through a special acceleration. This extension provides very good quality playback even on low-end machines.

To check whether the extension is running, use `xvinfo`:

```
% xvinfo
```

XVideo is supported for your card if the result looks like:

```
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
```

Also note that the formats listed (YUV2, YUV12, etc) are not present with every implementation of XVideo and their absence may hinder some players.

If the result looks like:

```
X-Video Extension version 2.2
screen #0
no adaptors present
```

Then XVideo is probably not supported for your card.

If XVideo is not supported for your card, this only means that it will be more difficult for your display to meet the computational demands of rendering video. Depending on your video card and processor, though, you might still be able to have a satisfying experience. You should probably read about ways of improving performance in the advanced reading Section 7.4.3.

7.4.1.2 Simple Directmedia Layer

The Simple Directmedia Layer, SDL, was intended to be a porting layer between Microsoft Windows, BeOS, and UNIX, allowing cross-platform applications to be developed which made efficient use of sound and graphics. The SDL layer provides a low-level abstraction to the hardware which can sometimes be more efficient than the X11 interface.

The SDL can be found at `devel/sdl12`.

7.4.1.3 Direct Graphics Access

Direct Graphics Access is an X11 extension which allows a program to bypass the X server and directly alter the framebuffer. Because it relies on a low level memory mapping to effect this sharing, programs using it must be run as `root`.

The DGA extension can be tested and benchmarked by dga(1). When `dga` is running, it changes the colors of the display whenever a key is pressed. To quit, use **q**.

7.4.2 Ports and Packages Dealing with Video

This section discusses the software available from the FreeBSD Ports Collection which can be used for video playback. Video playback is a very active area of software development, and the capabilities of various applications are bound to diverge somewhat from the descriptions given here.

Firstly, it is important to know that many of the video applications which run on FreeBSD were developed as Linux applications. Many of these applications are still beta-quality. Some of the problems that you may encounter with video packages on FreeBSD include:

1. An application cannot playback a file which another application produced.
2. An application cannot playback a file which the application itself produced.
3. The same application on two different machines, rebuilt on each machine for that machine, plays back the same file differently.
4. A seemingly trivial filter like rescaling of the image size results in very bad artifacts from a buggy rescaling routine.
5. An application frequently dumps core.
6. Documentation is not installed with the port and can be found either on the web or under the port's `work` directory.

Many of these applications may also exhibit "Linux-isms". That is, there may be issues resulting from the way some standard libraries are implemented in the Linux distributions, or some features of the Linux kernel which have been assumed by the authors of the applications. These issues are not always noticed and worked around by the port maintainers, which can lead to problems like these:

1. The use of `/proc/cpuinfo` to detect processor characteristics.
2. A misuse of threads which causes a program to hang upon completion instead of truly terminating.
3. Software not yet in the FreeBSD Ports Collection which is commonly used in conjunction with the application.

So far, these application developers have been cooperative with port maintainers to minimize the work-arounds needed for port-ing.

7.4.2.1 MPlayer

MPlayer is a recently developed and rapidly developing video player. The goals of the **MPlayer** team are speed and flexibility on Linux and other Unices. The project was started when the team founder got fed up with bad playback performance on then available players. Some would say that the graphical interface has been sacrificed for a streamlined design. However, once you get used to the command line options and the key-stroke controls, it works very well.

7.4.2.1.1 Building MPlayer

MPlayer resides in `multimedia/mplayer`. **MPlayer** performs a variety of hardware checks during the build process, resulting in a binary which will not be portable from one system to another. Therefore, it is important to build it from ports and not to use a binary package. Additionally, a number of options can be specified in the `make` command line, as described in the `Makefile` and at the start of the build:

```
# 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
```

The default port options should be sufficient for most users. However, if you need the XviD codec, you have to specify the `WITH_XVID` option in the command line. The default DVD device can also be defined with the `WITH_DVD_DEVICE` option, by default `/dev/acd0` will be used.

As of this writing, the **MPlayer** port will build its HTML documentation and two executables, `mplayer`, and `mencoder`, which is a tool for re-encoding video.

The HTML documentation for **MPlayer** is very informative. If the reader finds the information on video hardware and interfaces in this chapter lacking, the **MPlayer** documentation is a very thorough supplement. You should definitely take the time to read the **MPlayer** documentation if you are looking for information about video support in UNIX.

7.4.2.1.2 Using MPlayer

Any user of **MPlayer** must set up a `.mplayer` subdirectory of her home directory. To create this necessary subdirectory, you can type the following:

```
% cd /usr/ports/multimedia/mplayer
% make install-user
```

The command options for `mplayer` are listed in the manual page. For even more detail there is HTML documentation. In this section, we will describe only a few common uses.

To play a file, such as *`testfile.avi`*, through one of the various video interfaces set the `-vo` option:

```
% mplayer -vo xv testfile.avi

% mplayer -vo sdl testfile.avi

% mplayer -vo x11 testfile.avi

# mplayer -vo dga testfile.avi

# mplayer -vo 'sdl:dga' testfile.avi
```

It is worth trying all of these options, as their relative performance depends on many factors and will vary significantly with hardware.

To play from a DVD, replace the *`testfile.avi`* with `dvd://`*`N`* `-dvd-device` *`DEVICE`* where *`N`* is the title number to play and *`DEVICE`* is the device node for the DVD-ROM. For example, to play title 3 from `/dev/dvd`:

```
# mplayer -vo xv dvd://3 -dvd-device /dev/dvd
```

> **Note:** The default DVD device can be defined during the build of the **MPlayer** port via the `WITH_DVD_DEVICE` option. By default, this device is `/dev/acd0`. More details can be found in the port `Makefile`.

To stop, pause, advance and so on, consult the keybindings, which are output by running `mplayer -h` or read the manual page.

Additional important options for playback are: `-fs -zoom` which engages the fullscreen mode and `-framedrop` which helps performance.

In order for the mplayer command line to not become too large, the user can create a file `.mplayer/config` and set default options there:

```
vo=xv
fs=yes
zoom=yes
```

Finally, `mplayer` can be used to rip a DVD title into a `.vob` file. To dump out the second title from a DVD, type this:

```
# mplayer -dumpstream -dumpfile out.vob dvd://2 -dvd-device /dev/dvd
```

The output file, `out.vob`, will be MPEG and can be manipulated by the other packages described in this section.

7.4.2.1.3 mencoder

Before using `mencoder` it is a good idea to familiarize yourself with the options from the HTML documentation. There is a manual page, but it is not very useful without the HTML documentation. There are innumerable ways to improve quality, lower bitrate, and change formats, and some of these tricks may make the difference between good or bad performance. Here are a couple of examples to get you going. First a simple copy:

```
% mencoder input.avi -oac copy -ovc copy -o output.avi
```

Improper combinations of command line options can yield output files that are unplayable even by `mplayer`. Thus, if you just want to rip to a file, stick to the `-dumpfile` in `mplayer`.

To convert *`input.avi`* to the MPEG4 codec with MPEG3 audio encoding (`audio/lame` is required):

```
% mencoder input.avi -oac mp3lame -lameopts br=192 \
  -ovc lavc -lavcopts vcodec=mpeg4:vhq -o output.avi
```

This has produced output playable by `mplayer` and `xine`.

`input.avi` can be replaced with `dvd://1 -dvd-device /dev/dvd` and run as `root` to re-encode a DVD title directly. Since you are likely to be dissatisfied with your results the first time around, it is recommended you dump the title to a file and work on the file.

7.4.2.2 The xine Video Player

The **xine** video player is a project of wide scope aiming not only at being an all in one video solution, but also in producing a reusable base library and a modular executable which can be extended with plugins. It comes both as a package and as a port, `multimedia/xine`.

The **xine** player is still very rough around the edges, but it is clearly off to a good start. In practice, **xine** requires either a fast CPU with a fast video card, or support for the XVideo extension. The GUI is usable, but a bit clumsy.

As of this writing, there is no input module shipped with **xine** which will play CSS encoded DVDs. There are third party builds which do have modules for this built in them, but none of these are in the FreeBSD Ports Collection.

Compared to **MPlayer**, **xine** does more for the user, but at the same time, takes some of the more fine-grained control away from the user. The **xine** video player performs best on XVideo interfaces.

By default, **xine** player will start up in a graphical user interface. The menus can then be used to open a specific file:

```
% xine
```

Alternatively, it may be invoked to play a file immediately without the GUI with the command:

```
% xine -g -p mymovie.avi
```

7.4.2.3 The transcode Utilities

The software **transcode** is not a player, but a suite of tools for re-encoding video and audio files. With **transcode**, one has the ability to merge video files, repair broken files, using command line tools with `stdin/stdout` stream interfaces.

A great number of options can be specified during the build from the `multimedia/transcode` port, we recommend the following command line to build **transcode**:

```
# make WITH_OPTIMIZED_CFLAGS=yes WITH_LIBA52=yes WITH_LAME=yes WITH_OGG=yes \
WITH_MJPEG=yes -DWITH_XVID=yes
```

The proposed settings should be sufficient for most users.

To illustrate `transcode` capacities, one example to show how to convert a DivX file into a PAL MPEG-1 file (PAL VCD):

```
% transcode -i input.avi -V --export_prof vcd-pal -o output_vcd
% mplex -f 1 -o output_vcd.mpg output_vcd.m1v output_vcd.mpa
```

The resulting MPEG file, *`output_vcd.mpg`*, is ready to be played with **MPlayer**. You could even burn the file on a CD-R media to create a Video CD, in this case you will need to install and use both `multimedia/vcdimager` and `sysutils/cdrdao` programs.

There is a manual page for `transcode`, but you should also consult the transcode wiki (http://www.transcoding.org/cgi-bin/transcode) for further information and examples.

7.4.3 Further Reading

The various video software packages for FreeBSD are developing rapidly. It is quite possible that in the near future many of the problems discussed here will have been resolved. In the mean time, those who want to get the very most out of FreeBSD's A/V capabilities will have to cobble together knowledge from several FAQs and tutorials and use a few different applications. This section exists to give the reader pointers to such additional information.

The MPlayer documentation (http://www.mplayerhq.hu/DOCS/) is very technically informative. These documents should probably be consulted by anyone wishing to obtain a high level of expertise with UNIX video. The **MPlayer** mailing list is hostile to anyone who has not bothered to read the documentation, so if you plan on making bug reports to them, RTFM.

The xine HOWTO (http://dvd.sourceforge.net/xine-howto/en_GB/html/howto.html) contains a chapter on performance improvement which is general to all players.

Finally, there are some other promising applications which the reader may try:

- Avifile (http://avifile.sourceforge.net/) which is also a port `multimedia/avifile`.
- Ogle (http://www.dtek.chalmers.se/groups/dvd/) which is also a port `multimedia/ogle`.
- Xtheater (http://xtheater.sourceforge.net/)
- `multimedia/dvdauthor`, an open source package for authoring DVD content.

7.5 Setting Up TV Cards

7.5.1 Introduction

TV cards allow you to watch broadcast or cable TV on your computer. Most of them accept composite video via an RCA or S-video input and some of these cards come with a FM radio tuner.

FreeBSD provides support for PCI-based TV cards using a Brooktree Bt848/849/878/879 or a Conexant CN-878/Fusion 878a Video Capture Chip with the bktr(4) driver. You must also ensure the board comes with a supported tuner, consult the bktr(4) manual page for a list of supported tuners.

7.5.2 Adding the Driver

To use your card, you will need to load the bktr(4) driver, this can be done by adding the following line to the `/boot/loader.conf` file like this:

```
bktr_load="YES"
```

Alternatively, you may statically compile the support for the TV card in your kernel, in that case add the following lines to your kernel configuration:

```
device  bktr
device iicbus
device iicbb
device smbus
```

These additional device drivers are necessary because of the card components being interconnected via an I2C bus. Then build and install a new kernel.

Once the support was added to your system, you have to reboot your machine. During the boot process, your TV card should show up, like this:

```
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.
```

Of course these messages can differ according to your hardware. However you should check if the tuner is correctly detected; it is still possible to override some of the detected parameters with sysctl(8) MIBs and kernel configuration file options. For example, if you want to force the tuner to a Philips SECAM tuner, you should add the following line to your kernel configuration file:

```
options OVERRIDE_TUNER=6
```

or you can directly use sysctl(8):

```
# sysctl hw.bt848.tuner=6
```

See the bktr(4) manual page and the `/usr/src/sys/conf/NOTES` file for more details on the available options.

7.5.3 Useful Applications

To use your TV card you need to install one of the following applications:

- `multimedia/fxtv` provides TV-in-a-window and image/audio/video capture capabilities.
- `multimedia/xawtv` is also a TV application, with the same features as **fxtv**.
- `misc/alevt` decodes and displays Videotext/Teletext.
- `audio/xmradio`, an application to use the FM radio tuner coming with some TV cards.
- `audio/wmtune`, a handy desktop application for radio tuners.

More applications are available in the FreeBSD Ports Collection.

7.5.4 Troubleshooting

If you encounter any problem with your TV card, you should check at first if the video capture chip and the tuner are really supported by the bktr(4) driver and if you used the right configuration options. For more support and various questions about your TV card you may want to contact and use the archives of the freebsd-multimedia (http://lists.FreeBSD.org/mailman/listinfo/freebsd-multimedia) mailing list.

7.6 Image Scanners

7.6.1 Introduction

In FreeBSD, access to image scanners is provided by the **SANE** (Scanner Access Now Easy) API available through the FreeBSD Ports Collection. **SANE** will also use some FreeBSD device drivers to access to the scanner hardware.

FreeBSD supports both SCSI and USB scanners. Be sure your scanner is supported by **SANE** prior to performing any configuration. **SANE** has a supported devices (http://www.sane-project.org/sane-supported-devices.html) list that can provide you with information about the support for a scanner and its status. On systems prior to FreeBSD 8.X the uscanner(4) manual page also provides a list of supported USB scanners.

7.6.2 Kernel Configuration

As mentioned above both SCSI and USB interfaces are supported. According to your scanner interface, different device drivers are required.

7.6.2.1 USB Interface

The `GENERIC` kernel by default includes the device drivers needed to support USB scanners. Should you decide to use a custom kernel, be sure that the following lines are present in your kernel configuration file:

```
device usb
device uhci
device ohci
device ehci
```

On systems prior to FreeBSD 8.X, the following line is also needed:

```
device uscanner
```

On these versions of FreeBSD, the uscanner(4) device driver provides support for the USB scanners. Since FreeBSD 8.0, this support is directly provided by the libusb(3) library.

After rebooting with the correct kernel, plug in your USB scanner. A line showing the detection of your scanner should appear in the system message buffer (dmesg(8)):

```
ugen0.2: <EPSON> at usbus0
```

or on a FreeBSD 7.X system:

```
uscanner0: EPSON EPSON Scanner, rev 1.10/3.02, addr 2
```

These messages show that our scanner is using either `/dev/ugen0.2` or `/dev/uscanner0` as device node according to the FreeBSD version we run. For this example, a EPSON Perfection® 1650 USB scanner was used.

7.6.2.2 SCSI Interface

If your scanner comes with a SCSI interface, it is important to know which SCSI controller board you will use. According to the SCSI chipset used, you will have to tune your kernel configuration file. The `GENERIC` kernel supports the most common SCSI controllers. Be sure to read the `NOTES` file and add the correct line to your kernel configuration file. In addition to the SCSI adapter driver, you need to have the following lines in your kernel configuration file:

```
device scbus
device pass
```

Once your kernel has been properly compiled and installed, you should be able to see the devices in the system message buffer, when booting:

```
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
```

If your scanner was not powered-on at system boot, it is still possible to manually force the detection by performing a SCSI bus scan with the camcontrol(8) command:

```
# 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
```

Then the scanner will appear in the SCSI devices list:

```
# 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)
```

More details about SCSI devices are available in the scsi(4) and camcontrol(8) manual pages.

7.6.3 SANE Configuration

The **SANE** system is split in two parts: the backends (`graphics/sane-backends`) and the frontends (`graphics/sane-frontends`). The backends part provides access to the scanner itself. The **SANE**'s supported devices (http://www.sane-project.org/sane-supported-devices.html) list specifies which backend will support your image scanner. It is mandatory to determine the correct backend for your scanner if you want to be able to use your device. The frontends part provides the graphical scanning interface (**xscanimage**).

The first step is to install the `graphics/sane-backends` port or package. Then, use the `sane-find-scanner` command to check the scanner detection by the **SANE** system:

```
# sane-find-scanner -q
found SCSI scanner "AGFA SNAPSCAN 600 1.10" at /dev/pass3
```

The output will show the interface type of the scanner and the device node used to attach the scanner to the system. The vendor and the product model may not appear, it is not important.

> **Note:** Some USB scanners require you to load a firmware, this is explained in the backend manual page. You should also read sane-find-scanner(1) and sane(7) manual pages.

Now we have to check if the scanner will be identified by a scanning frontend. By default, the **SANE** backends comes with a command line tool called scanimage(1). This command allows you to list the devices and to perform an image acquisition from the command line. The `-L` option is used to list the scanner devices:

```
# scanimage -L
device ‘snapscan:/dev/pass3’ is a AGFA SNAPSCAN 600 flatbed scanner
```

Or, for example with the USB scanner used in the Section 7.6.2.1:

```
# scanimage -L
device ’epson2:libusb:/dev/usb:/dev/ugen0.2’ is a Epson GT-8200 flatbed scanner
```

This output comes from a FreeBSD 8.X system, the `'epson2:libusb:/dev/usb:/dev/ugen0.2'` item gives us the backend name (`epson2`) and the device node (`/dev/ugen0.2`) used by our scanner.

> **Note:** No output or a message saying that no scanners were identified indicates that scanimage(1) is unable to identify the scanner. If this happens, you will need to edit the backend configuration file and define the scanner device used. The `/usr/local/etc/sane.d/` directory contains all backend configuration files. This identification problem does appear with certain USB scanners.
>
> For example, with the USB scanner used in the Section 7.6.2.1, under FreeBSD 8.X the scanner is perfectly detected and working but under prior versions of FreeBSD (where uscanner(4) driver is used) `sane-find-scanner` gives us the following information:
>
> ```
> # sane-find-scanner -q
> found USB scanner (UNKNOWN vendor and product) at device /dev/uscanner0
> ```
>
> The scanner is correctly detected, it uses the USB interface and is attached to the `/dev/uscanner0` device node. We can now check if the scanner is correctly identified:
>
> ```
> # 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).
> ```
>
> Since the scanner is not identified, we will need to edit the `/usr/local/etc/sane.d/epson2.conf` file. The scanner model used was the EPSON Perfection 1650, so we know the scanner will use the `epson2` backend. Be sure to read the help comments in the backends configuration files. Line changes are quite simple: comment out all lines that have the wrong interface for your scanner (in our case, we will comment out all lines starting with the word `scsi` as our scanner uses the USB interface), then add at the end of the file a line specifying the interface and the device node used. In this case, we add the following line:
>
> ```
> usb /dev/uscanner0
> ```
>
> Please be sure to read the comments provided in the backend configuration file as well as the backend manual page for more details and correct syntax to use. We can now verify if the scanner is identified:
>
> ```
> # scanimage -L
> device ‘epson:/dev/uscanner0’ is a Epson GT-8200 flatbed scanner
> ```
>
> Our USB scanner has been identified. It is not important if the brand and the model do not match the scanner. The key item to be concerned with is the `‘epson:/dev/uscanner0’` field, which give us the right backend name and the right device node.

Once the `scanimage -L` command is able to see the scanner, the configuration is complete. The device is now ready to scan.

While scanimage(1) does allow us to perform an image acquisition from the command line, it is preferable to use a graphical user interface to perform image scanning. **SANE** offers a simple but efficient graphical interface: **xscanimage** (`graphics/sane-frontends`).

Xsane (`graphics/xsane`) is another popular graphical scanning frontend. This frontend offers advanced features such as various scanning mode (photocopy, fax, etc.), color correction, batch scans, etc. Both of these applications are usable as a **GIMP** plugin.

7.6.4 Giving Other Users Access to the Scanner

All previous operations have been done with `root` privileges. You may however, need other users to have access to the scanner. The user will need read and write permissions to the device node used by the scanner. As an example, our USB scanner uses the device node `/dev/ugen0.2` which is in fact just a symlink to the real device node called `/dev/usb/0.2.0` (a quick look at the contents of the `/dev` directory will confirm it). Both, the symlink and the device node, are owned respectively by the `wheel` and the `operator` groups. Adding the user *joe* to these groups will allow him to use the scanner but, for obvious security reasons, you should think twice before adding a user to any group, especially the `wheel` group. A better solution would be creating a specific group for using the USB devices and make the scanner device accessible to members of this group.

So we will use, for example, a group called *usb*. The first step is the creation of this group with the help of the pw(8) command:

```
# pw groupadd usb
```

Then we have to make the `/dev/ugen0.2` symlink and the `/dev/usb/0.2.0` device node accessible to the `usb` group with the correct write permissions (`0660` or `0664`), because by default only the owner of these files (`root`) can write to them. All of this is done by adding the following lines to the `/etc/devfs.rules` file:

```
[system=5]
add path ugen0.2 mode 0660 group usb
add path usb/0.2.0 mode 0666 group usb
```

FreeBSD 7.X users will probably need the following lines with the correct device node `/dev/uscanner0`:

```
[system=5]
add path uscanner0 mode 660 group usb
```

Then add the following to `/etc/rc.conf` and reboot the machine:

```
devfs_system_ruleset="system"
```

More information regarding these lines can be found in the devfs(8) manual page.

Now, one will just have to add users to the *usb* group to allow the access to the scanner:

```
# pw groupmod usb -m joe
```

For more details read the pw(8) manual page.

Chapter 8
Configuring the FreeBSD Kernel

8.1 Synopsis

The kernel is the core of the FreeBSD operating system. It is responsible for managing memory, enforcing security controls, networking, disk access, and much more. While more and more of FreeBSD becomes dynamically configurable it is still occasionally necessary to reconfigure and recompile your kernel.

After reading this chapter, you will know:

- Why you might need to build a custom kernel.
- How to write a kernel configuration file, or alter an existing configuration file.
- How to use the kernel configuration file to create and build a new kernel.
- How to install the new kernel.
- How to troubleshoot if things go wrong.

All of the commands listed within this chapter by way of example should be executed as `root` in order to succeed.

8.2 Why Build a Custom Kernel?

Traditionally, FreeBSD has had what is called a "monolithic" kernel. This means that the kernel was one large program, supported a fixed list of devices, and if you wanted to change the kernel's behavior then you had to compile a new kernel, and then reboot your computer with the new kernel.

Today, FreeBSD is rapidly moving to a model where much of the kernel's functionality is contained in modules which can be dynamically loaded and unloaded from the kernel as necessary. This allows the kernel to adapt to new hardware suddenly becoming available (such as PCMCIA cards in a laptop), or for new functionality to be brought into the kernel that was not necessary when the kernel was originally compiled. This is known as a modular kernel.

Despite this, it is still necessary to carry out some static kernel configuration. In some cases this is because the functionality is so tied to the kernel that it can not be made dynamically loadable. In others it may simply be because no one has yet taken the time to write a dynamic loadable kernel module for that functionality.

Building a custom kernel is one of the most important rites of passage for advanced BSD users. This process, while time consuming, will provide many benefits to your FreeBSD system. Unlike the `GENERIC` kernel, which must support a wide range of hardware, a custom kernel only contains support for *your* PC's hardware. This has a number of benefits, such as:

- Faster boot time. Since the kernel will only probe the hardware you have on your system, the time it takes your system to boot can decrease dramatically.

- Lower memory usage. A custom kernel often uses less memory than the `GENERIC` kernel by omitting unused features and device drivers. This is important because the kernel code remains resident in physical memory at all times, preventing that memory from being used by applications. For this reason, a custom kernel is especially useful on a system with a small amount of RAM.
- Additional hardware support. A custom kernel allows you to add in support for devices which are not present in the `GENERIC` kernel, such as sound cards.

8.3 Finding the System Hardware

Before venturing into kernel configuration, it would be wise to get an inventory of the machine's hardware. In cases where FreeBSD is not the primary operating system, the inventory list may easily be created by viewing the current operating system configuration. For example, Microsoft's **Device Manager** normally contains important information about installed devices. The **Device Manager** is located in the control panel.

> **Note:** Some versions of Microsoft Windows have a **System** icon which will display a screen where **Device Manager** may be accessed.

If another operating system does not exist on the machine, the administrator must find this information out manually. One method is using the dmesg(8) utility and the man(1) commands. Most device drivers on FreeBSD have a manual page, listing supported hardware, and during the boot probe, found hardware will be listed. For example, the following lines indicate that the `psm` driver found a mouse:

```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD]
psm0: model Generic PS/2 mouse, device ID 0
```

This driver will need to be included in the custom kernel configuration file or loaded using loader.conf(5).

On occasion, the data from `dmesg` will only show system messages instead of the boot probe output. In these situations, the output may be obtained by viewing the `/var/run/dmesg.boot` file.

Another method of finding hardware is by using the pciconf(8) utility which provides more verbose output. For example:

```
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
```

This bit of output, obtained using `pciconf -lv` shows that the `ath` driver located a wireless Ethernet device. Using `man ath` will return the ath(4) manual page.

The `-k` flag, when passed to man(1) can also be used to provide useful information. From the above, one can issue:

```
# man -k Atheros
```

To get a list of manual pages which contain that particular word:

```
ath(4)                  - Atheros IEEE 802.11 wireless network driver
ath_hal(4)              - Atheros Hardware Access Layer (HAL)
```

Armed with a hardware inventory list, the process of building a custom kernel should appear less daunting.

8.4 Kernel Drivers, Subsystems, and Modules

Before building a custom kernel, consider the reasons for doing so. If there is a need for specific hardware support, it may already exist as a module.

Kernel modules exist in the `/boot/kernel` directory and may be dynamically loaded into the running kernel using kldload(8). Most, if not all kernel drivers have a specific module and manual page. For example, the last section noted the `ath` wireless Ethernet driver. This device has the following information in its manual page:

```
Alternatively, to load the driver as a module at boot time, place the
following line in loader.conf(5):

    if_ath_load="YES"
```

As instructed, adding the `if_ath_load="YES"` line to the `/boot/loader.conf` file will enable loading this module dynamically at boot time.

In some cases; however, there is no associated module. This is mostly true for certain subsystems and very important drivers, for instance, the fast file system (FFS) is a required option in the kernel. As is network support (INET). Unfortunately the only way to tell if a driver is required is to check for the module itself.

> **Warning:** It is considerably easy to remove built in support for a device or option and have a broken kernel. For example, if the ata(4) driver is pulled from the kernel configuration file, a system using ATA disk drivers may not boot without the line added to `loader.conf`. When in doubt, check for the module and then just leave support in the kernel.

8.5 Building and Installing a Custom Kernel

First, let us take a quick tour of the kernel build directory. All directories mentioned will be relative to the main `/usr/src/sys` directory, which is also accessible through the path name `/sys`. There are a number of subdirectories here representing different parts of the kernel, but the most important for our purposes are *`arch`*`/conf`, where you will edit your custom kernel configuration, and `compile`, which is the staging area where your kernel will be built. *`arch`* represents one of `i386`, `amd64`, `ia64`, `powerpc`, `sparc64`, or `pc98` (an alternative development branch of PC hardware, popular in Japan). Everything inside a particular architecture's directory deals with that architecture only; the rest of the code is machine independent code common to all platforms to which FreeBSD could potentially be ported. Notice the logical organization of the directory structure, with each supported device, file system, and option in its own subdirectory.

This chapter assumes that you are using the i386 architecture in the examples. If this is not the case for your situation, make appropriate adjustments to the path names for your system's architecture.

Note: If there is *not* a `/usr/src/sys` directory on your system, then the kernel source has not been installed. The easiest way to do this is by running `sysinstall` as `root`, choosing Configure, then Distributions, then src, then base and sys. If you have an aversion to **sysinstall** and you have access to an "official" FreeBSD CDROM, then you can also install the source from the command line:

```
# mount /cdrom
# mkdir -p /usr/src/sys
# ln -s /usr/src/sys /sys
# cat /cdrom/src/ssys.[a-d]* | tar -xzvf -
# cat /cdrom/src/sbase.[a-d]* | tar -xzvf -
```

Next, move to the *arch*`/conf` directory and copy the `GENERIC` configuration file to the name you want to give your kernel. For example:

```
# cd /usr/src/sys/i386/conf
# cp GENERIC MYKERNEL
```

Traditionally, this name is in all capital letters and, if you are maintaining multiple FreeBSD machines with different hardware, it is a good idea to name it after your machine's hostname. We will call it *`MYKERNEL`* for the purpose of this example.

Tip: Storing your kernel configuration file directly under `/usr/src` can be a bad idea. If you are experiencing problems it can be tempting to just delete `/usr/src` and start again. After doing this, it usually only takes a few seconds for you to realize that you have deleted your custom kernel configuration file. Also, do not edit `GENERIC` directly, as it may get overwritten the next time you update your source tree, and your kernel modifications will be lost.

You might want to keep your kernel configuration file elsewhere, and then create a symbolic link to the file in the *`i386`* directory.

For example:

```
# cd /usr/src/sys/i386/conf
# mkdir /root/kernels
# cp GENERIC /root/kernels/MYKERNEL
# ln -s /root/kernels/MYKERNEL
```

Now, edit *`MYKERNEL`* with your favorite text editor. If you are just starting out, the only editor available will probably be **vi**, which is too complex to explain here, but is covered well in many books in the bibliography. However, FreeBSD does offer an easier editor called **ee** which, if you are a beginner, should be your editor of choice. Feel free to change the comment lines at the top to reflect your configuration or the changes you have made to differentiate it from `GENERIC`.

If you have built a kernel under SunOS or some other BSD operating system, much of this file will be very familiar to you. If you are coming from some other operating system such as DOS, on the other hand, the `GENERIC` configuration file might seem overwhelming to you, so follow the descriptions in the Configuration File section slowly and carefully.

Note: If you sync your source tree with the latest sources of the FreeBSD project, be sure to always check the file `/usr/src/UPDATING` before you perform any update steps. This file describes any important issues or areas

requiring special attention within the updated source code. `/usr/src/UPDATING` always matches your version of the FreeBSD source, and is therefore more up to date with new information than this handbook.

You must now compile the source code for the kernel.

Building a Kernel

1. Change to the `/usr/src` directory:

   ```
   # cd /usr/src
   ```

2. Compile the kernel:

   ```
   # make buildkernel KERNCONF=MYKERNEL
   ```

3. Install the new kernel:

   ```
   # make installkernel KERNCONF=MYKERNEL
   ```

Note: It is required to have full FreeBSD source tree to build the kernel.

Tip: By default, when you build a custom kernel, *all* kernel modules will be rebuilt as well. If you want to update a kernel faster or to build only custom modules, you should edit `/etc/make.conf` before starting to build the kernel:

```
MODULES_OVERRIDE = linux acpi sound/sound sound/driver/ds1 ntfs
```

This variable sets up a list of modules to build instead of all of them.

```
WITHOUT_MODULES = linux acpi sound ntfs
```

This variable sets up a list of top level modules to exclude from the build process. For other variables which you may find useful in the process of building kernel, refer to make.conf(5) manual page.

The new kernel will be copied to the `/boot/kernel` directory as `/boot/kernel/kernel` and the old kernel will be moved to `/boot/kernel.old/kernel`. Now, shutdown the system and reboot to use your new kernel. If something goes wrong, there are some troubleshooting instructions at the end of this chapter that you may find useful. Be sure to read the section which explains how to recover in case your new kernel does not boot.

Note: Other files relating to the boot process, such as the boot loader(8) and configuration are stored in `/boot`. Third party or custom modules can be placed in `/boot/kernel`, although users should be aware that keeping modules in sync with the compiled kernel is very important. Modules not intended to run with the compiled kernel may result in instability or incorrectness.

8.6 The Configuration File

The general format of a configuration file is quite simple. Each line contains a keyword and one or more arguments. For simplicity, most lines only contain one argument. Anything following a # is considered a comment and ignored. The following sections describe each keyword, in the order they are listed in `GENERIC`. For an exhaustive list of architecture dependent options and devices, see the `NOTES` file in the same directory as the `GENERIC` file. For architecture independent options, see `/usr/src/sys/conf/NOTES`.

An `include` directive is available for use in configuration files. This allows another configuration file to be logically included in the current one, making it easy to maintain small changes relative to an existing file. For example, if you require a `GENERIC` kernel with only a small number of additional options or drivers, this allows you to maintain only a delta with respect to GENERIC:

```
include GENERIC
ident MYKERNEL

options         IPFIREWALL
options         DUMMYNET
options         IPFIREWALL_DEFAULT_TO_ACCEPT
options         IPDIVERT
```

Many administrators will find that this model offers significant benefits over the historic writing of configuration files from scratch: the local configuration file will express only local differences from a `GENERIC` kernel and as upgrades are performed, new features added to `GENERIC` will be added to the local kernel unless specifically prevented using `nooptions` or `nodevice`. The remainder of this chapter addresses the contents of a typical configuration file and the role various options and devices play.

> **Note:** To build a file which contains all available options, as normally done for testing purposes, run the following command as `root`:
>
> ```
> # cd /usr/src/sys/i386/conf && make LINT
> ```

The following is an example of the `GENERIC` kernel configuration file with various additional comments where needed for clarity. This example should match your copy in `/usr/src/sys/i386/conf/GENERIC` fairly closely.

```
machine  i386
```

This is the machine architecture. It must be either `amd64`, `i386`, `ia64`, `pc98`, `powerpc`, or `sparc64`.

```
cpu          I486_CPU
cpu          I586_CPU
cpu          I686_CPU
```

The above option specifies the type of CPU you have in your system. You may have multiple instances of the CPU line (if, for example, you are not sure whether you should use `I586_CPU` or `I686_CPU`), but for a custom kernel it is best to specify only the CPU you have. If you are unsure of your CPU type, you can check the `/var/run/dmesg.boot` file to view your boot messages.

```
ident          GENERIC
```

This is the identification of the kernel. You should change this to whatever you named your kernel, i.e. *`MYKERNEL`* if you have followed the instructions of the previous examples. The value you put in the `ident` string will print when you boot up the kernel, so it is useful to give the new kernel a different name if you want to keep it separate from your usual kernel (e.g., you want to build an experimental kernel).

```
#To statically compile in device wiring instead of /boot/device.hints
#hints          "GENERIC.hints"         # Default places to look for devices.
```

The device.hints(5) is used to configure options of the device drivers. The default location that loader(8) will check at boot time is `/boot/device.hints`. Using the `hints` option you can compile these hints statically into your kernel. Then there is no need to create a `device.hints` file in `/boot`.

```
makeoptions     DEBUG=-g          # Build kernel with gdb(1) debug symbols
```

The normal build process of FreeBSD includes debugging information when building the kernel with the the `-g` option, which enables debugging information when passed to gcc(1).

```
options          SCHED_ULE         # ULE scheduler
```

The default system scheduler for FreeBSD. Keep this.

```
options          PREEMPTION         # Enable kernel thread preemption
```

Allows threads that are in the kernel to be preempted by higher priority threads. It helps with interactivity and allows interrupt threads to run sooner rather than waiting.

```
options          INET              # InterNETworking
```

Networking support. Leave this in, even if you do not plan to be connected to a network. Most programs require at least loopback networking (i.e., making network connections within your PC), so this is essentially mandatory.

```
options          INET6             # IPv6 communications protocols
```

This enables the IPv6 communication protocols.

```
options          FFS               # Berkeley Fast Filesystem
```

This is the basic hard drive file system. Leave it in if you boot from the hard disk.

```
options          SOFTUPDATES       # Enable FFS Soft Updates support
```

This option enables Soft Updates in the kernel, this will help speed up write access on the disks. Even when this functionality is provided by the kernel, it must be turned on for specific disks. Review the output from mount(8) to see if Soft Updates is enabled for your system disks. If you do not see the `soft-updates` option then you will need to activate it using the tunefs(8) (for existing file systems) or newfs(8) (for new file systems) commands.

```
options          UFS_ACL           # Support for access control lists
```

This option enables kernel support for access control lists. This relies on the use of extended attributes and UFS2, and the feature is described in detail in Section 14.11. ACLs are enabled by default and should not be disabled in the kernel if they have been used previously on a file system, as this will remove the access control lists, changing the way files are protected in unpredictable ways.

```
options         UFS_DIRHASH       # Improve performance on big directories
```

This option includes functionality to speed up disk operations on large directories, at the expense of using additional memory. You would normally keep this for a large server, or interactive workstation, and remove it if you are using FreeBSD on a smaller system where memory is at a premium and disk access speed is less important, such as a firewall.

```
options         MD_ROOT           # MD is a potential root device
```

This option enables support for a memory backed virtual disk used as a root device.

```
options         NFSCLIENT         # Network Filesystem Client
options         NFSSERVER         # Network Filesystem Server
options         NFS_ROOT          # NFS usable as /, requires NFSCLIENT
```

The network file system. Unless you plan to mount partitions from a UNIX file server over TCP/IP, you can comment these out.

```
options         MSDOSFS           # MSDOS Filesystem
```

The MS-DOS file system. Unless you plan to mount a DOS formatted hard drive partition at boot time, you can safely comment this out. It will be automatically loaded the first time you mount a DOS partition, as described above. Also, the excellent `emulators/mtools` software allows you to access DOS floppies without having to mount and unmount them (and does not require `MSDOSFS` at all).

```
options         CD9660            # ISO 9660 Filesystem
```

The ISO 9660 file system for CDROMs. Comment it out if you do not have a CDROM drive or only mount data CDs occasionally (since it will be dynamically loaded the first time you mount a data CD). Audio CDs do not need this file system.

```
options         PROCFS            # Process filesystem (requires PSEUDOFS)
```

The process file system. This is a “pretend” file system mounted on `/proc` which allows programs like ps(1) to give you more information on what processes are running. Use of `PROCFS` is not required under most circumstances, as most debugging and monitoring tools have been adapted to run without `PROCFS`: installs will not mount this file system by default.

```
options         PSEUDOFS          # Pseudo-filesystem framework
```

Kernels making use of `PROCFS` must also include support for `PSEUDOFS`.

```
options         GEOM_GPT          # GUID Partition Tables.
```

This option brings the ability to have a large number of partitions on a single disk.

```
options         COMPAT_43         # Compatible with BSD 4.3 [KEEP THIS!]
```

Compatibility with 4.3BSD. Leave this in; some programs will act strangely if you comment this out.

```
options         COMPAT_FREEBSD4   # Compatible with FreeBSD4
```

This option is required to support applications compiled on older versions of FreeBSD that use older system call interfaces. It is recommended that this option be used on all i386 systems that may run older applications; platforms that gained support only in 5.X, such as ia64 and SPARC64, do not require this option.

```
options         COMPAT_FREEBSD5   # Compatible with FreeBSD5
```

This option is required to support applications compiled on FreeBSD 5.X versions that use FreeBSD 5.X system call interfaces.

```
options         COMPAT_FREEBSD6   # Compatible with FreeBSD6
```

This option is required to support applications compiled on FreeBSD 6.X versions that use FreeBSD 6.X system call interfaces.

```
options         COMPAT_FREEBSD7   # Compatible with FreeBSD7
```

This option is required on FreeBSD 8 and above to support applications compiled on FreeBSD 7.X versions that use FreeBSD 7.X system call interfaces.

```
options         SCSI_DELAY=5000  # Delay (in ms) before probing SCSI
```

This causes the kernel to pause for 5 seconds before probing each SCSI device in your system. If you only have IDE hard drives, you can ignore this, otherwise you can try to lower this number, to speed up booting. Of course, if you do this and FreeBSD has trouble recognizing your SCSI devices, you will have to raise it again.

```
options         KTRACE            # ktrace(1) support
```

This enables kernel process tracing, which is useful in debugging.

```
options         SYSVSHM           # SYSV-style shared memory
```

This option provides for System V shared memory. The most common use of this is the XSHM extension in X, which many graphics-intensive programs will automatically take advantage of for extra speed. If you use X, you will definitely want to include this.

```
options         SYSVMSG           # SYSV-style message queues
```

Support for System V messages. This option only adds a few hundred bytes to the kernel.

```
options         SYSVSEM           # SYSV-style semaphores
```

Support for System V semaphores. Less commonly used but only adds a few hundred bytes to the kernel.

> **Note:** The `-p` option of the ipcs(1) command will list any processes using each of these System V facilities.

```
options       _KPOSIX_PRIORITY_SCHEDULING # POSIX P1003_1B real-time extensions
```

Real-time extensions added in the 1993 POSIX®. Certain applications in the Ports Collection use these (such as **StarOffice**).

```
options         KBD_INSTALL_CDEV  # install a CDEV entry in /dev
```

This option is required to allow the creation of keyboard device nodes in `/dev`.

```
options         ADAPTIVE_GIANT    # Giant mutex is adaptive.
```

Giant is the name of a mutual exclusion mechanism (a sleep mutex) that protects a large set of kernel resources. Today, this is an unacceptable performance bottleneck which is actively being replaced with locks that protect individual resources. The `ADAPTIVE_GIANT` option causes Giant to be included in the set of mutexes adaptively spun on. That is, when a thread wants to lock the Giant mutex, but it is already locked by a thread on another CPU, the first thread will keep running and wait for the lock to be released. Normally, the thread would instead go back to sleep and wait for its next chance to run. If you are not sure, leave this in.

> **Note:** Note that on FreeBSD 8.0-RELEASE and later versions, all mutexes are adaptive by default, unless explicitly set to non-adaptive by compiling with the `NO_ADAPTIVE_MUTEXES` option. As a result, Giant is adaptive by default now, and the `ADAPTIVE_GIANT` option has been removed from the kernel configuration.

```
device          apic              # I/O APIC
```

The apic device enables the use of the I/O APIC for interrupt delivery. The apic device can be used in both UP and SMP kernels, but is required for SMP kernels. Add `options SMP` to include support for multiple processors.

> **Note:** The apic device exists only on the i386 architecture, this configuration line should not be used on other architectures.

```
device          eisa
```

Include this if you have an EISA motherboard. This enables auto-detection and configuration support for all devices on the EISA bus.

```
device          pci
```

Include this if you have a PCI motherboard. This enables auto-detection of PCI cards and gatewaying from the PCI to ISA bus.

```
# Floppy drives
device          fdc
```

This is the floppy drive controller.

```
# ATA and ATAPI devices
device          ata
```

This driver supports all ATA and ATAPI devices. You only need one `device ata` line for the kernel to detect all PCI ATA/ATAPI devices on modern machines.

```
device          atadisk                 # ATA disk drives
```

This is needed along with `device ata` for ATA disk drives.

```
device          ataraid                 # ATA RAID drives
```

This is needed along with `device ata` for ATA RAID drives.

```
device          atapicd                 # ATAPI CDROM drives
```

This is needed along with `device ata` for ATAPI CDROM drives.

```
device          atapifd                 # ATAPI floppy drives
```

This is needed along with `device ata` for ATAPI floppy drives.

```
device          atapist                 # ATAPI tape drives
```

This is needed along with `device ata` for ATAPI tape drives.

```
options         ATA_STATIC_ID           # Static device numbering
```

This makes the controller number static; without this, the device numbers are dynamically allocated.

```
# SCSI Controllers
device          ahb        # EISA AHA1742 family
device          ahc        # AHA2940 and onboard AIC7xxx devices
options         AHC_REG_PRETTY_PRINT    # Print register bitfields in debug
                                        # output.  Adds ~128k to driver.
device          ahd        # AHA39320/29320 and onboard AIC79xx devices
options         AHD_REG_PRETTY_PRINT    # Print register bitfields in debug
                                        # output.  Adds ~215k to driver.
device          amd        # AMD 53C974 (Teckram DC-390(T))
device          isp        # Qlogic family
#device         ispfw      # Firmware for QLogic HBAs- normally a module
device          mpt        # LSI-Logic MPT-Fusion
#device         ncr        # NCR/Symbios Logic
device          sym        # NCR/Symbios Logic (newer chipsets + those of 'ncr')
device          trm        # Tekram DC395U/UW/F DC315U adapters

device          adv        # Advansys SCSI adapters
device          adw        # Advansys wide SCSI adapters
device          aha        # Adaptec 154x SCSI adapters
device          aic        # Adaptec 15[012]x SCSI adapters, AIC-6[23]60.
device          bt         # Buslogic/Mylex MultiMaster SCSI adapters

device          ncv        # NCR 53C500
device          nsp        # Workbit Ninja SCSI-3
device          stg        # TMC 18C30/18C50
```

SCSI controllers. Comment out any you do not have in your system. If you have an IDE only system, you can remove these altogether. The `*_REG_PRETTY_PRINT` lines are debugging options for their respective drivers.

```
# SCSI peripherals
device          scbus      # SCSI bus (required for SCSI)
device          ch         # SCSI media changers
device          da         # Direct Access (disks)
device          sa         # Sequential Access (tape etc)
device          cd         # CD
device          pass       # Passthrough device (direct SCSI access)
```

```
device          ses         # SCSI Environmental Services (and SAF-TE)
```

SCSI peripherals. Again, comment out any you do not have, or if you have only IDE hardware, you can remove them completely.

> **Note:** The USB umass(4) driver and a few other drivers use the SCSI subsystem even though they are not real SCSI devices. Therefore make sure not to remove SCSI support, if any such drivers are included in the kernel configuration.

```
# RAID controllers interfaced to the SCSI subsystem
device          amr         # AMI MegaRAID
device          arcmsr      # Areca SATA II RAID
device          asr         # DPT SmartRAID V, VI and Adaptec SCSI RAID
device          ciss        # Compaq Smart RAID 5*
device          dpt         # DPT Smartcache III, IV - See NOTES for options
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 controllers
device          aac         # Adaptec FSA RAID
device          aacp        # SCSI passthrough for aac (requires CAM)
device          ida         # Compaq Smart RAID
device          mfi         # LSI MegaRAID SAS
device          mlx         # Mylex DAC960 family
device          pst         # Promise Supertrak SX6000
device          twe         # 3ware ATA RAID
```

Supported RAID controllers. If you do not have any of these, you can comment them out or remove them.

```
# atkbdc0 controls both the keyboard and the PS/2 mouse
device          atkbdc      # AT keyboard controller
```

The keyboard controller (`atkbdc`) provides I/O services for the AT keyboard and PS/2 style pointing devices. This controller is required by the keyboard driver (`atkbd`) and the PS/2 pointing device driver (`psm`).

```
device          atkbd       # AT keyboard
```

The `atkbd` driver, together with `atkbdc` controller, provides access to the AT 84 keyboard or the AT enhanced keyboard which is connected to the AT keyboard controller.

```
device          psm         # PS/2 mouse
```

Use this device if your mouse plugs into the PS/2 mouse port.

```
device          kbdmux         # keyboard multiplexer
```

Basic support for keyboard multiplexing. If you do not plan to use more than one keyboard on the system, you can safely remove that line.

```
device          vga        # VGA video card driver
```

The video card driver.

```
device          splash     # Splash screen and screen saver support
```

Splash screen at start up! Screen savers require this too.

```
# syscons is the default console driver, resembling an SCO console
device          sc
```

`sc` is the default console driver and resembles a SCO console. Since most full-screen programs access the console through a terminal database library like `termcap`, it should not matter whether you use this or `vt`, the `VT220` compatible console driver. When you log in, set your `TERM` variable to `scoansi` if full-screen programs have trouble running under this console.

```
# Enable this for the pcvt (VT220 compatible) console driver
#device          vt
#options         XSERVER          # support for X server on a vt console
#options         FAT_CURSOR       # start with block cursor
```

This is a VT220-compatible console driver, backward compatible to VT100/102. It works well on some laptops which have hardware incompatibilities with `sc`. Also set your `TERM` variable to `vt100` or `vt220` when you log in. This driver might also prove useful when connecting to a large number of different machines over the network, where `termcap` or `terminfo` entries for the `sc` device are often not available — `vt100` should be available on virtually any platform.

```
device          agp
```

Include this if you have an AGP card in the system. This will enable support for AGP, and AGP GART for boards which have these features.

```
# Power management support (see NOTES for more options)
#device          apm
```

Advanced Power Management support. Useful for laptops, although this is disabled in `GENERIC` by default.

```
# Add suspend/resume support for the i8254.
device          pmtimer
```

Timer device driver for power management events, such as APM and ACPI.

```
# PCCARD (PCMCIA) support
# PCMCIA and cardbus bridge support
device          cbb             # cardbus (yenta) bridge
device          pccard          # PC Card (16-bit) bus
device          cardbus         # CardBus (32-bit) bus
```

PCMCIA support. You want this if you are using a laptop.

```
# Serial (COM) ports
device          sio             # 8250, 16[45]50 based serial ports
```

These are the serial ports referred to as `COM` ports in the MS-DOS/Windows world.

> **Note:** If you have an internal modem on `COM4` and a serial port at `COM2`, you will have to change the IRQ of the modem to 2 (for obscure technical reasons, IRQ2 = IRQ 9) in order to access it from FreeBSD. If you have a multiport serial card, check the manual page for sio(4) for more information on the proper values to add to your `/boot/device.hints`. Some video cards (notably those based on S3 chips) use IO addresses in the form of `0x*2e8`, and since many cheap serial cards do not fully decode the 16-bit IO address space, they clash with these cards making the `COM4` port practically unavailable.
>
> Each serial port is required to have a unique IRQ (unless you are using one of the multiport cards where shared interrupts are supported), so the default IRQs for `COM3` and `COM4` cannot be used.

```
# Parallel port
device          ppc
```

This is the ISA-bus parallel port interface.

```
device          ppbus      # Parallel port bus (required)
```

Provides support for the parallel port bus.

```
device          lpt        # Printer
```

Support for parallel port printers.

> **Note:** All three of the above are required to enable parallel printer support.

```
device          plip       # TCP/IP over parallel
```

This is the driver for the parallel network interface.

```
device          ppi        # Parallel port interface device
```

The general-purpose I/O (“geek port”) + IEEE1284 I/O.

```
#device         vpo        # Requires scbus and da
```

This is for an Iomega Zip drive. It requires `scbus` and `da` support. Best performance is achieved with ports in EPP 1.9 mode.

```
#device         puc
```

Uncomment this device if you have a “dumb” serial or parallel PCI card that is supported by the puc(4) glue driver.

```
# PCI Ethernet NICs.
device          de         # DEC/Intel DC21x4x (“Tulip”)
device          em         # Intel PRO/1000 adapter Gigabit Ethernet Card
device          ixgb       # Intel PRO/10GbE Ethernet Card
device          txp        # 3Com 3cR990 (“Typhoon”)
device          vx         # 3Com 3c590, 3c595 (“Vortex”)
```

Various PCI network card drivers. Comment out or remove any of these not present in your system.

```
# PCI Ethernet NICs that use the common MII bus controller code.
```

```
# NOTE: Be sure to keep the 'device miibus' line in order to use these NICs!
device          miibus     # MII bus support
```

MII bus support is required for some PCI 10/100 Ethernet NICs, namely those which use MII-compliant transceivers or implement transceiver control interfaces that operate like an MII. Adding `device miibus` to the kernel config pulls in support for the generic miibus API and all of the PHY drivers, including a generic one for PHYs that are not specifically handled by an individual driver.

```
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 and various workalikes
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 on-board Ethernet Networking
device          pcn        # AMD Am79C97x PCI 10/100 (precedence over 'lnc')
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")
```

Drivers that use the MII bus controller code.

```
# ISA Ethernet NICs.  pccard NICs included.
device          cs         # Crystal Semiconductor CS89x0 NIC
# 'device ed' requires 'device miibus'
device          ed         # NE[12]000, SMC Ultra, 3c503, DS8390 cards
device          ex         # Intel EtherExpress Pro/10 and Pro/10+
device          ep         # Etherlink III based cards
device          fe         # Fujitsu MB8696x based cards
device          ie         # EtherExpress 8/16, 3C507, StarLAN 10 etc.
device          lnc        # NE2100, NE32-VL Lance Ethernet cards
device          sn         # SMC's 9000 series of Ethernet chips
device          xe         # Xircom pccard Ethernet

# ISA devices that use the old ISA shims
#device         le
```

ISA Ethernet drivers. See `/usr/src/sys/`*`i386`*`/conf/NOTES` for details of which cards are supported by which driver.

```
# Wireless NIC cards
device          wlan            # 802.11 support
```

Generic 802.11 support. This line is required for wireless networking.

```
device          wlan_wep        # 802.11 WEP support
device          wlan_ccmp       # 802.11 CCMP support
device          wlan_tkip       # 802.11 TKIP support
```

Crypto support for 802.11 devices. These lines are needed if you intend to use encryption and 802.11i security protocols.

```
device          an         # Aironet 4500/4800 802.11 wireless NICs.
device          ath             # Atheros pci/cardbus NIC's
device          ath_hal         # Atheros HAL (Hardware Access Layer)
device          ath_rate_sample # SampleRate tx rate control for ath
device          awi        # BayStack 660 and others
device          ral        # Ralink Technology RT2500 wireless NICs.
device          wi         # WaveLAN/Intersil/Symbol 802.11 wireless NICs.
#device         wl         # Older non 802.11 Wavelan wireless NIC.
```

Support for various wireless cards.

```
# Pseudo devices
device   loop          # Network loopback
```

This is the generic loopback device for TCP/IP. If you telnet or FTP to `localhost` (a.k.a. `127.0.0.1`) it will come back at you through this device. This is *mandatory*.

```
device   random        # Entropy device
```

Cryptographically secure random number generator.

```
device   ether         # Ethernet support
```

`ether` is only needed if you have an Ethernet card. It includes generic Ethernet protocol code.

```
device   sl            # Kernel SLIP
```

`sl` is for SLIP support. This has been almost entirely supplanted by PPP, which is easier to set up, better suited for modem-to-modem connection, and more powerful.

```
device   ppp           # Kernel PPP
```

This is for kernel PPP support for dial-up connections. There is also a version of PPP implemented as a userland application that uses `tun` and offers more flexibility and features such as demand dialing.

```
device   tun           # Packet tunnel.
```

This is used by the userland PPP software. See the PPP section of this book for more information.

```
device   pty           # Pseudo-ttys (telnet etc)
```

This is a “pseudo-terminal” or simulated login port. It is used by incoming `telnet` and `rlogin` sessions, **xterm**, and some other applications such as **Emacs**.

```
device   md            # Memory “disks”
```

Memory disk pseudo-devices.

```
device   gif           # IPv6 and IPv4 tunneling
```

This implements IPv6 over IPv4 tunneling, IPv4 over IPv6 tunneling, IPv4 over IPv4 tunneling, and IPv6 over IPv6 tunneling. The `gif` device is “auto-cloning”, and will create device nodes as needed.

```
device   faith         # IPv6-to-IPv4 relaying (translation)
```

This pseudo-device captures packets that are sent to it and diverts them to the IPv4/IPv6 translation daemon.

```
# The ‘bpf’ device enables the Berkeley Packet Filter.
# Be aware of the administrative consequences of enabling this!
# Note that ’bpf’ is required for DHCP.
device   bpf           # Berkeley packet filter
```

This is the Berkeley Packet Filter. This pseudo-device allows network interfaces to be placed in promiscuous mode, capturing every packet on a broadcast network (e.g., an Ethernet). These packets can be captured to disk and or examined with the tcpdump(1) program.

> **Note:** The bpf(4) device is also used by dhclient(8) to obtain the IP address of the default router (gateway) and so on. If you use DHCP, leave this uncommented.

```
# USB support
device          uhci          # UHCI PCI->USB interface
device          ohci          # OHCI PCI->USB interface
device          ehci          # EHCI PCI->USB interface (USB 2.0)
device          usb           # USB Bus (required)
#device         udbp          # USB Double Bulk Pipe devices
device          ugen          # Generic
device          uhid          # “Human Interface Devices”
device          ukbd          # Keyboard
device          ulpt          # Printer
device          umass         # Disks/Mass storage - Requires scbus and da
device          ums           # Mouse
device          ural          # Ralink Technology RT2500USB wireless NICs
device          urio          # Diamond Rio 500 MP3 player
device          uscanner      # Scanners
# USB Ethernet, requires mii
device          aue           # ADMtek USB Ethernet
device          axe           # ASIX Electronics USB Ethernet
device          cdce          # Generic USB over Ethernet
device          cue           # CATC USB Ethernet
device          kue           # Kawasaki LSI USB Ethernet
device          rue           # RealTek RTL8150 USB Ethernet
```

Support for various USB devices.

```
# FireWire support
device          firewire      # FireWire bus code
device          sbp           # SCSI over FireWire (Requires scbus and da)
device          fwe           # Ethernet over FireWire (non-standard!)
```

Support for various Firewire devices.

For more information and additional devices supported by FreeBSD, see `/usr/src/sys/i386/conf/NOTES`.

8.6.1 Large Memory Configurations (PAE)

Large memory configuration machines require access to more than the 4 gigabyte limit on User+Kernel Virtual Address (KVA) space. Due to this limitation, Intel added support for 36-bit physical address space access in the Pentium Pro and later line of CPUs.

The Physical Address Extension (PAE) capability of the Intel Pentium Pro and later CPUs allows memory configurations of up to 64 gigabytes. FreeBSD provides support for this capability via the `PAE` kernel configuration option, available in all current release versions of FreeBSD. Due to the limitations of the Intel memory architecture, no distinction is made for memory above or below 4 gigabytes. Memory allocated above 4 gigabytes is simply added to the pool of available memory.

To enable PAE support in the kernel, simply add the following line to your kernel configuration file:

```
options      PAE
```

> **Note:** The PAE support in FreeBSD is only available for Intel IA-32 processors. It should also be noted, that the PAE support in FreeBSD has not received wide testing, and should be considered beta quality compared to other stable features of FreeBSD.

PAE support in FreeBSD has a few limitations:

- A process is not able to access more than 4 gigabytes of VM space.
- Device drivers that do not use the bus_dma(9) interface will cause data corruption in a PAE enabled kernel and are not recommended for use. For this reason, a `PAE` kernel configuration file is provided in FreeBSD which excludes all drivers not known to work in a PAE enabled kernel.
- Some system tunables determine memory resource usage by the amount of available physical memory. Such tunables can unnecessarily over-allocate due to the large memory nature of a PAE system. One such example is the `kern.maxvnodes` sysctl, which controls the maximum number of vnodes allowed in the kernel. It is advised to adjust this and other such tunables to a reasonable value.
- It might be necessary to increase the kernel virtual address (KVA) space or to reduce the amount of specific kernel resource that is heavily used (see above) in order to avoid KVA exhaustion. The `KVA_PAGES` kernel option can be used for increasing the KVA space.

For performance and stability concerns, it is advised to consult the tuning(7) manual page. The pae(4) manual page contains up-to-date information on FreeBSD's PAE support.

8.7 If Something Goes Wrong

There are four categories of trouble that can occur when building a custom kernel. They are:

`config` fails:

If the config(8) command fails when you give it your kernel description, you have probably made a simple error somewhere. Fortunately, config(8) will print the line number that it had trouble with, so that you can quickly locate the line containing the error. For example, if you see:

```
config: line 17: syntax error
```

Make sure the keyword is typed correctly by comparing it to the `GENERIC` kernel or another reference.

`make` fails:

If the `make` command fails, it usually signals an error in your kernel description which is not severe enough for config(8) to catch. Again, look over your configuration, and if you still cannot resolve the problem, send mail to the FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) with your kernel configuration, and it should be diagnosed quickly.

The kernel does not boot:

If your new kernel does not boot, or fails to recognize your devices, do not panic! Fortunately, FreeBSD has an excellent mechanism for recovering from incompatible kernels. Simply choose the kernel you want to boot from at the FreeBSD boot loader. You can access this when the system boot menu appears. Select the "Escape to a loader prompt" option, number six. At the prompt, type `unload kernel` and then type `boot /boot/`*`kernel.old`*`/kernel`, or the filename of any other kernel that will boot properly. When reconfiguring a kernel, it is always a good idea to keep a kernel that is known to work on hand.

After booting with a good kernel you can check over your configuration file and try to build it again. One helpful resource is the `/var/log/messages` file which records, among other things, all of the kernel messages from every successful boot. Also, the dmesg(8) command will print the kernel messages from the current boot.

> **Note:** If you are having trouble building a kernel, make sure to keep a `GENERIC`, or some other kernel that is known to work on hand as a different name that will not get erased on the next build. You cannot rely on `kernel.old` because when installing a new kernel, `kernel.old` is overwritten with the last installed kernel which may be non-functional. Also, as soon as possible, move the working kernel to the proper `/boot/kernel` location or commands such as ps(1) may not work properly. To do this, simply rename the directory containing the good kernel:
>
> ```
> # mv /boot/kernel /boot/kernel.bad
> # mv /boot/kernel.good /boot/kernel
> ```

The kernel works, but ps(1) does not work any more:

If you have installed a different version of the kernel from the one that the system utilities have been built with, for example, a -CURRENT kernel on a -RELEASE, many system-status commands like ps(1) and vmstat(8) will not work any more. You should recompile and install a world built with the same version of the source tree as your kernel. This is one reason it is not normally a good idea to use a different version of the kernel from the rest of the operating system.

Chapter 9
Printing

9.1 Synopsis

FreeBSD can be used to print with a wide variety of printers, from the oldest impact printer to the latest laser printers, and everything in between, allowing you to produce high-quality printed output from the applications you run.

FreeBSD can also be configured to act as a print server on a network; in this capacity FreeBSD can receive print jobs from a variety of other computers, including other FreeBSD computers, Windows and Mac OS hosts. FreeBSD will ensure that one job at a time is printed, and can keep statistics on which users and machines are doing the most printing, produce "banner" pages showing whose printout is whose, and more.

After reading this chapter, you will know:

- How to configure the FreeBSD print spooler.
- How to install print filters, to handle special print jobs differently, including converting incoming documents to print formats that your printers understand.
- How to enable header, or banner pages on your printout.
- How to print with printers connected to other computers.
- How to print with printers connected directly to the network.
- How to control printer restrictions, including limiting the size of print jobs, and preventing certain users from printing.
- How to keep printer statistics, and account for printer usage.
- How to troubleshoot printing problems.

Before reading this chapter, you should:

- Know how to configure and install a new kernel (Chapter 8).

9.2 Introduction

In order to use printers with FreeBSD you may set them up to work with the Berkeley line printer spooling system, also known as the **LPD** spooling system, or just **LPD**. It is the standard printer control system in FreeBSD. This chapter introduces **LPD** and will guide you through its configuration.

If you are already familiar with **LPD** or another printer spooling system, you may wish to skip to section Basic Setup.

LPD controls everything about a host's printers. It is responsible for a number of things:

- It controls access to attached printers and printers attached to other hosts on the network.

- It enables users to submit files to be printed; these submissions are known as *jobs*.
- It prevents multiple users from accessing a printer at the same time by maintaining a *queue* for each printer.
- It can print *header pages* (also known as *banner* or *burst* pages) so users can easily find jobs they have printed in a stack of printouts.
- It takes care of communications parameters for printers connected on serial ports.
- It can send jobs over the network to a **LPD** spooler on another host.
- It can run special filters to format jobs to be printed for various printer languages or printer capabilities.
- It can account for printer usage.

Through a configuration file (`/etc/printcap`), and by providing the special filter programs, you can enable the **LPD** system to do all or some subset of the above for a great variety of printer hardware.

9.2.1 Why You Should Use the Spooler

If you are the sole user of your system, you may be wondering why you should bother with the spooler when you do not need access control, header pages, or printer accounting. While it is possible to enable direct access to a printer, you should use the spooler anyway since:

- **LPD** prints jobs in the background; you do not have to wait for data to be copied to the printer.
- **LPD** can conveniently run a job to be printed through filters to add date/time headers or convert a special file format (such as a TEX DVI file) into a format the printer will understand. You will not have to do these steps manually.
- Many free and commercial programs that provide a print feature usually expect to talk to the spooler on your system. By setting up the spooling system, you will more easily support other software you may later add or already have.

9.3 Basic Setup

> **Warning:** As of FreeBSD 8.0, device nodes for serial ports have been renamed from `/dev/ttydN` to `/dev/ttyuN`. FreeBSD 7.X users will have to adapt the following documentation according to these changes.

To use printers with the **LPD** spooling system, you will need to set up both your printer hardware and the **LPD** software. This document describes two levels of setup:

- See section Simple Printer Setup to learn how to connect a printer, tell **LPD** how to communicate with it, and print plain text files to the printer.
- See section Advanced Printer Setup to learn how to print a variety of special file formats, to print header pages, to print across a network, to control access to printers, and to do printer accounting.

9.3.1 Simple Printer Setup

This section tells how to configure printer hardware and the **LPD** software to use the printer. It teaches the basics:

- Section Hardware Setup gives some hints on connecting the printer to a port on your computer.
- Section Software Setup shows how to set up the **LPD** spooler configuration file (`/etc/printcap`).

If you are setting up a printer that uses a network protocol to accept data to print instead of a computer's local interfaces, see Printers With Networked Data Stream Interfaces.

Although this section is called "Simple Printer Setup", it is actually fairly complex. Getting the printer to work with your computer and the **LPD** spooler is the hardest part. The advanced options like header pages and accounting are fairly easy once you get the printer working.

9.3.1.1 Hardware Setup

This section tells about the various ways you can connect a printer to your PC. It talks about the kinds of ports and cables, and also the kernel configuration you may need to enable FreeBSD to speak to the printer.

If you have already connected your printer and have successfully printed with it under another operating system, you can probably skip to section Software Setup.

9.3.1.1.1 Ports and Cables

Printers sold for use on PC's today generally come with one or more of the following three interfaces:

- *Serial* interfaces, also known as RS-232 or COM ports, use a serial port on your computer to send data to the printer. Serial interfaces are common in the computer industry and cables are readily available and also easy to construct. Serial interfaces sometimes need special cables and might require you to configure somewhat complex communications options. Most PC serial ports have a maximum transmission rate of 115200 bps, which makes printing large graphic print jobs with them impractical.
- *Parallel* interfaces use a parallel port on your computer to send data to the printer. Parallel interfaces are common in the PC market and are faster than RS-232 serial. Cables are readily available but more difficult to construct by hand. There are usually no communications options with parallel interfaces, making their configuration exceedingly simple.

  Parallel interfaces are sometimes known as "Centronics" interfaces, named after the connector type on the printer.
- USB interfaces, named for the Universal Serial Bus, can run at even faster speeds than parallel or RS-232 serial interfaces. Cables are simple and cheap. USB is superior to RS-232 Serial and to Parallel for printing, but it is not as well supported under UNIX systems. A way to avoid this problem is to purchase a printer that has both a USB interface and a Parallel interface, as many printers do.

In general, Parallel interfaces usually offer just one-way communication (computer to printer) while serial and USB gives you two-way. Newer parallel ports (EPP and ECP) and printers can communicate in both directions under FreeBSD when a IEEE-1284-compliant cable is used.

Two-way communication to the printer over a parallel port is generally done in one of two ways. The first method uses a custom-built printer driver for FreeBSD that speaks the proprietary language used by the printer. This is common with inkjet printers and can be used for reporting ink levels and other status information. The second method is used when the printer supports PostScript.

PostScript jobs are actually programs sent to the printer; they need not produce paper at all and may return results directly to the computer. PostScript also uses two-way communication to tell the computer about problems, such as errors in the PostScript program or paper jams. Your users may be appreciative of such information. Furthermore, the best way to do effective accounting with a PostScript printer requires two-way communication: you ask the printer for its page count (how many pages it has printed in its lifetime), then send the user's job, then ask again for its page count. Subtract the two values and you know how much paper to charge to the user.

9.3.1.1.2 Parallel Ports

To hook up a printer using a parallel interface, connect the Centronics cable between the printer and the computer. The instructions that came with the printer, the computer, or both should give you complete guidance.

Remember which parallel port you used on the computer. The first parallel port is `ppc0` to FreeBSD; the second is `ppc1`, and so on. The printer device name uses the same scheme: `/dev/lpt0` for the printer on the first parallel ports etc.

9.3.1.1.3 Serial Ports

To hook up a printer using a serial interface, connect the proper serial cable between the printer and the computer. The instructions that came with the printer, the computer, or both should give you complete guidance.

If you are unsure what the "proper serial cable" is, you may wish to try one of the following alternatives:

- A *modem* cable connects each pin of the connector on one end of the cable straight through to its corresponding pin of the connector on the other end. This type of cable is also known as a "DTE-to-DCE" cable.
- A *null-modem* cable connects some pins straight through, swaps others (send data to receive data, for example), and shorts some internally in each connector hood. This type of cable is also known as a "DTE-to-DTE" cable.
- A *serial printer* cable, required for some unusual printers, is like the null-modem cable, but sends some signals to their counterparts instead of being internally shorted.

You should also set up the communications parameters for the printer, usually through front-panel controls or DIP switches on the printer. Choose the highest `bps` (bits per second, sometimes *baud rate*) that both your computer and the printer can support. Choose 7 or 8 data bits; none, even, or odd parity; and 1 or 2 stop bits. Also choose a flow control protocol: either none, or XON/XOFF (also known as "in-band" or "software") flow control. Remember these settings for the software configuration that follows.

9.3.1.2 Software Setup

This section describes the software setup necessary to print with the **LPD** spooling system in FreeBSD.

Here is an outline of the steps involved:

1. Configure your kernel, if necessary, for the port you are using for the printer; section Kernel Configuration tells you what you need to do.
2. Set the communications mode for the parallel port, if you are using a parallel port; section Setting the Communication Mode for the Parallel Port gives details.

3. Test if the operating system can send data to the printer. Section Checking Printer Communications gives some suggestions on how to do this.
4. Set up **LPD** for the printer by modifying the file `/etc/printcap`. You will find out how to do this later in this chapter.

9.3.1.2.1 Kernel Configuration

The operating system kernel is compiled to work with a specific set of devices. The serial or parallel interface for your printer is a part of that set. Therefore, it might be necessary to add support for an additional serial or parallel port if your kernel is not already configured for one.

To find out if the kernel you are currently using supports a serial interface, type:

```
# grep sioN /var/run/dmesg.boot
```

Where *N* is the number of the serial port, starting from zero. If you see output similar to the following:

```
sio2 at port 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
```

then the kernel supports the port.

To find out if the kernel supports a parallel interface, type:

```
# grep ppcN /var/run/dmesg.boot
```

Where *N* is the number of the parallel port, starting from zero. If you see output similar to the following:

```
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
```

then the kernel supports the port.

You might have to reconfigure your kernel in order for the operating system to recognize and use the parallel or serial port you are using for the printer.

To add support for a serial port, see the section on kernel configuration. To add support for a parallel port, see that section *and* the section that follows.

9.3.1.3 Setting the Communication Mode for the Parallel Port

When you are using the parallel interface, you can choose whether FreeBSD should use interrupt-driven or polled communication with the printer. The generic printer device driver (lpt(4)) on FreeBSD uses the ppbus(4) system, which controls the port chipset with the ppc(4) driver.

- The *interrupt-driven* method is the default with the GENERIC kernel. With this method, the operating system uses an IRQ line to determine when the printer is ready for data.
- The *polled* method directs the operating system to repeatedly ask the printer if it is ready for more data. When it responds ready, the kernel sends more data.

The interrupt-driven method is usually somewhat faster but uses up a precious IRQ line. Some newer HP printers are claimed not to work correctly in interrupt mode, apparently due to some (not yet exactly understood) timing problem. These printers need polled mode. You should use whichever one works. Some printers will work in both modes, but are painfully slow in interrupt mode.

You can set the communications mode in two ways: by configuring the kernel or by using the lptcontrol(8) program.

To set the communications mode by configuring the kernel:

1. Edit your kernel configuration file. Look for an `ppc0` entry. If you are setting up the second parallel port, use `ppc1` instead. Use `ppc2` for the third port, and so on.
   - If you want interrupt-driven mode, edit the following line:

     `hint.ppc.0.irq="N"`

     in the `/boot/device.hints` file and replace *`N`* with the right IRQ number. The kernel configuration file must also contain the ppc(4) driver:

     `device ppc`
   - If you want polled mode, remove in your `/boot/device.hints` file, the following line:

     `hint.ppc.0.irq="N"`

     In some cases, this is not enough to put the port in polled mode under FreeBSD. Most of time it comes from acpi(4) driver, this latter is able to probe and attach devices, and therefore, control the access mode to the printer port. You should check your acpi(4) configuration to correct this problem.
2. Save the file. Then configure, build, and install the kernel, then reboot. See kernel configuration for more details.

To set the communications mode with lptcontrol(8):

1. Type:

   `# lptcontrol -i -d /dev/lptN`

   to set interrupt-driven mode for `lptN`.
2. Type:

   `# lptcontrol -p -d /dev/lptN`

   to set polled-mode for `lptN`.

You could put these commands in your `/etc/rc.local` file to set the mode each time your system boots. See lptcontrol(8) for more information.

9.3.1.4 Checking Printer Communications

Before proceeding to configure the spooling system, you should make sure the operating system can successfully send data to your printer. It is a lot easier to debug printer communication and the spooling system separately.

To test the printer, we will send some text to it. For printers that can immediately print characters sent to them, the program lptest(1) is perfect: it generates all 96 printable ASCII characters in 96 lines.

For a PostScript (or other language-based) printer, we will need a more sophisticated test. A small PostScript program, such as the following, will suffice:

```
%!PS
100 100 moveto 300 300 lineto stroke
310 310 moveto /Helvetica findfont 12 scalefont setfont
(Is this thing working?) show
showpage
```

The above PostScript code can be placed into a file and used as shown in the examples appearing in the following sections.

> **Note:** When this document refers to a printer language, it is assuming a language like PostScript, and not Hewlett Packard's PCL. Although PCL has great functionality, you can intermingle plain text with its escape sequences. PostScript cannot directly print plain text, and that is the kind of printer language for which we must make special accommodations.

9.3.1.4.1 Checking a Parallel Printer

This section tells you how to check if FreeBSD can communicate with a printer connected to a parallel port.

To test a printer on a parallel port:

1. Become `root` with su(1).
2. Send data to the printer.
   - If the printer can print plain text, then use lptest(1). Type:

     `# lptest > /dev/lptN`

     Where *`N`* is the number of the parallel port, starting from zero.
   - If the printer understands PostScript or other printer language, then send a small program to the printer. Type:

     `# cat > /dev/lptN`

     Then, line by line, type the program *carefully* as you cannot edit a line once you have pressed `RETURN` or `ENTER`. When you have finished entering the program, press `CONTROL+D`, or whatever your end of file key is.

     Alternatively, you can put the program in a file and type:

     `# cat file > /dev/lptN`

     Where *`file`* is the name of the file containing the program you want to send to the printer.

You should see something print. Do not worry if the text does not look right; we will fix such things later.

9.3.1.4.2 Checking a Serial Printer

This section tells you how to check if FreeBSD can communicate with a printer on a serial port.

To test a printer on a serial port:

1. Become `root` with su(1).
2. Edit the file `/etc/remote`. Add the following entry:

   `printer:dv=/dev/port:br#bps-rate:pa=parity`

Where *port* is the device entry for the serial port (`ttyu0`, `ttyu1`, etc.), *bps-rate* is the bits-per-second rate at which the printer communicates, and *parity* is the parity required by the printer (either `even`, `odd`, `none`, or `zero`).

Here is a sample entry for a printer connected via a serial line to the third serial port at 19200 bps with no parity:

```
printer:dv=/dev/ttyu2:br#19200:pa=none
```

3. Connect to the printer with tip(1). Type:

```
# tip printer
```

If this step does not work, edit the file `/etc/remote` again and try using `/dev/cuaa`*N* instead of `/dev/ttyu`*N*.

4. Send data to the printer.

- If the printer can print plain text, then use lptest(1). Type:

```
% $lptest
```

- If the printer understands PostScript or other printer language, then send a small program to the printer. Type the program, line by line, *very carefully* as backspacing or other editing keys may be significant to the printer. You may also need to type a special end-of-file key for the printer so it knows it received the whole program. For PostScript printers, press `CONTROL+D`.

  Alternatively, you can put the program in a file and type:

```
% >file
```

  Where *file* is the name of the file containing the program. After tip(1) sends the file, press any required end-of-file key.

You should see something print. Do not worry if the text does not look right; we will fix that later.

9.3.1.5 Enabling the Spooler: the `/etc/printcap` File

At this point, your printer should be hooked up, your kernel configured to communicate with it (if necessary), and you have been able to send some simple data to the printer. Now, we are ready to configure **LPD** to control access to your printer.

You configure **LPD** by editing the file `/etc/printcap`. The **LPD** spooling system reads this file each time the spooler is used, so updates to the file take immediate effect.

The format of the printcap(5) file is straightforward. Use your favorite text editor to make changes to `/etc/printcap`. The format is identical to other capability files like `/usr/share/misc/termcap` and `/etc/remote`. For complete information about the format, see the cgetent(3).

The simple spooler configuration consists of the following steps:

1. Pick a name (and a few convenient aliases) for the printer, and put them in the `/etc/printcap` file; see the Naming the Printer section for more information on naming.
2. Turn off header pages (which are on by default) by inserting the `sh` capability; see the Suppressing Header Pages section for more information.
3. Make a spooling directory, and specify its location with the `sd` capability; see the Making the Spooling Directory section for more information.

4. Set the `/dev` entry to use for the printer, and note it in `/etc/printcap` with the `lp` capability; see the Identifying the Printer Device for more information. Also, if the printer is on a serial port, set up the communication parameters with the `ms#` capability which is discussed in the Configuring Spooler Communications Parameters section.
5. Install a plain text input filter; see the Installing the Text Filter section for details.
6. Test the setup by printing something with the lpr(1) command. More details are available in the Trying It Out and Troubleshooting sections.

> **Note:** Language-based printers, such as PostScript printers, cannot directly print plain text. The simple setup outlined above and described in the following sections assumes that if you are installing such a printer you will print only files that the printer can understand.

Users often expect that they can print plain text to any of the printers installed on your system. Programs that interface to **LPD** to do their printing usually make the same assumption. If you are installing such a printer and want to be able to print jobs in the printer language *and* print plain text jobs, you are strongly urged to add an additional step to the simple setup outlined above: install an automatic plain-text-to-PostScript (or other printer language) conversion program. The section entitled Accommodating Plain Text Jobs on PostScript Printers tells how to do this.

9.3.1.5.1 Naming the Printer

The first (easy) step is to pick a name for your printer. It really does not matter whether you choose functional or whimsical names since you can also provide a number of aliases for the printer.

At least one of the printers specified in the `/etc/printcap` should have the alias `lp`. This is the default printer's name. If users do not have the `PRINTER` environment variable nor specify a printer name on the command line of any of the **LPD** commands, then `lp` will be the default printer they get to use.

Also, it is common practice to make the last alias for a printer be a full description of the printer, including make and model.

Once you have picked a name and some common aliases, put them in the `/etc/printcap` file. The name of the printer should start in the leftmost column. Separate each alias with a vertical bar and put a colon after the last alias.

In the following example, we start with a skeletal `/etc/printcap` that defines two printers (a Diablo 630 line printer and a Panasonic KX-P4455 PostScript laser printer):

```
#
#  /etc/printcap for host rose
#
rattan|line|diablo|lp|Diablo 630 Line Printer:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:
```

In this example, the first printer is named `rattan` and has as aliases `line`, `diablo`, `lp`, and `Diablo 630 Line Printer`. Since it has the alias `lp`, it is also the default printer. The second is named `bamboo`, and has as aliases `ps`, `PS`, `S`, `panasonic`, and `Panasonic KX-P4455 PostScript v51.4`.

9.3.1.5.2 Suppressing Header Pages

The **LPD** spooling system will by default print a *header page* for each job. The header page contains the user name who requested the job, the host from which the job came, and the name of the job, in nice large letters. Unfortunately, all this extra text gets in the way of debugging the simple printer setup, so we will suppress header pages.

To suppress header pages, add the `sh` capability to the entry for the printer in `/etc/printcap`. Here is an example `/etc/printcap` with `sh` added:

```
#
#  /etc/printcap for host rose - no header pages anywhere
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:
```

Note how we used the correct format: the first line starts in the leftmost column, and subsequent lines are indented. Every line in an entry except the last ends in a backslash character.

9.3.1.5.3 Making the Spooling Directory

The next step in the simple spooler setup is to make a *spooling directory*, a directory where print jobs reside until they are printed, and where a number of other spooler support files live.

Because of the variable nature of spooling directories, it is customary to put these directories under `/var/spool`. It is not necessary to backup the contents of spooling directories, either. Recreating them is as simple as running mkdir(1).

It is also customary to make the directory with a name that is identical to the name of the printer, as shown below:

```
# mkdir /var/spool/printer-name
```

However, if you have a lot of printers on your network, you might want to put the spooling directories under a single directory that you reserve just for printing with **LPD**. We will do this for our two example printers `rattan` and `bamboo`:

```
# mkdir /var/spool/lpd
# mkdir /var/spool/lpd/rattan
# mkdir /var/spool/lpd/bamboo
```

> **Note:** If you are concerned about the privacy of jobs that users print, you might want to protect the spooling directory so it is not publicly accessible. Spooling directories should be owned and be readable, writable, and searchable by user `daemon` and group `daemon`, and no one else. We will do this for our example printers:
>
> ```
> # 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
> ```

Finally, you need to tell **LPD** about these directories using the `/etc/printcap` file. You specify the pathname of the spooling directory with the `sd` capability:

```
#
#  /etc/printcap for host rose - added spooling directories
#
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:
```

Note that the name of the printer starts in the first column but all other entries describing the printer should be indented and each line end escaped with a backslash.

If you do not specify a spooling directory with `sd`, the spooling system will use `/var/spool/lpd` as a default.

9.3.1.5.4 Identifying the Printer Device

In the Hardware Setup section, we identified the port and the relevant `/dev` directory entry that FreeBSD will use to communicate with the printer. Now, we tell **LPD** that information. When the spooling system has a job to print, it will open the specified device on behalf of the filter program (which is responsible for passing data to the printer).

List the `/dev` entry pathname in the `/etc/printcap` file using the `lp` capability.

In our running example, let us assume that `rattan` is on the first parallel port, and `bamboo` is on a sixth serial port; here are the additions to `/etc/printcap`:

```
#
#  /etc/printcap for host rose - identified what devices to use
#
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/ttyu5:
```

If you do not specify the `lp` capability for a printer in your `/etc/printcap` file, **LPD** uses `/dev/lp` as a default. `/dev/lp` currently does not exist in FreeBSD.

If the printer you are installing is connected to a parallel port, skip to the section entitled, Installing the Text Filter. Otherwise, be sure to follow the instructions in the next section.

9.3.1.5.5 Configuring Spooler Communication Parameters

For printers on serial ports, **LPD** can set up the bps rate, parity, and other serial communication parameters on behalf of the filter program that sends data to the printer. This is advantageous since:

- It lets you try different communication parameters by simply editing the `/etc/printcap` file; you do not have to recompile the filter program.

- It enables the spooling system to use the same filter program for multiple printers which may have different serial communication settings.

The following `/etc/printcap` capabilities control serial communication parameters of the device listed in the `lp` capability:

`br#`*`bps-rate`*

Sets the communications speed of the device to *`bps-rate`*, where *`bps-rate`* can be 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, or 115200 bits-per-second.

`ms#`*`stty-mode`*

Sets the options for the terminal device after opening the device. stty(1) explains the available options.

When **LPD** opens the device specified by the `lp` capability, it sets the characteristics of the device to those specified with the `ms#` capability. Of particular interest will be the `parenb`, `parodd`, `cs5`, `cs6`, `cs7`, `cs8`, `cstopb`, `crtscts`, and `ixon` modes, which are explained in the stty(1) manual page.

Let us add to our example printer on the sixth serial port. We will set the bps rate to 38400. For the mode, we will set no parity with `-parenb`, 8-bit characters with `cs8`, no modem control with `clocal` and hardware flow control with `crtscts`:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:
```

9.3.1.5.6 Installing the Text Filter

We are now ready to tell **LPD** what text filter to use to send jobs to the printer. A *text filter*, also known as an *input filter*, is a program that **LPD** runs when it has a job to print. When **LPD** runs the text filter for a printer, it sets the filter's standard input to the job to print, and its standard output to the printer device specified with the `lp` capability. The filter is expected to read the job from standard input, perform any necessary translation for the printer, and write the results to standard output, which will get printed. For more information on the text filter, see the Filters section.

For our simple printer setup, the text filter can be a small shell script that just executes `/bin/cat` to send the job to the printer. FreeBSD comes with another filter called `lpf` that handles backspacing and underlining for printers that might not deal with such character streams well. And, of course, you can use any other filter program you want. The filter `lpf` is described in detail in section entitled lpf: a Text Filter.

First, let us make the shell script `/usr/local/libexec/if-simple` be a simple text filter. Put the following text into that file with your favorite text editor:

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout.  Ignores all filter arguments.

/bin/cat && exit 0
exit 2
```

Make the file executable:

```
# chmod 555 /usr/local/libexec/if-simple
```

And then tell LPD to use it by specifying it with the `if` capability in `/etc/printcap`. We will add it to the two printers we have so far in the example `/etc/printcap`:

```
#
#  /etc/printcap for host rose - added text filter
#
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/ttyu5:ms#-parenb cs8 clocal crtscts:\
        :if=/usr/local/libexec/if-simple:
```

Note: A copy of the `if-simple` script can be found in the `/usr/share/examples/printing` directory.

*9.3.1.5.7 Turn on **LPD***

lpd(8) is run from `/etc/rc`, controlled by the `lpd_enable` variable. This variable defaults to `NO`. If you have not done so already, add the line:

```
lpd_enable="YES"
```

to `/etc/rc.conf`, and then either restart your machine, or just run lpd(8).

```
# lpd
```

9.3.1.5.8 Trying It Out

You have reached the end of the simple **LPD** setup. Unfortunately, congratulations are not quite yet in order, since we still have to test the setup and correct any problems. To test the setup, try printing something. To print with the **LPD** system, you use the command lpr(1), which submits a job for printing.

You can combine lpr(1) with the lptest(1) program, introduced in section Checking Printer Communications to generate some test text.

*To test the simple **LPD** setup:*

Type:

```
# lptest 20 5 | lpr -Pprinter-name
```

Where *`printer-name`* is a the name of a printer (or an alias) specified in `/etc/printcap`. To test the default printer, type lpr(1) without any `-P` argument. Again, if you are testing a printer that expects PostScript, send a

PostScript program in that language instead of using lptest(1). You can do so by putting the program in a file and typing `lpr file`.

For a PostScript printer, you should get the results of the program. If you are using lptest(1), then your results should look like the following:

```
!"#$%&'()*+,-./01234
"#$%&'()*+,-./012345
#$%&'()*+,-./0123456
$%&'()*+,-./01234567
%&'()*+,-./012345678
```

To further test the printer, try downloading larger programs (for language-based printers) or running lptest(1) with different arguments. For example, `lptest 80 60` will produce 60 lines of 80 characters each.

If the printer did not work, see the Troubleshooting section.

9.4 Advanced Printer Setup

> **Warning:** As of FreeBSD 8.0, device nodes for serial ports have been renamed from `/dev/ttydN` to `/dev/ttyuN`. FreeBSD 7.X users will have to adapt the following documentation according to these changes.

This section describes filters for printing specially formatted files, header pages, printing across networks, and restricting and accounting for printer usage.

9.4.1 Filters

Although **LPD** handles network protocols, queuing, access control, and other aspects of printing, most of the *real* work happens in the *filters*. Filters are programs that communicate with the printer and handle its device dependencies and special requirements. In the simple printer setup, we installed a plain text filter—an extremely simple one that should work with most printers (section Installing the Text Filter).

However, in order to take advantage of format conversion, printer accounting, specific printer quirks, and so on, you should understand how filters work. It will ultimately be the filter's responsibility to handle these aspects. And the bad news is that most of the time *you* have to provide filters yourself. The good news is that many are generally available; when they are not, they are usually easy to write.

Also, FreeBSD comes with one, `/usr/libexec/lpr/lpf`, that works with many printers that can print plain text. (It handles backspacing and tabs in the file, and does accounting, but that is about all it does.) There are also several filters and filter components in the FreeBSD Ports Collection.

Here is what you will find in this section:

- Section How Filters Work, tries to give an overview of a filter's role in the printing process. You should read this section to get an understanding of what is happening "under the hood" when **LPD** uses filters. This knowledge

could help you anticipate and debug problems you might encounter as you install more and more filters for each of your printers.

- **LPD** expects every printer to be able to print plain text by default. This presents a problem for PostScript printers (or other language-based printers) which cannot directly print plain text. Section Accommodating Plain Text Jobs on PostScript Printers tells you what you should do to overcome this problem. You should read this section if you have a PostScript printer.
- PostScript is a popular output format for many programs. Some people even write PostScript code directly. Unfortunately, PostScript printers are expensive. Section Simulating PostScript on Non PostScript Printers tells how you can further modify a printer's text filter to accept and print PostScript data on a *non PostScript* printer. You should read this section if you do not have a PostScript printer.
- Section Conversion Filters tells about a way you can automate the conversion of specific file formats, such as graphic or typesetting data, into formats your printer can understand. After reading this section, you should be able to set up your printers such that users can type `lpr -t` to print troff data, or `lpr -d` to print TEX DVI data, or `lpr -v` to print raster image data, and so forth. The reading of this section is recommended.
- Section Output Filters tells all about a not often used feature of **LPD**: output filters. Unless you are printing header pages (see Header Pages), you can probably skip that section altogether.
- Section lpf: a Text Filter describes `lpf`, a fairly complete if simple text filter for line printers (and laser printers that act like line printers) that comes with FreeBSD. If you need a quick way to get printer accounting working for plain text, or if you have a printer which emits smoke when it sees backspace characters, you should definitely consider `lpf`.

> **Note:** A copy of the various scripts described below can be found in the `/usr/share/examples/printing` directory.

9.4.1.1 How Filters Work

As mentioned before, a filter is an executable program started by **LPD** to handle the device-dependent part of communicating with the printer.

When **LPD** wants to print a file in a job, it starts a filter program. It sets the filter's standard input to the file to print, its standard output to the printer, and its standard error to the error logging file (specified in the `lf` capability in `/etc/printcap`, or `/dev/console` by default).

Which filter **LPD** starts and the filter's arguments depend on what is listed in the `/etc/printcap` file and what arguments the user specified for the job on the lpr(1) command line. For example, if the user typed `lpr -t`, **LPD** would start the troff filter, listed in the `tf` capability for the destination printer. If the user wanted to print plain text, it would start the `if` filter (this is mostly true: see Output Filters for details).

There are three kinds of filters you can specify in `/etc/printcap`:

- The *text filter*, confusingly called the *input filter* in **LPD** documentation, handles regular text printing. Think of it as the default filter. **LPD** expects every printer to be able to print plain text by default, and it is the text filter's job to make sure backspaces, tabs, or other special characters do not confuse the printer. If you are in an environment where you have to account for printer usage, the text filter must also account for pages printed, usually by counting the number of lines printed and comparing that to the number of lines per page the printer supports. The text filter is started with the following argument list:

`filter-name` [-c] -w *`width`* -l *`length`* -i *`indent`* -n *`login`* -h *`host acct-file`*

where

`-c`

appears if the job is submitted with `lpr -l`

`width`

is the value from the `pw` (page width) capability specified in `/etc/printcap`, default 132

`length`

is the value from the `pl` (page length) capability, default 66

`indent`

is the amount of the indentation from `lpr -i`, default 0

`login`

is the account name of the user printing the file

`host`

is the host name from which the job was submitted

`acct-file`

is the name of the accounting file from the `af` capability.

- A *conversion filter* converts a specific file format into one the printer can render onto paper. For example, ditroff typesetting data cannot be directly printed, but you can install a conversion filter for ditroff files to convert the ditroff data into a form the printer can digest and print. Section Conversion Filters tells all about them. Conversion filters also need to do accounting, if you need printer accounting. Conversion filters are started with the following arguments:

  `filter-name` -x *`pixel-width`* -y *`pixel-height`* -n *`login`* -h *`host acct-file`*

  where *`pixel-width`* is the value from the `px` capability (default 0) and *`pixel-height`* is the value from the `py` capability (default 0).

- The *output filter* is used only if there is no text filter, or if header pages are enabled. In our experience, output filters are rarely used. Section Output Filters describes them. There are only two arguments to an output filter:

  `filter-name` -w *`width`* -l *`length`*

  which are identical to the text filters `-w` and `-l` arguments.

Filters should also *exit* with the following exit status:

exit 0

If the filter printed the file successfully.

exit 1

If the filter failed to print the file but wants **LPD** to try to print the file again. **LPD** will restart a filter if it exits with this status.

exit 2

If the filter failed to print the file and does not want **LPD** to try again. **LPD** will throw out the file.

The text filter that comes with the FreeBSD release, `/usr/libexec/lpr/lpf`, takes advantage of the page width and length arguments to determine when to send a form feed and how to account for printer usage. It uses the login, host, and accounting file arguments to make the accounting entries.

If you are shopping for filters, see if they are LPD-compatible. If they are, they must support the argument lists described above. If you plan on writing filters for general use, then have them support the same argument lists and exit codes.

9.4.1.2 Accommodating Plain Text Jobs on PostScript® Printers

If you are the only user of your computer and PostScript (or other language-based) printer, and you promise to never send plain text to your printer and to never use features of various programs that will want to send plain text to your printer, then you do not need to worry about this section at all.

But, if you would like to send both PostScript and plain text jobs to the printer, then you are urged to augment your printer setup. To do so, we have the text filter detect if the arriving job is plain text or PostScript. All PostScript jobs must start with `%!` (for other printer languages, see your printer documentation). If those are the first two characters in the job, we have PostScript, and can pass the rest of the job directly. If those are not the first two characters in the file, then the filter will convert the text into PostScript and print the result.

How do we do this?

If you have got a serial printer, a great way to do it is to install `lprps`. `lprps` is a PostScript printer filter which performs two-way communication with the printer. It updates the printer's status file with verbose information from the printer, so users and administrators can see exactly what the state of the printer is (such as `toner low` or `paper jam`). But more importantly, it includes a program called `psif` which detects whether the incoming job is plain text and calls `textps` (another program that comes with `lprps`) to convert it to PostScript. It then uses `lprps` to send the job to the printer.

`lprps` is part of the FreeBSD Ports Collection (see The Ports Collection). You can install one of the both `print/lprps-a4` and `print/lprps-letter` ports according to the paper size used. After installing `lprps`, just specify the pathname to the `psif` program that is part of `lprps`. If you installed `lprps` from the Ports Collection, use the following in the serial PostScript printer's entry in `/etc/printcap`:

```
:if=/usr/local/libexec/psif:
```

The `rw` capability should be also included in order to let **LPD** to open the printer in the read-write mode.

If you have a parallel PostScript printer (and therefore cannot use two-way communication with the printer, which `lprps` needs), you can use the following shell script as the text filter:

```
#!/bin/sh
#
#  psif - Print PostScript or plain text on a PostScript printer
#  Script version; NOT the version that comes with lprps
```

```
#  Installed in /usr/local/libexec/psif
#

IFS="" read -r first_line
first_two_chars=‘expr "$first_line" : ’\(..\)’‘

if [ "$first_two_chars" = "%!" ]; then
    #
    #  PostScript job, print it.
    #
    echo "$first_line" && cat && printf "\004" && exit 0
    exit 2
else
    #
    #  Plain text, convert it, then print it.
    #
    ( echo "$first_line"; cat ) | /usr/local/bin/textps && printf "\004" && exit 0
    exit 2
fi
```

In the above script, `textps` is a program we installed separately to convert plain text to PostScript. You can use any text-to-PostScript program you wish. The FreeBSD Ports Collection (see The Ports Collection) includes a full featured text-to-PostScript program called `a2ps` that you might want to investigate.

9.4.1.3 Simulating PostScript on Non PostScript Printers

PostScript is the *de facto* standard for high quality typesetting and printing. PostScript is, however, an *expensive* standard. Thankfully, Aladdin Enterprises has a free PostScript work-alike called **Ghostscript** that runs with FreeBSD. **Ghostscript** can read most PostScript files and can render their pages onto a variety of devices, including many brands of non-PostScript printers. By installing **Ghostscript** and using a special text filter for your printer, you can make your non PostScript printer act like a real PostScript printer.

Ghostscript is in the FreeBSD Ports Collection, many versions are available, the most commonly used version is `print/ghostscript-gpl`.

To simulate PostScript, we have the text filter detect if it is printing a PostScript file. If it is not, then the filter will pass the file directly to the printer; otherwise, it will use **Ghostscript** to first convert the file into a format the printer will understand.

Here is an example: the following script is a text filter for Hewlett Packard DeskJet 500 printers. For other printers, substitute the `-sDEVICE` argument to the `gs` (**Ghostscript**) command. (Type `gs -h` to get a list of devices the current installation of **Ghostscript** supports.)

```
#!/bin/sh
#
#  ifhp - Print Ghostscript-simulated PostScript on a DeskJet 500
#  Installed in /usr/local/libexec/ifhp

#
#  Treat LF as CR+LF (to avoid the "staircase effect" on HP/PCL
#  printers):
#
printf "\033&k2G" || exit 2
```

```
#
#  Read first two characters of the file
#
IFS="" read -r first_line
first_two_chars=‘expr "$first_line" : ’\(..\)’‘

if [ "$first_two_chars" = "%!" ]; then
    #
    #  It is PostScript; use Ghostscript to scan-convert and print it.
    #
    /usr/local/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=djet500 \
      -sOutputFile=- - && exit 0
else
    #
    #  Plain text or HP/PCL, so just print it directly; print a form feed
    #  at the end to eject the last page.
    #
    echo "$first_line" && cat && printf "\033&l0H" &&
exit 0
fi

exit 2
```

Finally, you need to notify **LPD** of the filter via the `if` capability:

```
:if=/usr/local/libexec/ifhp:
```

That is it. You can type `lpr` *`plain.text`* and `lpr` *`whatever.ps`* and both should print successfully.

9.4.1.4 Conversion Filters

After completing the simple setup described in Simple Printer Setup, the first thing you will probably want to do is install conversion filters for your favorite file formats (besides plain ASCII text).

9.4.1.4.1 Why Install Conversion Filters?

Conversion filters make printing various kinds of files easy. As an example, suppose we do a lot of work with the TEX typesetting system, and we have a PostScript printer. Every time we generate a DVI file from TEX, we cannot print it directly until we convert the DVI file into PostScript. The command sequence goes like this:

```
% dvips seaweed-analysis.dvi
% lpr seaweed-analysis.ps
```

By installing a conversion filter for DVI files, we can skip the hand conversion step each time by having **LPD** do it for us. Now, each time we get a DVI file, we are just one step away from printing it:

```
% lpr -d seaweed-analysis.dvi
```

We got **LPD** to do the DVI file conversion for us by specifying the `-d` option. Section Formatting and Conversion Options lists the conversion options.

For each of the conversion options you want a printer to support, install a *conversion filter* and specify its pathname in `/etc/printcap`. A conversion filter is like the text filter for the simple printer setup (see section Installing the Text Filter) except that instead of printing plain text, the filter converts the file into a format the printer can understand.

9.4.1.4.2 Which Conversion Filters Should I Install?

You should install the conversion filters you expect to use. If you print a lot of DVI data, then a DVI conversion filter is in order. If you have got plenty of troff to print out, then you probably want a troff filter.

The following table summarizes the filters that **LPD** works with, their capability entries for the `/etc/printcap` file, and how to invoke them with the `lpr` command:

File type	`/etc/printcap` capability	`lpr` option
cifplot	`cf`	`-c`
DVI	`df`	`-d`
plot	`gf`	`-g`
ditroff	`nf`	`-n`
FORTRAN text	`rf`	`-f`
troff	`tf`	`-f`
raster	`vf`	`-v`
plain text	`if`	none, `-p`, or `-l`

In our example, using `lpr -d` means the printer needs a `df` capability in its entry in `/etc/printcap`.

Despite what others might contend, formats like FORTRAN text and plot are probably obsolete. At your site, you can give new meanings to these or any of the formatting options just by installing custom filters. For example, suppose you would like to directly print Printerleaf files (files from the Interleaf desktop publishing program), but will never print plot files. You could install a Printerleaf conversion filter under the `gf` capability and then educate your users that `lpr -g` mean “print Printerleaf files.”

9.4.1.4.3 Installing Conversion Filters

Since conversion filters are programs you install outside of the base FreeBSD installation, they should probably go under `/usr/local`. The directory `/usr/local/libexec` is a popular location, since they are specialized programs that only **LPD** will run; regular users should not ever need to run them.

To enable a conversion filter, specify its pathname under the appropriate capability for the destination printer in `/etc/printcap`.

In our example, we will add the DVI conversion filter to the entry for the printer named `bamboo`. Here is the example `/etc/printcap` file again, with the new `df` capability for the printer `bamboo`:

```
#
#  /etc/printcap for host rose - added df filter for bamboo
#
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/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

The DVI filter is a shell script named `/usr/local/libexec/psdf`. Here is that script:

```
#!/bin/sh
#
#  psdf - DVI to PostScript printer filter
#  Installed in /usr/local/libexec/psdf
#
# Invoked by lpd when user runs lpr -d
#
exec /usr/local/bin/dvips -f | /usr/local/libexec/lprps "$@"
```

This script runs `dvips` in filter mode (the `-f` argument) on standard input, which is the job to print. It then starts the PostScript printer filter `lprps` (see section Accommodating Plain Text Jobs on PostScript Printers) with the arguments **LPD** passed to this script. The `lprps` utility will use those arguments to account for the pages printed.

9.4.1.4.4 More Conversion Filter Examples

There is no fixed set of steps to install conversion filters, some working examples are described in this section. Use these as guidance to making your own filters. Use them directly, if appropriate.

This example script is a raster (well, GIF file, actually) conversion filter for a Hewlett Packard LaserJet III-Si printer:

```
#!/bin/sh
#
#  hpvf - Convert GIF files into HP/PCL, then print
#  Installed in /usr/local/libexec/hpvf

PATH=/usr/X11R6/bin:$PATH; export PATH
giftopnm | ppmtopgm | pgmtopbm | pbmtolj -resolution 300 \
    && exit 0 \
    || exit 2
```

It works by converting the GIF file into a portable anymap, converting that into a portable graymap, converting that into a portable bitmap, and converting that into LaserJet/PCL-compatible data.

Here is the `/etc/printcap` file with an entry for a printer using the above filter:

```
#
#  /etc/printcap for host 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:
```

The following script is a conversion filter for troff data from the groff typesetting system for the PostScript printer named `bamboo`:

```
#!/bin/sh
#
#  pstf - Convert groff's troff data into PS, then print.
#  Installed in /usr/local/libexec/pstf
#
exec grops | /usr/local/libexec/lprps "$@"
```

The above script makes use of `lprps` again to handle the communication with the printer. If the printer were on a parallel port, we would use this script instead:

```
#!/bin/sh
#
#  pstf - Convert groff's troff data into PS, then print.
#  Installed in /usr/local/libexec/pstf
#
exec grops
```

That is it. Here is the entry we need to add to `/etc/printcap` to enable the filter:

```
:tf=/usr/local/libexec/pstf:
```

Here is an example that might make old hands at FORTRAN blush. It is a FORTRAN-text filter for any printer that can directly print plain text. We will install it for the printer `teak`:

```
#!/bin/sh
#
# hprf - FORTRAN text filter for LaserJet 3si:
# Installed in /usr/local/libexec/hprf
#

printf "\033&k2G" && fpr && printf "\033&l0H" &&
 exit 0
exit 2
```

And we will add this line to the `/etc/printcap` for the printer `teak` to enable this filter:

```
:rf=/usr/local/libexec/hprf:
```

Here is one final, somewhat complex example. We will add a DVI filter to the LaserJet printer `teak` introduced earlier. First, the easy part: updating `/etc/printcap` with the location of the DVI filter:

```
:df=/usr/local/libexec/hpdf:
```

Now, for the hard part: making the filter. For that, we need a DVI-to-LaserJet/PCL conversion program. The FreeBSD Ports Collection (see The Ports Collection) has one: `print/dvi2xx`. Installing this port gives us the program we need, `dvilj2p`, which converts DVI into LaserJet IIp, LaserJet III, and LaserJet 2000 compatible codes.

The `dvilj2p` utility makes the filter `hpdf` quite complex since `dvilj2p` cannot read from standard input. It wants to work with a filename. What is worse, the filename has to end in `.dvi` so using `/dev/fd/0` for standard input is problematic. We can get around that problem by linking (symbolically) a temporary file name (one that ends in `.dvi`) to `/dev/fd/0`, thereby forcing `dvilj2p` to read from standard input.

The only other fly in the ointment is the fact that we cannot use `/tmp` for the temporary link. Symbolic links are owned by user and group `bin`. The filter runs as user `daemon`. And the `/tmp` directory has the sticky bit set. The filter can create the link, but it will not be able clean up when done and remove it since the link will belong to a different user.

Instead, the filter will make the symbolic link in the current working directory, which is the spooling directory (specified by the `sd` capability in `/etc/printcap`). This is a perfect place for filters to do their work, especially since there is (sometimes) more free disk space in the spooling directory than under `/tmp`.

Here, finally, is the filter:

```
#!/bin/sh
#
#  hpdf - Print DVI data on HP/PCL printer
#  Installed in /usr/local/libexec/hpdf

PATH=/usr/local/bin:$PATH; export PATH

#
#  Define a function to clean up our temporary files.  These exist
#  in the current directory, which will be the spooling directory
#  for the printer.
#
cleanup() {
   rm -f hpdf$$.dvi
}

#
#  Define a function to handle fatal errors: print the given message
#  and exit 2.  Exiting with 2 tells LPD to do not try to reprint the
#  job.
#
fatal() {
    echo "$@" 1>&2
    cleanup
    exit 2
}

#
#  If user removes the job, LPD will send SIGINT, so trap SIGINT
#  (and a few other signals) to clean up after ourselves.
#
trap cleanup 1 2 15

#
#  Make sure we are not colliding with any existing files.
#
cleanup

#
#  Link the DVI input file to standard input (the file to print).
#
ln -s /dev/fd/0 hpdf$$.dvi || fatal "Cannot symlink /dev/fd/0"
```

```
#
#  Make LF = CR+LF
#
printf "\033&k2G" || fatal "Cannot initialize printer"

#
#  Convert and print.  Return value from dvilj2p does not seem to be
#  reliable, so we ignore it.
#
dvilj2p -M1 -q -e- dfhp$$.dvi

#
#  Clean up and exit
#
cleanup
exit 0
```

9.4.1.4.5 Automated Conversion: an Alternative to Conversion Filters

All these conversion filters accomplish a lot for your printing environment, but at the cost forcing the user to specify (on the lpr(1) command line) which one to use. If your users are not particularly computer literate, having to specify a filter option will become annoying. What is worse, though, is that an incorrectly specified filter option may run a filter on the wrong type of file and cause your printer to spew out hundreds of sheets of paper.

Rather than install conversion filters at all, you might want to try having the text filter (since it is the default filter) detect the type of file it has been asked to print and then automatically run the right conversion filter. Tools such as `file` can be of help here. Of course, it will be hard to determine the differences between *some* file types—and, of course, you can still provide conversion filters just for them.

The FreeBSD Ports Collection has a text filter that performs automatic conversion called `apsfilter` (`print/apsfilter`). It can detect plain text, PostScript, DVI and almost any kind of files, run the proper conversions, and print.

9.4.1.5 Output Filters

The **LPD** spooling system supports one other type of filter that we have not yet explored: an output filter. An output filter is intended for printing plain text only, like the text filter, but with many simplifications. If you are using an output filter but no text filter, then:

- **LPD** starts an output filter once for the entire job instead of once for each file in the job.
- **LPD** does not make any provision to identify the start or the end of files within the job for the output filter.
- **LPD** does not pass the user's login or host to the filter, so it is not intended to do accounting. In fact, it gets only two arguments:

  `filter-name -w`*`width`* `-l`*`length`*

  Where *`width`* is from the `pw` capability and *`length`* is from the `pl` capability for the printer in question.

Do not be seduced by an output filter's simplicity. If you would like each file in a job to start on a different page an output filter *will not work*. Use a text filter (also known as an input filter); see section Installing the Text Filter. Furthermore, an output filter is actually *more complex* in that it has to examine the byte stream being sent to it for special flag characters and must send signals to itself on behalf of **LPD**.

However, an output filter is *necessary* if you want header pages and need to send escape sequences or other initialization strings to be able to print the header page. (But it is also *futile* if you want to charge header pages to the requesting user's account, since **LPD** does not give any user or host information to the output filter.)

On a single printer, **LPD** allows both an output filter and text or other filters. In such cases, **LPD** will start the output filter to print the header page (see section Header Pages) only. **LPD** then expects the output filter to *stop itself* by sending two bytes to the filter: ASCII 031 followed by ASCII 001. When an output filter sees these two bytes (031, 001), it should stop by sending `SIGSTOP` to itself. When **LPD**'s done running other filters, it will restart the output filter by sending `SIGCONT` to it.

If there is an output filter but *no* text filter and **LPD** is working on a plain text job, **LPD** uses the output filter to do the job. As stated before, the output filter will print each file of the job in sequence with no intervening form feeds or other paper advancement, and this is probably *not* what you want. In almost all cases, you need a text filter.

The program `lpf`, which we introduced earlier as a text filter, can also run as an output filter. If you need a quick-and-dirty output filter but do not want to write the byte detection and signal sending code, try `lpf`. You can also wrap `lpf` in a shell script to handle any initialization codes the printer might require.

9.4.1.6 `lpf`: a Text Filter

The program `/usr/libexec/lpr/lpf` that comes with FreeBSD binary distribution is a text filter (input filter) that can indent output (job submitted with `lpr -i`), allow literal characters to pass (job submitted with `lpr -l`), adjust the printing position for backspaces and tabs in the job, and account for pages printed. It can also act like an output filter.

The `lpf` filter is suitable for many printing environments. And although it has no capability to send initialization sequences to a printer, it is easy to write a shell script to do the needed initialization and then execute `lpf`.

In order for `lpf` to do page accounting correctly, it needs correct values filled in for the `pw` and `pl` capabilities in the `/etc/printcap` file. It uses these values to determine how much text can fit on a page and how many pages were in a user's job. For more information on printer accounting, see Accounting for Printer Usage.

9.4.2 Header Pages

If you have *lots* of users, all of them using various printers, then you probably want to consider *header pages* as a necessary evil.

Header pages, also known as *banner* or *burst pages* identify to whom jobs belong after they are printed. They are usually printed in large, bold letters, perhaps with decorative borders, so that in a stack of printouts they stand out from the real documents that comprise users' jobs. They enable users to locate their jobs quickly. The obvious drawback to a header page is that it is yet one more sheet that has to be printed for every job, their ephemeral usefulness lasting not more than a few minutes, ultimately finding themselves in a recycling bin or rubbish heap. (Note that header pages go with each job, not each file in a job, so the paper waste might not be that bad.)

The **LPD** system can provide header pages automatically for your printouts *if* your printer can directly print plain text. If you have a PostScript printer, you will need an external program to generate the header page; see Header Pages on PostScript Printers.

9.4.2.1 Enabling Header Pages

In the Simple Printer Setup section, we turned off header pages by specifying `sh` (meaning "suppress header") in the `/etc/printcap` file. To enable header pages for a printer, just remove the `sh` capability.

Sounds too easy, right?

You are right. You *might* have to provide an output filter to send initialization strings to the printer. Here is an example output filter for Hewlett Packard PCL-compatible printers:

```
#!/bin/sh
#
#  hpof - Output filter for Hewlett Packard PCL-compatible printers
#  Installed in /usr/local/libexec/hpof

printf "\033&k2G" || exit 2
exec /usr/libexec/lpr/lpf
```

Specify the path to the output filter in the `of` capability. See the Output Filters section for more information.

Here is an example `/etc/printcap` file for the printer `teak` that we introduced earlier; we enabled header pages and added the above output filter:

```
#
#  /etc/printcap for host 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:
```

Now, when users print jobs to `teak`, they get a header page with each job. If users want to spend time searching for their printouts, they can suppress header pages by submitting the job with `lpr -h`; see the Header Page Options section for more lpr(1) options.

> **Note: LPD** prints a form feed character after the header page. If your printer uses a different character or sequence of characters to eject a page, specify them with the `ff` capability in `/etc/printcap`.

9.4.2.2 Controlling Header Pages

By enabling header pages, **LPD** will produce a *long header*, a full page of large letters identifying the user, host, and job. Here is an example (`kelly` printed the job named "outline" from host `rose`):

```
      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      eeeeee      l        l     y    y
kk k     e           l        l     y    y
k   k    e    e      l        l     y   yy
k    k    eeee      lll      lll     yyy y
                                        y
                                       y    y
                                        yyyy


                            ll
                    t        l        i
                    t        l
 oooo    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   eeeeee
o    o   u    u     t        l        i     n    n   e
o    o   u   uu     t  t     l        i     n    n   e    e
 oooo     uuu u      tt      lll      iii    n    n    eeee
```

```
r rrr     oooo     ssss     eeee
rr   r   o    o   s    s   e    e
r        o    o    ss      eeeeee
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
```

LPD appends a form feed after this text so the job starts on a new page (unless you have `sf` (suppress form feeds) in the destination printer's entry in `/etc/printcap`).

If you prefer, **LPD** can make a *short header*; specify `sb` (short banner) in the `/etc/printcap` file. The header page will look like this:

```
rose:kelly  Job: outline  Date: Sun Sep 17 11:07:51 1995
```

Also by default, **LPD** prints the header page first, then the job. To reverse that, specify `hl` (header last) in `/etc/printcap`.

9.4.2.3 Accounting for Header Pages

Using **LPD**'s built-in header pages enforces a particular paradigm when it comes to printer accounting: header pages must be *free of charge*.

Why?

Because the output filter is the only external program that will have control when the header page is printed that could do accounting, and it is not provided with any *user or host* information or an accounting file, so it has no idea whom to charge for printer use. It is also not enough to just "increase the page count by one" by modifying the text filter or any of the conversion filters (which do have user and host information) since users can suppress header pages with `lpr -h`. They could still be charged for header pages they did not print. Basically, `lpr -h` will be the preferred option of environmentally-minded users, but you cannot offer any incentive to use it.

It is *still not enough* to have each of the filters generate their own header pages (thereby being able to charge for them). If users wanted the option of suppressing the header pages with `lpr -h`, they will still get them and be charged for them since **LPD** does not pass any knowledge of the `-h` option to any of the filters.

So, what are your options?

You can:

- Accept **LPD**'s paradigm and make header pages free.
- Install an alternative to **LPD**, such as **LPRng**. Section Alternatives to the Standard Spooler tells more about other spooling software you can substitute for **LPD**.
- Write a *smart* output filter. Normally, an output filter is not meant to do anything more than initialize a printer or do some simple character conversion. It is suited for header pages and plain text jobs (when there is no text (input) filter). But, if there is a text filter for the plain text jobs, then **LPD** will start the output filter only for the header pages. And the output filter can parse the header page text that **LPD** generates to determine what user and host to charge for the header page. The only other problem with this method is that the output filter still does not know what accounting file to use (it is not passed the name of the file from the `af` capability), but if you have a well-known accounting file, you can hard-code that into the output filter. To facilitate the parsing step, use the `sh` (short header) capability in `/etc/printcap`. Then again, all that might be too much trouble, and users will certainly appreciate the more generous system administrator who makes header pages free.

9.4.2.4 Header Pages on PostScript Printers

As described above, **LPD** can generate a plain text header page suitable for many printers. Of course, PostScript cannot directly print plain text, so the header page feature of **LPD** is useless—or mostly so.

One obvious way to get header pages is to have every conversion filter and the text filter generate the header page. The filters should use the user and host arguments to generate a suitable header page. The drawback of this method is that users will always get a header page, even if they submit jobs with `lpr -h`.

Let us explore this method. The following script takes three arguments (user login name, host name, and job name) and makes a simple PostScript header page:

```
#!/bin/sh
```

```
#
#  make-ps-header - make a PostScript header page on stdout
#  Installed in /usr/local/libexec/make-ps-header
#

#
#  These are PostScript units (72 to the inch).  Modify for A4 or
#  whatever size paper you are using:
#
page_width=612
page_height=792
border=72

#
#  Check arguments
#
if [ $# -ne 3 ]; then
    echo "Usage: ‘basename $0‘ <user> <host> <job>" 1>&2
    exit 1
fi

#
#  Save these, mostly for readability in the PostScript, below.
#
user=$1
host=$2
job=$3
date=‘date‘

#
#  Send the PostScript code to stdout.
#
exec cat <<EOF
%!PS

%
%  Make sure we do not interfere with user’s job that will follow
%
save

%
%  Make a thick, unpleasant border around the edge of the paper.
%
$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

%
%  Display user’s login name, nice and large and prominent
%
```

```
/Helvetica-Bold findfont 64 scalefont setfont
$page_width ($user) stringwidth pop sub 2 div $page_height 200 sub moveto
($user) show

%
%  Now show the boring particulars
%
/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

%
% That is it
%
restore
showpage
EOF
```

Now, each of the conversion filters and the text filter can call this script to first generate the header page, and then print the user's job. Here is the DVI conversion filter from earlier in this document, modified to make a header page:

```
#!/bin/sh
#
#  psdf - DVI to PostScript printer filter
#  Installed in /usr/local/libexec/psdf
#
#  Invoked by lpd when user runs lpr -d
#

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
```

Notice how the filter has to parse the argument list in order to determine the user and host name. The parsing for the other conversion filters is identical. The text filter takes a slightly different set of arguments, though (see section How Filters Work).

As we have mentioned before, the above scheme, though fairly simple, disables the “suppress header page” option (the `-h` option) to `lpr`. If users wanted to save a tree (or a few pennies, if you charge for header pages), they would not be able to do so, since every filter’s going to print a header page with every job.

To allow users to shut off header pages on a per-job basis, you will need to use the trick introduced in section Accounting for Header Pages: write an output filter that parses the LPD-generated header page and produces a PostScript version. If the user submits the job with `lpr -h`, then **LPD** will not generate a header page, and neither will your output filter. Otherwise, your output filter will read the text from **LPD** and send the appropriate header page PostScript code to the printer.

If you have a PostScript printer on a serial line, you can make use of `lprps`, which comes with an output filter, `psof`, which does the above. Note that `psof` does not charge for header pages.

9.4.3 Networked Printing

FreeBSD supports networked printing: sending jobs to remote printers. Networked printing generally refers to two different things:

- Accessing a printer attached to a remote host. You install a printer that has a conventional serial or parallel interface on one host. Then, you set up **LPD** to enable access to the printer from other hosts on the network. Section Printers Installed on Remote Hosts tells how to do this.
- Accessing a printer attached directly to a network. The printer has a network interface in addition to (or in place of) a more conventional serial or parallel interface. Such a printer might work as follows:
  - It might understand the **LPD** protocol and can even queue jobs from remote hosts. In this case, it acts just like a regular host running **LPD**. Follow the same procedure in section Printers Installed on Remote Hosts to set up such a printer.
  - It might support a data stream network connection. In this case, you “attach” the printer to one host on the network by making that host responsible for spooling jobs and sending them to the printer. Section Printers with Networked Data Stream Interfaces gives some suggestions on installing such printers.

9.4.3.1 Printers Installed on Remote Hosts

The **LPD** spooling system has built-in support for sending jobs to other hosts also running **LPD** (or are compatible with **LPD**). This feature enables you to install a printer on one host and make it accessible from other hosts. It also works with printers that have network interfaces that understand the **LPD** protocol.

To enable this kind of remote printing, first install a printer on one host, the *printer host*, using the simple printer setup described in the Simple Printer Setup section. Do any advanced setup in Advanced Printer Setup that you need. Make sure to test the printer and see if it works with the features of **LPD** you have enabled. Also ensure that the *local host* has authorization to use the **LPD** service in the *remote host* (see Restricting Jobs from Remote Hosts).

If you are using a printer with a network interface that is compatible with **LPD**, then the *printer host* in the discussion below is the printer itself, and the *printer name* is the name you configured for the printer. See the documentation that accompanied your printer and/or printer-network interface.

> **Tip:** If you are using a Hewlett Packard Laserjet then the printer name `text` will automatically perform the LF to CRLF conversion for you, so you will not require the `hpif` script.

Then, on the other hosts you want to have access to the printer, make an entry in their `/etc/printcap` files with the following:

1. Name the entry anything you want. For simplicity, though, you probably want to use the same name and aliases as on the printer host.
2. Leave the `lp` capability blank, explicitly (`:lp=:`).
3. Make a spooling directory and specify its location in the `sd` capability. **LPD** will store jobs here before they get sent to the printer host.
4. Place the name of the printer host in the `rm` capability.
5. Place the printer name on the *printer host* in the `rp` capability.

That is it. You do not need to list conversion filters, page dimensions, or anything else in the `/etc/printcap` file.

Here is an example. The host `rose` has two printers, `bamboo` and `rattan`. We will enable users on the host `orchid` to print to those printers. Here is the `/etc/printcap` file for `orchid` (back from section Enabling Header Pages). It already had the entry for the printer `teak`; we have added entries for the two printers on the host `rose`:

```
#
#  /etc/printcap for host orchid - added (remote) printers on rose
#

#
#  teak is local; it is connected directly to orchid:
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/ifhp:\
        :vf=/usr/local/libexec/vfhp:\
        :of=/usr/local/libexec/ofhp:

#
#  rattan is connected to rose; send jobs for rattan to rose:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

#
```

```
#  bamboo is connected to rose as well:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:
```

Then, we just need to make spooling directories on `orchid`:

```
# 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
```

Now, users on `orchid` can print to `rattan` and `bamboo`. If, for example, a user on `orchid` typed:

```
% lpr -P bamboo -d sushi-review.dvi
```

the **LPD** system on `orchid` would copy the job to the spooling directory `/var/spool/lpd/bamboo` and note that it was a DVI job. As soon as the host `rose` has room in its `bamboo` spooling directory, the two **LPD**s would transfer the file to `rose`. The file would wait in `rose`'s queue until it was finally printed. It would be converted from DVI to PostScript (since `bamboo` is a PostScript printer) on `rose`.

9.4.3.2 Printers with Networked Data Stream Interfaces

Often, when you buy a network interface card for a printer, you can get two versions: one which emulates a spooler (the more expensive version), or one which just lets you send data to it as if you were using a serial or parallel port (the cheaper version). This section tells how to use the cheaper version. For the more expensive one, see the previous section Printers Installed on Remote Hosts.

The format of the `/etc/printcap` file lets you specify what serial or parallel interface to use, and (if you are using a serial interface), what baud rate, whether to use flow control, delays for tabs, conversion of newlines, and more. But there is no way to specify a connection to a printer that is listening on a TCP/IP or other network port.

To send data to a networked printer, you need to develop a communications program that can be called by the text and conversion filters. Here is one such example: the script `netprint` takes all data on standard input and sends it to a network-attached printer. We specify the hostname of the printer as the first argument and the port number to which to connect as the second argument to `netprint`. Note that this supports one-way communication only (FreeBSD to printer); many network printers support two-way communication, and you might want to take advantage of that (to get printer status, perform accounting, etc.).

```
#!/usr/bin/perl
#
#  netprint - Text filter for printer attached to network
#  Installed in /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;
```

We can then use this script in various filters. Suppose we had a Diablo 750-N line printer connected to the network. The printer accepts data to print on port number 5100. The host name of the printer is `scrivener`. Here is the text filter for the printer:

```
#!/bin/sh
#
#  diablo-if-net - Text filter for Diablo printer 'scrivener' listening
#  on port 5100.   Installed in /usr/local/libexec/diablo-if-net
#
exec /usr/libexec/lpr/lpf "$@" | /usr/local/libexec/netprint scrivener 5100
```

9.4.4 Restricting Printer Usage

This section gives information on restricting printer usage. The **LPD** system lets you control who can access a printer, both locally or remotely, whether they can print multiple copies, how large their jobs can be, and how large the printer queues can get.

9.4.4.1 Restricting Multiple Copies

The **LPD** system makes it easy for users to print multiple copies of a file. Users can print jobs with `lpr -#5` (for example) and get five copies of each file in the job. Whether this is a good thing is up to you.

If you feel multiple copies cause unnecessary wear and tear on your printers, you can disable the `-#` option to lpr(1) by adding the `sc` capability to the `/etc/printcap` file. When users submit jobs with the `-#` option, they will see:

```
lpr: multiple copies are not allowed
```

Note that if you have set up access to a printer remotely (see section Printers Installed on Remote Hosts), you need the `sc` capability on the remote `/etc/printcap` files as well, or else users will still be able to submit multiple-copy jobs by using another host.

Here is an example. This is the `/etc/printcap` file for the host `rose`. The printer `rattan` is quite hearty, so we will allow multiple copies, but the laser printer `bamboo` is a bit more delicate, so we will disable multiple copies by adding the `sc` capability:

```
#
#  /etc/printcap for host rose - restrict multiple copies on bamboo
#
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/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Now, we also need to add the `sc` capability on the host `orchid`'s `/etc/printcap` (and while we are at it, let us disable multiple copies for the printer `teak`):

```
#
#  /etc/printcap for host orchid - no multiple copies for local
#  printer teak or remote printer bamboo
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:
```

By using the `sc` capability, we prevent the use of `lpr -#`, but that still does not prevent users from running lpr(1) multiple times, or from submitting the same file multiple times in one job like this:

```
% lpr forsale.sign forsale.sign forsale.sign forsale.sign forsale.sign
```

There are many ways to prevent this abuse (including ignoring it) which you are free to explore.

9.4.4.2 Restricting Access to Printers

You can control who can print to what printers by using the UNIX group mechanism and the `rg` capability in `/etc/printcap`. Just place the users you want to have access to a printer in a certain group, and then name that group in the `rg` capability.

If users outside the group (including `root`) try to print to the controlled printer then they will be greeted with the following message:

```
lpr: Not a member of the restricted group
```

As with the `sc` (suppress multiple copies) capability, you need to specify `rg` on remote hosts that also have access to your printers, if you feel it is appropriate (see section Printers Installed on Remote Hosts).

For example, we will let anyone access the printer `rattan`, but only those in group `artists` can use `bamboo`. Here is the familiar `/etc/printcap` for host `rose`:

```
#
#  /etc/printcap for host rose - restricted group for bamboo
#
```

```
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/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Let us leave the other example `/etc/printcap` file (for the host `orchid`) alone. Of course, anyone on `orchid` can print to `bamboo`. It might be the case that we only allow certain logins on `orchid` anyway, and want them to have access to the printer. Or not.

> **Note:** There can be only one restricted group per printer.

9.4.4.3 Controlling Sizes of Jobs Submitted

If you have many users accessing the printers, you probably need to put an upper limit on the sizes of the files users can submit to print. After all, there is only so much free space on the filesystem that houses the spooling directories, and you also need to make sure there is room for the jobs of other users.

LPD enables you to limit the maximum byte size a file in a job can be with the `mx` capability. The units are in `BUFSIZ` blocks, which are 1024 bytes. If you put a zero for this capability, there will be no limit on file size; however, if no `mx` capability is specified, then a default limit of 1000 blocks will be used.

> **Note:** The limit applies to *files* in a job, and *not* the total job size.

LPD will not refuse a file that is larger than the limit you place on a printer. Instead, it will queue as much of the file up to the limit, which will then get printed. The rest will be discarded. Whether this is correct behavior is up for debate.

Let us add limits to our example printers `rattan` and `bamboo`. Since those `artists`' PostScript files tend to be large, we will limit them to five megabytes. We will put no limit on the plain text line printer:

```
#
#  /etc/printcap for host rose
#

#
#  No limit on job size:
#
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:
```

```
#
#  Limit of five megabytes:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Again, the limits apply to the local users only. If you have set up access to your printers remotely, remote users will not get those limits. You will need to specify the `mx` capability in the remote `/etc/printcap` files as well. See section Printers Installed on Remote Hosts for more information on remote printing.

There is another specialized way to limit job sizes from remote printers; see section Restricting Jobs from Remote Hosts.

9.4.4.4 Restricting Jobs from Remote Hosts

The **LPD** spooling system provides several ways to restrict print jobs submitted from remote hosts:

Host restrictions

You can control from which remote hosts a local **LPD** accepts requests with the files `/etc/hosts.equiv` and `/etc/hosts.lpd`. **LPD** checks to see if an incoming request is from a host listed in either one of these files. If not, **LPD** refuses the request.

The format of these files is simple: one host name per line. Note that the file `/etc/hosts.equiv` is also used by the ruserok(3) protocol, and affects programs like rsh(1) and rcp(1), so be careful.

For example, here is the `/etc/hosts.lpd` file on the host `rose`:

```
orchid
violet
madrigal.fishbaum.de
```

This means `rose` will accept requests from the hosts `orchid`, `violet`, and `madrigal.fishbaum.de`. If any other host tries to access `rose`'s **LPD**, the job will be refused.

Size restrictions

You can control how much free space there needs to remain on the filesystem where a spooling directory resides. Make a file called `minfree` in the spooling directory for the local printer. Insert in that file a number representing how many disk blocks (512 bytes) of free space there has to be for a remote job to be accepted.

This lets you insure that remote users will not fill your filesystem. You can also use it to give a certain priority to local users: they will be able to queue jobs long after the free disk space has fallen below the amount specified in the `minfree` file.

For example, let us add a `minfree` file for the printer `bamboo`. We examine `/etc/printcap` to find the spooling directory for this printer; here is `bamboo`'s entry:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:mx#5000:\
        :if=/usr/local/libexec/psif:\
```

```
:df=/usr/local/libexec/psdf:
```

The spooling directory is given in the `sd` capability. We will make three megabytes (which is 6144 disk blocks) the amount of free disk space that must exist on the filesystem for **LPD** to accept remote jobs:

```
# echo 6144 > /var/spool/lpd/bamboo/minfree
```

User restrictions

You can control which remote users can print to local printers by specifying the `rs` capability in `/etc/printcap`. When `rs` appears in the entry for a locally-attached printer, **LPD** will accept jobs from remote hosts *if* the user submitting the job also has an account of the same login name on the local host. Otherwise, **LPD** refuses the job.

This capability is particularly useful in an environment where there are (for example) different departments sharing a network, and some users transcend departmental boundaries. By giving them accounts on your systems, they can use your printers from their own departmental systems. If you would rather allow them to use *only* your printers and not your computer resources, you can give them "token" accounts, with no home directory and a useless shell like `/usr/bin/false`.

9.4.5 Accounting for Printer Usage

So, you need to charge for printouts. And why not? Paper and ink cost money. And then there are maintenance costs—printers are loaded with moving parts and tend to break down. You have examined your printers, usage patterns, and maintenance fees and have come up with a per-page (or per-foot, per-meter, or per-whatever) cost. Now, how do you actually start accounting for printouts?

Well, the bad news is the **LPD** spooling system does not provide much help in this department. Accounting is highly dependent on the kind of printer in use, the formats being printed, and *your* requirements in charging for printer usage.

To implement accounting, you have to modify a printer's text filter (to charge for plain text jobs) and the conversion filters (to charge for other file formats), to count pages or query the printer for pages printed. You cannot get away with using the simple output filter, since it cannot do accounting. See section Filters.

Generally, there are two ways to do accounting:

- *Periodic accounting* is the more common way, possibly because it is easier. Whenever someone prints a job, the filter logs the user, host, and number of pages to an accounting file. Every month, semester, year, or whatever time period you prefer, you collect the accounting files for the various printers, tally up the pages printed by users, and charge for usage. Then you truncate all the logging files, starting with a clean slate for the next period.
- *Timely accounting* is less common, probably because it is more difficult. This method has the filters charge users for printouts as soon as they use the printers. Like disk quotas, the accounting is immediate. You can prevent users from printing when their account goes in the red, and might provide a way for users to check and adjust their "print quotas". But this method requires some database code to track users and their quotas.

The **LPD** spooling system supports both methods easily: since you have to provide the filters (well, most of the time), you also have to provide the accounting code. But there is a bright side: you have enormous flexibility in your accounting methods. For example, you choose whether to use periodic or timely accounting. You choose what information to log: user names, host names, job types, pages printed, square footage of paper used, how long the job took to print, and so forth. And you do so by modifying the filters to save this information.

9.4.5.1 Quick and Dirty Printer Accounting

FreeBSD comes with two programs that can get you set up with simple periodic accounting right away. They are the text filter `lpf`, described in section lpf: a Text Filter, and pac(8), a program to gather and total entries from printer accounting files.

As mentioned in the section on filters (Filters), **LPD** starts the text and the conversion filters with the name of the accounting file to use on the filter command line. The filters can use this argument to know where to write an accounting file entry. The name of this file comes from the `af` capability in `/etc/printcap`, and if not specified as an absolute path, is relative to the spooling directory.

LPD starts `lpf` with page width and length arguments (from the `pw` and `pl` capabilities). The `lpf` filter uses these arguments to determine how much paper will be used. After sending the file to the printer, it then writes an accounting entry in the accounting file. The entries look like this:

```
2.00 rose:andy
3.00 rose:kelly
3.00 orchid:mary
5.00 orchid:mary
2.00 orchid:zhang
```

You should use a separate accounting file for each printer, as `lpf` has no file locking logic built into it, and two `lpf`s might corrupt each other's entries if they were to write to the same file at the same time. An easy way to insure a separate accounting file for each printer is to use `af=acct` in `/etc/printcap`. Then, each accounting file will be in the spooling directory for a printer, in a file named `acct`.

When you are ready to charge users for printouts, run the pac(8) program. Just change to the spooling directory for the printer you want to collect on and type `pac`. You will get a dollar-centric summary like the following:

```
  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
```

These are the arguments pac(8) expects:

`-P`*`printer`*

> Which *`printer`* to summarize. This option works only if there is an absolute path in the `af` capability in `/etc/printcap`.

`-c`

> Sort the output by cost instead of alphabetically by user name.

`-m`

> Ignore host name in the accounting files. With this option, user `smith` on host `alpha` is the same user `smith` on host `gamma`. Without, they are different users.

`-p`*`price`*

Compute charges with *`price`* dollars per page or per foot instead of the price from the `pc` capability in `/etc/printcap`, or two cents (the default). You can specify *`price`* as a floating point number.

`-r`

Reverse the sort order.

`-s`

Make an accounting summary file and truncate the accounting file.

`name ...`

Print accounting information for the given user *`names`* only.

In the default summary that pac(8) produces, you see the number of pages printed by each user from various hosts. If, at your site, host does not matter (because users can use any host), run `pac -m`, to produce the following summary:

```
  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
```

To compute the dollar amount due, pac(8) uses the `pc` capability in the `/etc/printcap` file (default of 200, or 2 cents per page). Specify, in hundredths of cents, the price per page or per foot you want to charge for printouts in this capability. You can override this value when you run pac(8) with the `-p` option. The units for the `-p` option are in dollars, though, not hundredths of cents. For example,

```
# pac -p1.50
```

makes each page cost one dollar and fifty cents. You can really rake in the profits by using this option.

Finally, running `pac -s` will save the summary information in a summary accounting file, which is named the same as the printer's accounting file, but with `_sum` appended to the name. It then truncates the accounting file. When you run pac(8) again, it rereads the summary file to get starting totals, then adds information from the regular accounting file.

9.4.5.2 How Can You Count Pages Printed?

In order to perform even remotely accurate accounting, you need to be able to determine how much paper a job uses. This is the essential problem of printer accounting.

For plain text jobs, the problem is not that hard to solve: you count how many lines are in a job and compare it to how many lines per page your printer supports. Do not forget to take into account backspaces in the file which overprint lines, or long logical lines that wrap onto one or more additional physical lines.

The text filter `lpf` (introduced in lpf: a Text Filter) takes into account these things when it does accounting. If you are writing a text filter which needs to do accounting, you might want to examine `lpf`'s source code.

How do you handle other file formats, though?

Well, for DVI-to-LaserJet or DVI-to-PostScript conversion, you can have your filter parse the diagnostic output of `dvilj` or `dvips` and look to see how many pages were converted. You might be able to do similar things with other file formats and conversion programs.

But these methods suffer from the fact that the printer may not actually print all those pages. For example, it could jam, run out of toner, or explode—and the user would still get charged.

So, what can you do?

There is only one *sure* way to do *accurate* accounting. Get a printer that can tell you how much paper it uses, and attach it via a serial line or a network connection. Nearly all PostScript printers support this notion. Other makes and models do as well (networked Imagen laser printers, for example). Modify the filters for these printers to get the page usage after they print each job and have them log accounting information based on that value *only*. There is no line counting nor error-prone file examination required.

Of course, you can always be generous and make all printouts free.

9.5 Using Printers

This section tells you how to use printers you have set up with FreeBSD. Here is an overview of the user-level commands:

lpr(1)

: Print jobs

lpq(1)

: Check printer queues

lprm(1)

: Remove jobs from a printer's queue

There is also an administrative command, lpc(8), described in the section Administering Printers, used to control printers and their queues.

All three of the commands lpr(1), lprm(1), and lpq(1) accept an option `-P` *`printer-name`* to specify on which printer/queue to operate, as listed in the `/etc/printcap` file. This enables you to submit, remove, and check on jobs for various printers. If you do not use the `-P` option, then these commands use the printer specified in the `PRINTER` environment variable. Finally, if you do not have a `PRINTER` environment variable, these commands default to the printer named `lp`.

Hereafter, the terminology *default printer* means the printer named in the `PRINTER` environment variable, or the printer named `lp` when there is no `PRINTER` environment variable.

9.5.1 Printing Jobs

To print files, type:

```
% lpr filename ...
```

This prints each of the listed files to the default printer. If you list no files, lpr(1) reads data to print from standard input. For example, this command prints some important system files:

```
% lpr /etc/host.conf /etc/hosts.equiv
```

To select a specific printer, type:

```
% lpr -P printer-name filename ...
```

This example prints a long listing of the current directory to the printer named `rattan`:

```
% ls -l | lpr -P rattan
```

Because no files were listed for the lpr(1) command, `lpr` read the data to print from standard input, which was the output of the `ls -l` command.

The lpr(1) command can also accept a wide variety of options to control formatting, apply file conversions, generate multiple copies, and so forth. For more information, see the section Printing Options.

9.5.2 Checking Jobs

When you print with lpr(1), the data you wish to print is put together in a package called a “print job”, which is sent to the **LPD** spooling system. Each printer has a queue of jobs, and your job waits in that queue along with other jobs from yourself and from other users. The printer prints those jobs in a first-come, first-served order.

To display the queue for the default printer, type lpq(1). For a specific printer, use the `-P` option. For example, the command

```
% lpq -P bamboo
```

shows the queue for the printer named `bamboo`. Here is an example of the output of the `lpq` command:

```
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
```

This shows three jobs in the queue for `bamboo`. The first job, submitted by user kelly, got assigned “job number” 9. Every job for a printer gets a unique job number. Most of the time you can ignore the job number, but you will need it if you want to cancel the job; see section Removing Jobs for details.

Job number nine consists of two files; multiple files given on the lpr(1) command line are treated as part of a single job. It is the currently active job (note the word `active` under the “Rank” column), which means the printer should be currently printing that job. The second job consists of data passed as the standard input to the lpr(1) command. The third job came from user `mary`; it is a much larger job. The pathname of the file she is trying to print is too long to fit, so the lpq(1) command just shows three dots.

The very first line of the output from lpq(1) is also useful: it tells what the printer is currently doing (or at least what **LPD** thinks the printer is doing).

The lpq(1) command also support a `-l` option to generate a detailed long listing. Here is an example of `lpq -l`:

```
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 Removing Jobs

If you change your mind about printing a job, you can remove the job from the queue with the lprm(1) command. Often, you can even use lprm(1) to remove an active job, but some or all of the job might still get printed.

To remove a job from the default printer, first use lpq(1) to find the job number. Then type:

```
% lprm job-number
```

To remove the job from a specific printer, add the `-P` option. The following command removes job number 10 from the queue for the printer `bamboo`:

```
% lprm -P bamboo 10
```

The lprm(1) command has a few shortcuts:

lprm -

> Removes all jobs (for the default printer) belonging to you.

lprm *user*

> Removes all jobs (for the default printer) belonging to *user*. The superuser can remove other users' jobs; you can remove only your own jobs.

lprm

> With no job number, user name, or `-` appearing on the command line, lprm(1) removes the currently active job on the default printer, if it belongs to you. The superuser can remove any active job.

Just use the `-P` option with the above shortcuts to operate on a specific printer instead of the default. For example, the following command removes all jobs for the current user in the queue for the printer named `rattan`:

```
% lprm -P rattan -
```

> **Note:** If you are working in a networked environment, lprm(1) will let you remove jobs only from the host from which the jobs were submitted, even if the same printer is available from other hosts. The following command sequence demonstrates this:
>
> ```
> % 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 Beyond Plain Text: Printing Options

The lpr(1) command supports a number of options that control formatting text, converting graphic and other file formats, producing multiple copies, handling of the job, and more. This section describes the options.

9.5.4.1 Formatting and Conversion Options

The following lpr(1) options control formatting of the files in the job. Use these options if the job does not contain plain text or if you want plain text formatted through the pr(1) utility.

For example, the following command prints a DVI file (from the TEX typesetting system) named *fish-report.dvi* to the printer named bamboo:

```
% lpr -P bamboo -d fish-report.dvi
```

These options apply to every file in the job, so you cannot mix (say) DVI and ditroff files together in a job. Instead, submit the files as separate jobs, using a different conversion option for each job.

> **Note:** All of these options except `-p` and `-T` require conversion filters installed for the destination printer. For example, the `-d` option requires the DVI conversion filter. Section Conversion Filters gives details.

`-c`

Print cifplot files.

`-d`

Print DVI files.

`-f`

Print FORTRAN text files.

`-g`

Print plot data.

`-i` *number*

Indent the output by *number* columns; if you omit *number*, indent by 8 columns. This option works only with certain conversion filters.

> **Note:** Do not put any space between the `-i` and the number.

`-l`

Print literal text data, including control characters.

`-n`

Print ditroff (device independent troff) data.

`-p`

Format plain text with pr(1) before printing. See pr(1) for more information.

`-T` *title*

Use *title* on the pr(1) header instead of the file name. This option has effect only when used with the `-p` option.

`-t`

Print troff data.

`-v`

Print raster data.

Here is an example: this command prints a nicely formatted version of the ls(1) manual page on the default printer:

```
% zcat /usr/share/man/man1/ls.1.gz | troff -t -man | lpr -t
```

The zcat(1) command uncompresses the source of the ls(1) manual page and passes it to the troff(1) command, which formats that source and makes GNU troff output and passes it to lpr(1), which submits the job to the **LPD** spooler. Because we used the `-t` option to lpr(1), the spooler will convert the GNU troff output into a format the default printer can understand when it prints the job.

9.5.4.2 Job Handling Options

The following options to lpr(1) tell **LPD** to handle the job specially:

`-#` *copies*

Produce a number of *copies* of each file in the job instead of just one copy. An administrator may disable this option to reduce printer wear-and-tear and encourage photocopier usage. See section Restricting Multiple Copies.

This example prints three copies of *parser.c* followed by three copies of *parser.h* to the default printer:

```
% lpr -#3 parser.c parser.h
```

-m

Send mail after completing the print job. With this option, the **LPD** system will send mail to your account when it finishes handling your job. In its message, it will tell you if the job completed successfully or if there was an error, and (often) what the error was.

-s

Do not copy the files to the spooling directory, but make symbolic links to them instead.

If you are printing a large job, you probably want to use this option. It saves space in the spooling directory (your job might overflow the free space on the filesystem where the spooling directory resides). It saves time as well since **LPD** will not have to copy each and every byte of your job to the spooling directory.

There is a drawback, though: since **LPD** will refer to the original files directly, you cannot modify or remove them until they have been printed.

> **Note:** If you are printing to a remote printer, **LPD** will eventually have to copy files from the local host to the remote host, so the `-s` option will save space only on the local spooling directory, not the remote. It is still useful, though.

-r

Remove the files in the job after copying them to the spooling directory, or after printing them with the `-s` option. Be careful with this option!

9.5.4.3 Header Page Options

These options to lpr(1) adjust the text that normally appears on a job's header page. If header pages are suppressed for the destination printer, these options have no effect. See section Header Pages for information about setting up header pages.

-C *`text`*

Replace the hostname on the header page with *`text`*. The hostname is normally the name of the host from which the job was submitted.

-J *`text`*

Replace the job name on the header page with *`text`*. The job name is normally the name of the first file of the job, or `stdin` if you are printing standard input.

-h

Do not print any header page.

> **Note:** At some sites, this option may have no effect due to the way header pages are generated. See Header Pages for details.

9.5.5 Administering Printers

As an administrator for your printers, you have had to install, set up, and test them. Using the lpc(8) command, you can interact with your printers in yet more ways. With lpc(8), you can

- Start and stop the printers
- Enable and disable their queues
- Rearrange the order of the jobs in each queue.

First, a note about terminology: if a printer is *stopped*, it will not print anything in its queue. Users can still submit jobs, which will wait in the queue until the printer is *started* or the queue is cleared.

If a queue is *disabled*, no user (except `root`) can submit jobs for the printer. An *enabled* queue allows jobs to be submitted. A printer can be *started* for a disabled queue, in which case it will continue to print jobs in the queue until the queue is empty.

In general, you have to have `root` privileges to use the lpc(8) command. Ordinary users can use the lpc(8) command to get printer status and to restart a hung printer only.

Here is a summary of the lpc(8) commands. Most of the commands take a *`printer-name`* argument to tell on which printer to operate. You can use `all` for the *`printer-name`* to mean all printers listed in `/etc/printcap`.

`abort` *`printer-name`*

Cancel the current job and stop the printer. Users can still submit jobs if the queue is enabled.

`clean` *`printer-name`*

Remove old files from the printer's spooling directory. Occasionally, the files that make up a job are not properly removed by **LPD**, particularly if there have been errors during printing or a lot of administrative activity. This command finds files that do not belong in the spooling directory and removes them.

`disable` *`printer-name`*

Disable queuing of new jobs. If the printer is running, it will continue to print any jobs remaining in the queue. The superuser (`root`) can always submit jobs, even to a disabled queue.

This command is useful while you are testing a new printer or filter installation: disable the queue and submit jobs as `root`. Other users will not be able to submit jobs until you complete your testing and re-enable the queue with the `enable` command.

`down` *`printer-name message`*

Take a printer down. Equivalent to `disable` followed by `stop`. The *`message`* appears as the printer's status whenever a user checks the printer's queue with lpq(1) or status with `lpc status`.

`enable` *`printer-name`*

Enable the queue for a printer. Users can submit jobs but the printer will not print anything until it is started.

`help` *`command-name`*

Print help on the command *`command-name`*. With no *`command-name`*, print a summary of the commands available.

`restart` *`printer-name`*

Start the printer. Ordinary users can use this command if some extraordinary circumstance hangs **LPD**, but they cannot start a printer stopped with either the `stop` or `down` commands. The `restart` command is equivalent to `abort` followed by `start`.

`start` *`printer-name`*

Start the printer. The printer will print jobs in its queue.

`stop` *`printer-name`*

Stop the printer. The printer will finish the current job and will not print anything else in its queue. Even though the printer is stopped, users can still submit jobs to an enabled queue.

`topq` *`printer-name job-or-username`*

Rearrange the queue for *`printer-name`* by placing the jobs with the listed *`job`* numbers or the jobs belonging to *`username`* at the top of the queue. For this command, you cannot use `all` as the *`printer-name`*.

`up` *`printer-name`*

Bring a printer up; the opposite of the `down` command. Equivalent to `start` followed by `enable`.

lpc(8) accepts the above commands on the command line. If you do not enter any commands, lpc(8) enters an interactive mode, where you can enter commands until you type `exit`, `quit`, or end-of-file.

9.6 Alternatives to the Standard Spooler

If you have been reading straight through this manual, by now you have learned just about everything there is to know about the **LPD** spooling system that comes with FreeBSD. You can probably appreciate many of its shortcomings, which naturally leads to the question: “What other spooling systems are out there (and work with FreeBSD)?”

LPRng

LPRng, which purportedly means “LPR: the Next Generation” is a complete rewrite of PLP. Patrick Powell and Justin Mason (the principal maintainer of PLP) collaborated to make **LPRng**. The main site for **LPRng** is http://www.lprng.org/.

CUPS

CUPS, the Common UNIX Printing System, provides a portable printing layer for UNIX-based operating systems. It has been developed by Easy Software Products to promote a standard printing solution for all UNIX vendors and users.

CUPS uses the Internet Printing Protocol (IPP) as the basis for managing print jobs and queues. The Line Printer Daemon (LPD), Server Message Block (SMB), and AppSocket (a.k.a. JetDirect) protocols are also

supported with reduced functionality. CUPS adds network printer browsing and PostScript Printer Description (PPD) based printing options to support real-world printing under UNIX.

The main site for **CUPS** is http://www.cups.org/.

HPLIP

HPLIP, the HP Linux Imaging and Printing system, is an HP-developed suite of programs that supports printing, scanning and fax facilities for HP appliances. This suite of programs utilizes the **CUPS** printing system as a backend for some of its printing features.

The main site for **HPLIP** is http://hplipopensource.com/hplip-web/index.html.

9.7 Troubleshooting

After performing the simple test with lptest(1), you might have gotten one of the following results instead of the correct printout:

It worked, after awhile; or, it did not eject a full sheet.

The printer printed the above, but it sat for awhile and did nothing. In fact, you might have needed to press a PRINT REMAINING or FORM FEED button on the printer to get any results to appear.

If this is the case, the printer was probably waiting to see if there was any more data for your job before it printed anything. To fix this problem, you can have the text filter send a FORM FEED character (or whatever is necessary) to the printer. This is usually sufficient to have the printer immediately print any text remaining in its internal buffer. It is also useful to make sure each print job ends on a full sheet, so the next job does not start somewhere on the middle of the last page of the previous job.

The following replacement for the shell script `/usr/local/libexec/if-simple` prints a form feed after it sends the job to the printer:

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout.  Ignores all filter arguments.
# Writes a form feed character (\f) after printing job.

/bin/cat && printf "\f" && exit 0
exit 2
```

It produced the “staircase effect.”

You got the following on paper:

```
!"#$%&'()*+,-./01234
               "#$%&'()*+,-./012345
                              #$%&'()*+,-./0123456
```

You have become another victim of the *staircase effect*, caused by conflicting interpretations of what characters should indicate a new line. UNIX style operating systems use a single character: ASCII code 10, the line feed (LF). MS-DOS, OS/2®, and others uses a pair of characters, ASCII code 10 *and* ASCII code 13 (the carriage return or CR). Many printers use the MS-DOS convention for representing new-lines.

When you print with FreeBSD, your text used just the line feed character. The printer, upon seeing a line feed character, advanced the paper one line, but maintained the same horizontal position on the page for the next character to print. That is what the carriage return is for: to move the location of the next character to print to the left edge of the paper.

Here is what FreeBSD wants your printer to do:

Printer received CR	Printer prints CR
Printer received LF	Printer prints CR + LF

Here are some ways to achieve this:

- Use the printer's configuration switches or control panel to alter its interpretation of these characters. Check your printer's manual to find out how to do this.

  > **Note:** If you boot your system into other operating systems besides FreeBSD, you may have to *reconfigure* the printer to use a an interpretation for CR and LF characters that those other operating systems use. You might prefer one of the other solutions, below.

- Have FreeBSD's serial line driver automatically convert LF to CR+LF. Of course, this works with printers on serial ports *only*. To enable this feature, use the `ms#` capability and set the `onlcr` mode in the `/etc/printcap` file for the printer.
- Send an *escape code* to the printer to have it temporarily treat LF characters differently. Consult your printer's manual for escape codes that your printer might support. When you find the proper escape code, modify the text filter to send the code first, then send the print job.

  Here is an example text filter for printers that understand the Hewlett-Packard PCL escape codes. This filter makes the printer treat LF characters as a LF and CR; then it sends the job; then it sends a form feed to eject the last page of the job. It should work with nearly all Hewlett Packard printers.

  ```
  #!/bin/sh
  #
  # hpif - Simple text input filter for lpd for HP-PCL based printers
  # Installed in /usr/local/libexec/hpif
  #
  # Simply copies stdin to stdout.  Ignores all filter arguments.
  # Tells printer to treat LF as CR+LF.  Ejects the page when done.

  printf "\033&k2G" && cat && printf "\033&l0H" && exit 0
  exit 2
  ```

  Here is an example `/etc/printcap` from a host called `orchid`. It has a single printer attached to its first parallel port, a Hewlett Packard LaserJet 3Si named `teak`. It is using the above script as its text filter:

  ```
  #
  #  /etc/printcap for host orchid
  #
  teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
          :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
          :if=/usr/local/libexec/hpif:
  ```

It overprinted each line.

The printer never advanced a line. All of the lines of text were printed on top of each other on one line.

This problem is the “opposite” of the staircase effect, described above, and is much rarer. Somewhere, the LF characters that FreeBSD uses to end a line are being treated as CR characters to return the print location to the left edge of the paper, but not also down a line.

Use the printer’s configuration switches or control panel to enforce the following interpretation of LF and CR characters:

Printer receives	Printer prints
CR	CR
LF	CR + LF

The printer lost characters.

While printing, the printer did not print a few characters in each line. The problem might have gotten worse as the printer ran, losing more and more characters.

The problem is that the printer cannot keep up with the speed at which the computer sends data over a serial line (this problem should not occur with printers on parallel ports). There are two ways to overcome the problem:

- If the printer supports XON/XOFF flow control, have FreeBSD use it by specifying the `ixon` mode in the `ms#` capability.
- If the printer supports the Request to Send / Clear to Send hardware handshake (commonly known as `RTS/CTS`), specify the `crtscts` mode in the `ms#` capability. Make sure the cable connecting the printer to the computer is correctly wired for hardware flow control.

It printed garbage.

The printer printed what appeared to be random garbage, but not the desired text.

This is usually another symptom of incorrect communications parameters with a serial printer. Double-check the bps rate in the `br` capability, and the parity setting in the `ms#` capability; make sure the printer is using the same settings as specified in the `/etc/printcap` file.

Nothing happened.

If nothing happened, the problem is probably within FreeBSD and not the hardware. Add the log file (`lf`) capability to the entry for the printer you are debugging in the `/etc/printcap` file. For example, here is the entry for `rattan`, with the `lf` capability:

```
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
```

Then, try printing again. Check the log file (in our example, `/var/log/rattan.log`) to see any error messages that might appear. Based on the messages you see, try to correct the problem.

If you do not specify a `lf` capability, **LPD** uses `/dev/console` as a default.

Chapter 10
Linux Binary Compatibility

10.1 Synopsis

FreeBSD provides binary compatibility with several other UNIX like operating systems, including Linux. At this point, you may be asking yourself why exactly, does FreeBSD need to be able to run Linux binaries? The answer to that question is quite simple. Many companies and developers develop only for Linux, since it is the latest "hot thing" in the computing world. That leaves the rest of us FreeBSD users bugging these same companies and developers to put out native FreeBSD versions of their applications. The problem is, that most of these companies do not really realize how many people would use their product if there were FreeBSD versions too, and most continue to only develop for Linux. So what is a FreeBSD user to do? This is where the Linux binary compatibility of FreeBSD comes into play.

In a nutshell, the compatibility allows FreeBSD users to run about 90% of all Linux applications without modification. This includes applications such as **StarOffice**, the Linux version of **Netscape**, **Adobe Acrobat**, **RealPlayer**, **VMware™**, **Oracle**, **WordPerfect®**, **Doom**, **Quake**, and more. It is also reported that in some situations, Linux binaries perform better on FreeBSD than they do under Linux.

There are, however, some Linux-specific operating system features that are not supported under FreeBSD. Linux binaries will not work on FreeBSD if they overly use i386 specific calls, such as enabling virtual 8086 mode.

After reading this chapter, you will know:

- How to enable Linux binary compatibility on your system.
- How to install additional Linux shared libraries.
- How to install Linux applications on your FreeBSD system.
- The implementation details of Linux compatibility in FreeBSD.

Before reading this chapter, you should:

- Know how to install additional third-party software (Chapter 4).

10.2 Installation

Linux binary compatibility is not turned on by default. The easiest way to enable this functionality is to load the `linux` KLD object ("Kernel LoaDable object"). You can load this module by typing the following as `root`:

```
# kldload linux
```

If you would like Linux compatibility to always be enabled, then you should add the following line to `/etc/rc.conf`:

```
linux_enable="YES"
```

The kldstat(8) command can be used to verify that the KLD is loaded:

```
% kldstat
Id Refs Address    Size     Name
 1    2 0xc0100000 16bdb8   kernel
 7    1 0xc24db000 d000     linux.ko
```

If for some reason you do not want to or cannot load the KLD, then you may statically link Linux binary compatibility into the kernel by adding `options COMPAT_LINUX` to your kernel configuration file. Then install your new kernel as described in Chapter 8.

10.2.1 Installing Linux Runtime Libraries

This can be done one of two ways, either by using the linux_base port, or by installing them manually.

10.2.1.1 Installing Using the linux_base Port

This is by far the easiest method to use when installing the runtime libraries. It is just like installing any other port from the Ports Collection (/usr/ports/). Simply do the following:

```
# cd /usr/ports/emulators/linux_base-f10
# make install distclean
```

> **Note:** On FreeBSD systems prior to FreeBSD 8.0, you will have to use the `emulators/linux_base-fc4` port instead of `emulators/linux_base-f10`.

You should now have working Linux binary compatibility. Some programs may complain about incorrect minor versions of the system libraries. In general, however, this does not seem to be a problem.

> **Note:** There may be multiple versions of the `emulators/linux_base` port available, corresponding to different versions of various Linux distributions. You should install the port most closely resembling the requirements of the Linux applications you would like to install.

10.2.1.2 Installing Libraries Manually

If you do not have the “ports” collection installed, you can install the libraries by hand instead. You will need the Linux shared libraries that the program depends on and the runtime linker. Also, you will need to create a “shadow root” directory, `/compat/linux`, for Linux libraries on your FreeBSD system. Any shared libraries opened by Linux programs run under FreeBSD will look in this tree first. So, if a Linux program loads, for example, `/lib/libc.so`, FreeBSD will first try to open `/compat/linux/lib/libc.so`, and if that does not exist, it will then try `/lib/libc.so`. Shared libraries should be installed in the shadow tree `/compat/linux/lib` rather than the paths that the Linux `ld.so` reports.

Generally, you will need to look for the shared libraries that Linux binaries depend on only the first few times that you install a Linux program on your FreeBSD system. After a while, you will have a sufficient set of Linux shared libraries on your system to be able to run newly imported Linux binaries without any extra work.

10.2.1.3 How to Install Additional Shared Libraries

What if you install the `linux_base` port and your application still complains about missing shared libraries? How do you know which shared libraries Linux binaries need, and where to get them? Basically, there are 2 possibilities (when following these instructions you will need to be `root` on your FreeBSD system).

If you have access to a Linux system, see what shared libraries the application needs, and copy them to your FreeBSD system. Look at the following example:

Let us assume you used FTP to get the Linux binary of **Doom**, and put it on a Linux system you have access to. You then can check which shared libraries it needs by running `ldd linuxdoom`, like so:

```
% 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 4.5pl26) => /lib/libc.so.4.6.29
```

You would need to get all the files from the last column, and put them under `/compat/linux`, with the names in the first column as symbolic links pointing to them. This means you eventually have these files on your FreeBSD system:

```
/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
```

> **Note:** Note that if you already have a Linux shared library with a matching major revision number to the first column of the `ldd` output, you will not need to copy the file named in the last column to your system, the one you already have should work. It is advisable to copy the shared library anyway if it is a newer version, though. You can remove the old one, as long as you make the symbolic link point to the new one. So, if you have these libraries on your system:
>
> ```
> /compat/linux/lib/libc.so.4.6.27
> /compat/linux/lib/libc.so.4 -> libc.so.4.6.27
> ```
>
> and you find a new binary that claims to require a later version according to the output of `ldd`:
>
> ```
> libc.so.4 (DLL Jump 4.5pl26) -> libc.so.4.6.29
> ```
>
> If it is only one or two versions out of date in the trailing digit then do not worry about copying `/lib/libc.so.4.6.29` too, because the program should work fine with the slightly older version. However, if you like, you can decide to replace the `libc.so` anyway, and that should leave you with:
>
> ```
> /compat/linux/lib/libc.so.4.6.29
> /compat/linux/lib/libc.so.4 -> libc.so.4.6.29
> ```

> **Note:** The symbolic link mechanism is *only* needed for Linux binaries. The FreeBSD runtime linker takes care of looking for matching major revision numbers itself and you do not need to worry about it.

10.2.2 Installing Linux ELF Binaries

ELF binaries sometimes require an extra step of “branding”. If you attempt to run an unbranded ELF binary, you will get an error message like the following:

```
% ./my-linux-elf-binary
ELF binary type not known
Abort
```

To help the FreeBSD kernel distinguish between a FreeBSD ELF binary and a Linux binary, use the brandelf(1) utility.

```
% brandelf -t Linux my-linux-elf-binary
```

The GNU toolchain now places the appropriate branding information into ELF binaries automatically, so this step should become increasingly unnecessary in the future.

10.2.3 Installing a Random Linux RPM Based Application

FreeBSD has its own package database and it is used to track all ports (Linux ports as well). So the Linux RPM database is not used (not supported).

However if you need to install a random Linux RPM-based application it can be achieved by:

```
# cd /compat/linux
# rpm2cpio -q < /path/to/linux.archive.rpm | cpio -id
```

Then brandelf installed ELF binaries (not libraries!). You will not be able to do a clean uninstall, but it may help you to do tests.

10.2.4 Configuring the Hostname Resolver

If DNS does not work or you get this message:

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

You will need to configure a `/compat/linux/etc/host.conf` file containing:

```
order hosts, bind
multi on
```

The order here specifies that `/etc/hosts` is searched first and DNS is searched second. When `/compat/linux/etc/host.conf` is not installed, Linux applications find FreeBSD’s `/etc/host.conf` and

complain about the incompatible FreeBSD syntax. You should remove `bind` if you have not configured a name server using the `/etc/resolv.conf` file.

10.3 Installing Mathematica®

This document describes the process of installing the Linux version of **Mathematica 5.X** onto a FreeBSD system.

The Linux version of **Mathematica** or **Mathematica for Students** can be ordered directly from Wolfram at http://www.wolfram.com/.

10.3.1 Running the Mathematica Installer

First, you have to tell FreeBSD that **Mathematica**'s Linux binaries use the Linux ABI. The easiest way to do so is to set the default ELF brand to Linux for all unbranded binaries with the command:

```
# sysctl kern.fallback_elf_brand=3
```

This will make FreeBSD assume that unbranded ELF binaries use the Linux ABI and so you should be able to run the installer straight from the CDROM.

Now, copy the file `MathInstaller` to your hard drive:

```
# mount /cdrom
# cp /cdrom/Unix/Installers/Linux/MathInstaller /localdir/
```

and in this file, replace `/bin/sh` in the first line by `/compat/linux/bin/sh`. This makes sure that the installer is executed by the Linux version of sh(1). Next, replace all occurrences of `Linux)` by `FreeBSD)` with a text editor or the script below in the next section. This tells the **Mathematica** installer, who calls `uname -s` to determine the operating system, to treat FreeBSD as a Linux-like operating system. Invoking `MathInstaller` will now install **Mathematica**.

10.3.2 Modifying the Mathematica Executables

The shell scripts that **Mathematica** created during installation have to be modified before you can use them. If you chose `/usr/local/bin` as the directory to place the **Mathematica** executables in, you will find symlinks in this directory to files called `math`, `mathematica`, `Mathematica`, and `MathKernel`. In each of these, replace `Linux)` by `FreeBSD)` with a text editor or the following shell script:

```
#!/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 Obtaining Your Mathematica Password

When you start **Mathematica** for the first time, you will be asked for a password. If you have not yet obtained a password from Wolfram, run the program `mathinfo` in the installation directory to obtain your "machine ID". This machine ID is based solely on the MAC address of your first Ethernet card, so you cannot run your copy of **Mathematica** on different machines.

When you register with Wolfram, either by email, phone or fax, you will give them the "machine ID" and they will respond with a corresponding password consisting of groups of numbers.

10.3.4 Running the Mathematica Frontend over a Network

Mathematica uses some special fonts to display characters not present in any of the standard font sets (integrals, sums, Greek letters, etc.). The X protocol requires these fonts to be install *locally*. This means you will have to copy these fonts from the CDROM or from a host with **Mathematica** installed to your local machine. These fonts are normally stored in `/cdrom/Unix/Files/SystemFiles/Fonts` on the CDROM, or `/usr/local/mathematica/SystemFiles/Fonts` on your hard drive. The actual fonts are in the subdirectories `Type1` and `X`. There are several ways to use them, as described below.

The first way is to copy them into one of the existing font directories in `/usr/X11R6/lib/X11/fonts`. This will require editing the `fonts.dir` file, adding the font names to it, and changing the number of fonts on the first line. Alternatively, you should also just be able to run mkfontdir(1) in the directory you have copied them to.

The second way to do this is to copy the directories to `/usr/X11R6/lib/X11/fonts`:

```
# 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
```

Now add the new font directories to your font path:

```
# xset fp+ /usr/X11R6/lib/X11/fonts/X
# xset fp+ /usr/X11R6/lib/X11/fonts/MathType1
# xset fp rehash
```

If you are using the **Xorg** server, you can have these font directories loaded automatically by adding them to your `xorg.conf` file.

If you *do not* already have a directory called `/usr/X11R6/lib/X11/fonts/Type1`, you can change the name of the `MathType1` directory in the example above to `Type1`.

10.4 Installing Maple™

Maple™ is a commercial mathematics program similar to **Mathematica**. You must purchase this software from http://www.maplesoft.com/ and then register there for a license file. To install this software on FreeBSD, please follow these simple steps.

1. Execute the `INSTALL` shell script from the product distribution. Choose the "RedHat" option when prompted by the installation program. A typical installation directory might be `/usr/local/maple`.
2. If you have not done so, order a license for **Maple** from Maple Waterloo Software (http://register.maplesoft.com/) and copy it to `/usr/local/maple/license/license.dat`.
3. Install the **FLEXlm** license manager by running the `INSTALL_LIC` install shell script that comes with **Maple**. Specify the primary hostname for your machine for the license server.
4. Patch the `/usr/local/maple/bin/maple.system.type` file with the following:

```
   ----- snip ------------------
*** 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
   ----- snip end of patch -----
```

   Please note that after the `"FreeBSD"|\` no other whitespace should be present.

   This patch instructs **Maple** to recognize "FreeBSD" as a type of Linux system. The `bin/maple` shell script calls the `bin/maple.system.type` shell script which in turn calls `uname -a` to find out the operating system name. Depending on the OS name it will find out which binaries to use.

5. Start the license server.

   The following script, installed as `/usr/local/etc/rc.d/lmgrd.sh` is a convenient way to start up `lmgrd`:

```
   ----- snip ------------

#! /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
   ----- snip ------------
```

6. Test-start **Maple**:

```
% cd /usr/local/maple/bin
% ./xmaple
```

You should be up and running. Make sure to write Maplesoft to let them know you would like a native FreeBSD version!

10.4.1 Common Pitfalls

- The **FLEXlm** license manager can be a difficult tool to work with. Additional documentation on the subject can be found at http://www.globetrotter.com/.
- `lmgrd` is known to be very picky about the license file and to core dump if there are any problems. A correct license file should look like this:

```
# ====================================================
# License File for UNIX Installations ("Pointer File")
# ====================================================
SERVER chillig ANY
#USE_SERVER
VENDOR maplelmg

FEATURE Maple maplelmg 2000.0831 permanent 1 XXXXXXXXXXX \
         PLATFORMS=i86_r ISSUER="Waterloo Maple Inc." \
         ISSUED=11-may-2000 NOTICE=" Technische Universitat Wien" \
         SN=XXXXXXXXX
```

Note: Serial number and key ’X”ed out. `chillig` is a hostname.

Editing the license file works as long as you do not touch the “FEATURE” line (which is protected by the license key).

10.5 Installing MATLAB®

This document describes the process of installing the Linux version of **MATLAB® version 6.5** onto a FreeBSD system. It works quite well, with the exception of the **Java Virtual Machine™** (see Section 10.5.3).

The Linux version of **MATLAB** can be ordered directly from The MathWorks at http://www.mathworks.com. Make sure you also get the license file or instructions how to create it. While you are there, let them know you would like a native FreeBSD version of their software.

10.5.1 Installing MATLAB

To install **MATLAB**, do the following:

1. Insert the installation CD and mount it. Become `root`, as recommended by the installation script. To start the installation script type:

   ```
   # /compat/linux/bin/sh /cdrom/install
   ```

   > **Tip:** The installer is graphical. If you get errors about not being able to open a display, type `setenv HOME ~USER`, where *USER* is the user you did a su(1) as.

2. When asked for the **MATLAB** root directory, type: **`/compat/linux/usr/local/matlab`**.

   > **Tip:** For easier typing on the rest of the installation process, type this at your shell prompt: `set MATLAB=/compat/linux/usr/local/matlab`

3. Edit the license file as instructed when obtaining the **MATLAB** license.

   > **Tip:** You can prepare this file in advance using your favorite editor, and copy it to `$MATLAB/license.dat` before the installer asks you to edit it.

4. Complete the installation process.

At this point your **MATLAB** installation is complete. The following steps apply “glue” to connect it to your FreeBSD system.

10.5.2 License Manager Startup

1. Create symlinks for the license manager scripts:

   ```
   # ln -s $MATLAB/etc/lmboot /usr/local/etc/lmboot_TMW
   # ln -s $MATLAB/etc/lmdown /usr/local/etc/lmdown_TMW
   ```

2. Create a startup file at `/usr/local/etc/rc.d/flexlm.sh`. The example below is a modified version of the distributed `$MATLAB/etc/rc.lm.glnx86`. The changes are file locations, and startup of the license manager under Linux emulation.

```
#!/bin/sh
case "$1" in
  start)
        if [ -f /usr/local/etc/lmboot_TMW ]; then
              /compat/linux/bin/sh /usr/local/etc/lmboot_TMW -u username && echo 'MATLAB_lmgrd'
        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
```

Important: The file must be made executable:

```
# chmod +x /usr/local/etc/rc.d/flexlm.sh
```

You must also replace *username* above with the name of a valid user on your system (and not `root`).

3. Start the license manager with the command:

```
# /usr/local/etc/rc.d/flexlm.sh start
```

10.5.3 Linking the Java Runtime Environment

Change the **Java** Runtime Environment (JRE) link to one working under FreeBSD:

```
# cd $MATLAB/sys/java/jre/glnx86/
# unlink jre; ln -s ./jre1.1.8 ./jre
```

10.5.4 Creating a MATLAB Startup Script

1. Place the following startup script in `/usr/local/bin/matlab`:

```
#!/bin/sh
/compat/linux/bin/sh /compat/linux/usr/local/matlab/bin/matlab "$@"
```

2. Then type the command `chmod +x /usr/local/bin/matlab`.

Tip: Depending on your version of `emulators/linux_base`, you may run into errors when running this script. To avoid that, edit the file `/compat/linux/usr/local/matlab/bin/matlab`, and change the line that says:

```
if [ `expr "$lscmd" : '.*->.*'` -ne 0 ]; then
```

(in version 13.0.1 it is on line 410) to this line:

```
if test -L $newbase; then
```

10.5.5 Creating a MATLAB Shutdown Script

The following is needed to solve a problem with MATLAB not exiting correctly.

1. Create a file `$MATLAB/toolbox/local/finish.m`, and in it put the single line:

   ```
   ! $MATLAB/bin/finish.sh
   ```

   Note: The `$MATLAB` is literal.

   Tip: In the same directory, you will find the files `finishsav.m` and `finishdlg.m`, which let you save your workspace before quitting. If you use either of them, insert the line above immediately after the `save` command.

2. Create a file `$MATLAB/bin/finish.sh`, which will contain the following:

   ```
   #!/usr/compat/linux/bin/sh
   (sleep 5; killall -1 matlab_helper) &
   exit 0
   ```

3. Make the file executable:

   ```
   # chmod +x $MATLAB/bin/finish.sh
   ```

10.5.6 Using MATLAB

At this point you are ready to type `matlab` and start using it.

10.6 Installing Oracle®

10.6.1 Preface

This document describes the process of installing **Oracle 8.0.5** and **Oracle 8.0.5.1 Enterprise Edition** for Linux onto a FreeBSD machine.

10.6.2 Installing the Linux Environment

Make sure you have both `emulators/linux_base` and `devel/linux_devtools` from the Ports Collection installed. If you run into difficulties with these ports, you may have to use the packages or older versions available in the Ports Collection.

If you want to run the intelligent agent, you will also need to install the Red Hat Tcl package: `tcl-8.0.3-20.i386.rpm`. The general command for installing packages with the official **RPM** port (`archivers/rpm`) is:

```
# rpm -i --ignoreos --root /compat/linux --dbpath /var/lib/rpm package
```

Installation of the *package* should not generate any errors.

10.6.3 Creating the Oracle Environment

Before you can install **Oracle**, you need to set up a proper environment. This document only describes what to do *specially* to run **Oracle** for Linux on FreeBSD, not what has been described in the **Oracle** installation guide.

10.6.3.1 Kernel Tuning

As described in the **Oracle** installation guide, you need to set the maximum size of shared memory. Do not use `SHMMAX` under FreeBSD. `SHMMAX` is merely calculated out of `SHMMAXPGS` and `PGSIZE`. Therefore define `SHMMAXPGS`. All other options can be used as described in the guide. For example:

```
options SHMMAXPGS=10000
options SHMMNI=100
options SHMSEG=10
options SEMMNS=200
options SEMMNI=70
options SEMMSL=61
```

Set these options to suit your intended use of **Oracle**.

Also, make sure you have the following options in your kernel configuration file:

```
options SYSVSHM #SysV shared memory
options SYSVSEM #SysV semaphores
options SYSVMSG #SysV interprocess communication
```

10.6.3.2 Oracle Account

Create an `oracle` account just as you would create any other account. The `oracle` account is special only that you need to give it a Linux shell. Add `/compat/linux/bin/bash` to `/etc/shells` and set the shell for the `oracle` account to `/compat/linux/bin/bash`.

10.6.3.3 Environment

Besides the normal **Oracle** variables, such as `ORACLE_HOME` and `ORACLE_SID` you must set the following environment variables:

Variable	Value
LD_LIBRARY_PATH	$ORACLE_HOME/lib
CLASSPATH	$ORACLE_HOME/jdbc/lib/classes111.zip
PATH	/compat/linux/bin /compat/linux/sbin /compat/linux/usr/bin /compat/linux/usr/sbin /bin /sbin /usr/bin /usr/sbin /usr/local/bin $ORACLE_HOME/bin

It is advised to set all the environment variables in `.profile`. A complete example is:

```
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 Installing Oracle

Due to a slight inconsistency in the Linux emulator, you need to create a directory named `.oracle` in `/var/tmp` before you start the installer. Let it be owned by the `oracle` user. You should be able to install **Oracle** without any problems. If you have problems, check your **Oracle** distribution and/or configuration first! After you have installed **Oracle**, apply the patches described in the next two subsections.

A frequent problem is that the TCP protocol adapter is not installed right. As a consequence, you cannot start any TCP listeners. The following actions help solve this problem:

```
# 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
```

Do not forget to run `root.sh` again!

10.6.4.1 Patching root.sh

When installing **Oracle**, some actions, which need to be performed as `root`, are recorded in a shell script called `root.sh`. This script is written in the `orainst` directory. Apply the following patch to `root.sh`, to have it use to proper location of `chown` or alternatively run the script under a Linux native shell.

```
*** 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
```

When you do not install **Oracle** from CD, you can patch the source for `root.sh`. It is called `rthd.sh` and is located in the `orainst` directory in the source tree.

10.6.4.2 Patching genclntsh

The script `genclntsh` is used to create a single shared client library. It is used when building the demos. Apply the following patch to comment out the definition of `PATH`:

```
*** 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 Running Oracle

When you have followed the instructions, you should be able to run **Oracle** as if it was run on Linux itself.

10.7 Advanced Topics

If you are curious as to how the Linux binary compatibility works, this is the section you want to read. Most of what follows is based heavily on an email written to FreeBSD chat mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat) by Terry Lambert <tlambert@primenet.com> (Message ID: <199906020108.SAA07001@usr09.primenet.com>).

10.7.1 How Does It Work?

FreeBSD has an abstraction called an "execution class loader". This is a wedge into the execve(2) system call.

What happens is that FreeBSD has a list of loaders, instead of a single loader with a fallback to the `#!` loader for running any shell interpreters or shell scripts.

Historically, the only loader on the UNIX platform examined the magic number (generally the first 4 or 8 bytes of the file) to see if it was a binary known to the system, and if so, invoked the binary loader.

If it was not the binary type for the system, the execve(2) call returned a failure, and the shell attempted to start executing it as shell commands.

The assumption was a default of "whatever the current shell is".

Later, a hack was made for sh(1) to examine the first two characters, and if they were `:\n`, then it invoked the csh(1) shell instead (we believe SCO first made this hack).

What FreeBSD does now is go through a list of loaders, with a generic `#!` loader that knows about interpreters as the characters which follow to the next whitespace next to last, followed by a fallback to `/bin/sh`.

For the Linux ABI support, FreeBSD sees the magic number as an ELF binary (it makes no distinction between FreeBSD, Solaris, Linux, or any other OS which has an ELF image type, at this point).

The ELF loader looks for a specialized *brand*, which is a comment section in the ELF image, and which is not present on SVR4/Solaris ELF binaries.

For Linux binaries to function, they must be *branded* as type `Linux` from brandelf(1):

```
# brandelf -t Linux file
```

When this is done, the ELF loader will see the `Linux` brand on the file.

When the ELF loader sees the `Linux` brand, the loader replaces a pointer in the `proc` structure. All system calls are indexed through this pointer (in a traditional UNIX system, this would be the `sysent[]` structure array, containing the system calls). In addition, the process is flagged for special handling of the trap vector for the signal trampoline code, and several other (minor) fix-ups that are handled by the Linux kernel module.

The Linux system call vector contains, among other things, a list of `sysent[]` entries whose addresses reside in the kernel module.

When a system call is called by the Linux binary, the trap code dereferences the system call function pointer off the `proc` structure, and gets the Linux, not the FreeBSD, system call entry points.

In addition, the Linux mode dynamically *reroots* lookups; this is, in effect, what the `union` option to file system mounts (*not* the `unionfs` file system type!) does. First, an attempt is made to lookup the file in the `/compat/linux/`*`original-path`* directory, *then* only if that fails, the lookup is done in the */original-path* directory. This makes sure that binaries that require other binaries can run (e.g., the Linux toolchain can all run under Linux ABI support). It also means that the Linux binaries can load and execute FreeBSD binaries, if there are no

corresponding Linux binaries present, and that you could place a uname(1) command in the `/compat/linux` directory tree to ensure that the Linux binaries could not tell they were not running on Linux.

In effect, there is a Linux kernel in the FreeBSD kernel; the various underlying functions that implement all of the services provided by the kernel are identical to both the FreeBSD system call table entries, and the Linux system call table entries: file system operations, virtual memory operations, signal delivery, System V IPC, etc... The only difference is that FreeBSD binaries get the FreeBSD *glue* functions, and Linux binaries get the Linux *glue* functions (most older OS's only had their own *glue* functions: addresses of functions in a static global `sysent[]` structure array, instead of addresses of functions dereferenced off a dynamically initialized pointer in the `proc` structure of the process making the call).

Which one is the native FreeBSD ABI? It does not matter. Basically the only difference is that (currently; this could easily be changed in a future release, and probably will be after this) the FreeBSD *glue* functions are statically linked into the kernel, and the Linux *glue* functions can be statically linked, or they can be accessed via a kernel module.

Yeah, but is this really emulation? No. It is an ABI implementation, not an emulation. There is no emulator (or simulator, to cut off the next question) involved.

So why is it sometimes called "Linux emulation"? To make it hard to sell FreeBSD! Really, it is because the historical implementation was done at a time when there was really no word other than that to describe what was going on; saying that FreeBSD ran Linux binaries was not true, if you did not compile the code in or load a module, and there needed to be a word to describe what was being loaded—hence "the Linux emulator".

III. System Administration

The remaining chapters of the FreeBSD Handbook cover all aspects of FreeBSD system administration. Each chapter starts by describing what you will learn as a result of reading the chapter, and also details what you are expected to know before tackling the material.

These chapters are designed to be read when you need the information. You do not have to read them in any particular order, nor do you need to read all of them before you can begin using FreeBSD.

Chapter 11
Configuration and Tuning

11.1 Synopsis

One of the important aspects of FreeBSD is system configuration. Correct system configuration will help prevent headaches during future upgrades. This chapter will explain much of the FreeBSD configuration process, including some of the parameters which can be set to tune a FreeBSD system.

After reading this chapter, you will know:

- How to efficiently work with file systems and swap partitions.
- The basics of `rc.conf` configuration and `/usr/local/etc/rc.d` startup systems.
- How to configure and test a network card.
- How to configure virtual hosts on your network devices.
- How to use the various configuration files in `/etc`.
- How to tune FreeBSD using `sysctl` variables.
- How to tune disk performance and modify kernel limitations.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Chapter 3).
- Be familiar with the basics of kernel configuration/compilation (Chapter 8).

11.2 Initial Configuration

11.2.1 Partition Layout

11.2.1.1 Base Partitions

When laying out file systems with bsdlabel(8) or sysinstall(8), remember that hard drives transfer data faster from the outer tracks to the inner. Thus smaller and heavier-accessed file systems should be closer to the outside of the drive, while larger partitions like `/usr` should be placed toward the inner parts of the disk. It is a good idea to create partitions in an order similar to: root, swap, `/var`, `/usr`.

The size of the `/var` partition reflects the intended machine usage. The `/var` file system is used to hold mailboxes, log files, and printer spools. Mailboxes and log files can grow to unexpected sizes depending on how many users exist and how long log files are kept. Most users will rarely need more than about a gigabyte of free disk space in `/var`.

Note: There are a few times that a lot of disk space is required in `/var/tmp`. When new software is installed with pkg_add(1) the packaging tools extract a temporary copy of the packages under `/var/tmp`. Large software packages, like **Firefox**, or **OpenOffice** may be tricky to install if there is not enough disk space under `/var/tmp`.

The `/usr` partition holds many of the files required to support the system, including the ports(7) collection (recommended) and the source code (optional). Both the ports and the sources of the base system are optional at install time, but we recommend at least 2 gigabytes for this partition.

When selecting partition sizes, keep the space requirements in mind. Running out of space in one partition while barely using another can be a hassle.

Note: Some users have found that sysinstall(8)'s `Auto-defaults` partition sizer will sometimes select smaller than adequate `/var` and `/` partitions. Partition wisely and generously.

11.2.1.2 Swap Partition

As a rule of thumb, the swap partition should be about double the size of system memory (RAM). For example, if the machine has 128 megabytes of memory, the swap file should be 256 megabytes. Systems with less memory may perform better with more swap. Less than 256 megabytes of swap is not recommended and memory expansion should be considered. The kernel's VM paging algorithms are tuned to perform best when the swap partition is at least two times the size of main memory. Configuring too little swap can lead to inefficiencies in the VM page scanning code and might create issues later if more memory is added.

On larger systems with multiple SCSI disks (or multiple IDE disks operating on different controllers), it is recommend that a swap is configured on each drive (up to four drives). The swap partitions should be approximately the same size. The kernel can handle arbitrary sizes but internal data structures scale to 4 times the largest swap partition. Keeping the swap partitions near the same size will allow the kernel to optimally stripe swap space across disks. Large swap sizes are fine, even if swap is not used much. It might be easier to recover from a runaway program before being forced to reboot.

11.2.1.3 Why Partition?

Several users think a single large partition will be fine, but there are several reasons why this is a bad idea. First, each partition has different operational characteristics and separating them allows the file system to tune accordingly. For example, the root and `/usr` partitions are read-mostly, without much writing. While a lot of reading and writing could occur in `/var` and `/var/tmp`.

By properly partitioning a system, fragmentation introduced in the smaller write heavy partitions will not bleed over into the mostly-read partitions. Keeping the write-loaded partitions closer to the disk's edge, will increase I/O performance in the partitions where it occurs the most. Now while I/O performance in the larger partitions may be needed, shifting them more toward the edge of the disk will not lead to a significant performance improvement over moving `/var` to the edge. Finally, there are safety concerns. A smaller, neater root partition which is mostly read-only has a greater chance of surviving a bad crash.

11.3 Core Configuration

The principal location for system configuration information is within `/etc/rc.conf`. This file contains a wide range of configuration information, principally used at system startup to configure the system. Its name directly implies this; it is configuration information for the `rc*` files.

An administrator should make entries in the `rc.conf` file to override the default settings from `/etc/defaults/rc.conf`. The defaults file should not be copied verbatim to `/etc` - it contains default values, not examples. All system-specific changes should be made in the `rc.conf` file itself.

A number of strategies may be applied in clustered applications to separate site-wide configuration from system-specific configuration in order to keep administration overhead down. The recommended approach is to place site-wide configuration into another file, such as `/etc/rc.conf.site`, and then include this file into `/etc/rc.conf`, which will contain only system-specific information.

As `rc.conf` is read by sh(1) it is trivial to achieve this. For example:

- 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"
```

The `rc.conf.site` file can then be distributed to every system using `rsync` or a similar program, while the `rc.conf` file remains unique.

Upgrading the system using sysinstall(8) or `make world` will not overwrite the `rc.conf` file, so system configuration information will not be lost.

11.4 Application Configuration

Typically, installed applications have their own configuration files, with their own syntax, etc. It is important that these files be kept separate from the base system, so that they may be easily located and managed by the package management tools.

Typically, these files are installed in `/usr/local/etc`. In the case where an application has a large number of configuration files, a subdirectory will be created to hold them.

Normally, when a port or package is installed, sample configuration files are also installed. These are usually identified with a `.default` suffix. If there are no existing configuration files for the application, they will be created by copying the `.default` files.

For example, consider the contents of the directory `/usr/local/etc/apache`:

```
-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
```

The file sizes show that only the `srm.conf` file has been changed. A later update of the **Apache** port would not overwrite this changed file.

11.5 Starting Services

Many users choose to install third party software on FreeBSD from the Ports Collection. In many of these situations it may be necessary to configure the software in a manner which will allow it to be started upon system initialization. Services, such as `mail/postfix` or `www/apache13` are just two of the many software packages which may be started during system initialization. This section explains the procedures available for starting third party software.

In FreeBSD, most included services, such as cron(8), are started through the system start up scripts. These scripts may differ depending on FreeBSD or vendor version; however, the most important aspect to consider is that their start up configuration can be handled through simple startup scripts.

11.5.1 Extended Application Configuration

Now that FreeBSD includes `rc.d`, configuration of application startup has become easier, and more featureful. Using the key words discussed in the rc.d section, applications may now be set to start after certain other services for example DNS; may permit extra flags to be passed through `rc.conf` in place of hard coded flags in the start up script, etc. A basic script may look similar to the following:

```
#!/bin/sh
#
# PROVIDE: utility
# REQUIRE: DAEMON
# KEYWORD: shutdown

. /etc/rc.subr

name="utility"
rcvar=‘set_rcvar‘
command="/usr/local/sbin/utility"

load_rc_config $name

#
# DO NOT CHANGE THESE DEFAULT VALUES HERE
# SET THEM IN THE /etc/rc.conf FILE
#
utility_enable=${utility_enable-"NO"}
utility_pidfile=${utility_pidfile-"/var/run/utility.pid"}

pidfile="${utility_pidfile}"
```

```
run_rc_command "$1"
```

This script will ensure that the provided **utility** will be started after the `DAEMON` pseudo-service. It also provides a method for setting and tracking the PID, or process ID file.

This application could then have the following line placed in `/etc/rc.conf`:

```
utility_enable="YES"
```

This method also allows for easier manipulation of the command line arguments, inclusion of the default functions provided in `/etc/rc.subr`, compatibility with the rcorder(8) utility and provides for easier configuration via the `rc.conf` file.

11.5.2 Using Services to Start Services

Other services, such as POP3 server daemons, IMAP, etc. could be started using inetd(8). This involves installing the service utility from the Ports Collection with a configuration line added to the `/etc/inetd.conf` file, or by uncommenting one of the current configuration lines. Working with **inetd** and its configuration is described in depth in the inetd section.

In some cases it may make more sense to use the cron(8) daemon to start system services. This approach has a number of advantages because `cron` runs these processes as the `crontab`'s file owner. This allows regular users to start and maintain some applications.

The `cron` utility provides a unique feature, `@reboot`, which may be used in place of the time specification. This will cause the job to be run when cron(8) is started, normally during system initialization.

11.6 Configuring the `cron` Utility

One of the most useful utilities in FreeBSD is cron(8). The `cron` utility runs in the background and constantly checks the `/etc/crontab` file. The `cron` utility also checks the `/var/cron/tabs` directory, in search of new `crontab` files. These `crontab` files store information about specific functions which `cron` is supposed to perform at certain times.

The `cron` utility uses two different types of configuration files, the system crontab and user crontabs. The only difference between these two formats is the sixth field. In the system crontab, the sixth field is the name of a user for the command to run as. This gives the system crontab the ability to run commands as any user. In a user crontab, the sixth field is the command to run, and all commands run as the user who created the crontab; this is an important security feature.

> **Note:** User crontabs allow individual users to schedule tasks without the need for `root` privileges. Commands in a user's crontab run with the permissions of the user who owns the crontab.
>
> The `root` user can have a user crontab just like any other user. This one is different from `/etc/crontab` (the system crontab). Because of the system crontab, there is usually no need to create a user crontab for `root`.

Let us take a look at the `/etc/crontab` file (the system crontab):

```
# /etc/crontab - root's crontab for FreeBSD
```

```
#
# $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 mday month wday who command ❸
#
#
*/5 * * * * root /usr/libexec/atrun ❹
```

❶ Like most FreeBSD configuration files, the # character represents a comment. A comment can be placed in the file as a reminder of what and why a desired action is performed. Comments cannot be on the same line as a command or else they will be interpreted as part of the command; they must be on a new line. Blank lines are ignored.

❷ First, the environment must be defined. The equals (=) character is used to define any environment settings, as with this example where it is used for the `SHELL`, `PATH`, and `HOME` options. If the shell line is omitted, `cron` will use the default, which is `sh`. If the `PATH` variable is omitted, no default will be used and file locations will need to be absolute. If `HOME` is omitted, `cron` will use the invoking users home directory.

❸ This line defines a total of seven fields. Listed here are the values `minute`, `hour`, `mday`, `month`, `wday`, `who`, and `command`. These are almost all self explanatory. `minute` is the time in minutes the command will be run. `hour` is similar to the `minute` option, just in hours. `mday` stands for day of the month. `month` is similar to `hour` and `minute`, as it designates the month. The `wday` option stands for day of the week. All these fields must be numeric values, and follow the twenty-four hour clock. The `who` field is special, and only exists in the `/etc/crontab` file. This field specifies which user the command should be run as. When a user installs his or her `crontab` file, they will not have this option. Finally, the `command` option is listed. This is the last field, so naturally it should designate the command to be executed.

❹ This last line will define the values discussed above. Notice here we have a `*/5` listing, followed by several more `*` characters. These `*` characters mean “first-last”, and can be interpreted as *every* time. So, judging by this line, it is apparent that the `atrun` command is to be invoked by `root` every five minutes regardless of what day or month it is. For more information on the `atrun` command, see the atrun(8) manual page.

Commands can have any number of flags passed to them; however, commands which extend to multiple lines need to be broken with the backslash “\” continuation character.

This is the basic setup for every `crontab` file, although there is one thing different about this one. Field number six, where we specified the username, only exists in the system `/etc/crontab` file. This field should be omitted for individual user `crontab` files.

11.6.1 Installing a Crontab

Important: You must not use the procedure described here to edit/install the system crontab. Simply use your favorite editor: the `cron` utility will notice that the file has changed and immediately begin using the updated version. See this FAQ entry

(http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/admin.html#ROOT-NOT-FOUND-CRON-ERRORS) for more information.

To install a freshly written user `crontab`, first use your favorite editor to create a file in the proper format, and then use the `crontab` utility. The most common usage is:

```
% crontab crontab-file
```

In this example, `crontab-file` is the filename of a `crontab` that was previously created.

There is also an option to list installed `crontab` files: just pass the `-l` option to `crontab` and look over the output.

For users who wish to begin their own crontab file from scratch, without the use of a template, the `crontab -e` option is available. This will invoke the selected editor with an empty file. When the file is saved, it will be automatically installed by the `crontab` command.

If you later want to remove your user `crontab` completely, use `crontab` with the `-r` option.

11.7 Using rc under FreeBSD

In 2002 FreeBSD integrated the NetBSD `rc.d` system for system initialization. Users should notice the files listed in the `/etc/rc.d` directory. Many of these files are for basic services which can be controlled with the `start`, `stop`, and `restart` options. For instance, sshd(8) can be restarted with the following command:

```
# /etc/rc.d/sshd restart
```

This procedure is similar for other services. Of course, services are usually started automatically at boot time as specified in rc.conf(5). For example, enabling the Network Address Translation daemon at startup is as simple as adding the following line to `/etc/rc.conf`:

```
natd_enable="YES"
```

If a `natd_enable="NO"` line is already present, then simply change the `NO` to `YES`. The rc scripts will automatically load any other dependent services during the next reboot, as described below.

Since the `rc.d` system is primarily intended to start/stop services at system startup/shutdown time, the standard `start`, `stop` and `restart` options will only perform their action if the appropriate `/etc/rc.conf` variables are set. For instance the above `sshd restart` command will only work if `sshd_enable` is set to `YES` in `/etc/rc.conf`. To `start`, `stop` or `restart` a service regardless of the settings in `/etc/rc.conf`, the commands should be prefixed with “one”. For instance to restart `sshd` regardless of the current `/etc/rc.conf` setting, execute the following command:

```
# /etc/rc.d/sshd onerestart
```

It is easy to check if a service is enabled in `/etc/rc.conf` by running the appropriate `rc.d` script with the option `rcvar`. Thus, an administrator can check that `sshd` is in fact enabled in `/etc/rc.conf` by running:

```
# /etc/rc.d/sshd rcvar
# sshd
$sshd_enable=YES
```

Note: The second line (`# sshd`) is the output from the `sshd` command, not a `root` console.

To determine if a service is running, a `status` option is available. For instance to verify that `sshd` is actually started:

```
# /etc/rc.d/sshd status
sshd is running as pid 433.
```

In some cases it is also possible to `reload` a service. This will attempt to send a signal to an individual service, forcing the service to reload its configuration files. In most cases this means sending the service a `SIGHUP` signal. Support for this feature is not included for every service.

The `rc.d` system is not only used for network services, it also contributes to most of the system initialization. For instance, consider the `bgfsck` file. When this script is executed, it will print out the following message:

```
Starting background file system checks in 60 seconds.
```

Therefore this file is used for background file system checks, which are done only during system initialization.

Many system services depend on other services to function properly. For example, NIS and other RPC-based services may fail to start until after the `rpcbind` (portmapper) service has started. To resolve this issue, information about dependencies and other meta-data is included in the comments at the top of each startup script. The rcorder(8) program is then used to parse these comments during system initialization to determine the order in which system services should be invoked to satisfy the dependencies.

The following words must be included in all startup scripts (they are required by rc.subr(8) to “enable” the startup script):

- `PROVIDE`: Specifies the services this file provides.

The following words may be included at the top of each startup file. They are not strictly necessary, but they are useful as hints to rcorder(8):

- `REQUIRE`: Lists services which are required for this service. This file will run *after* the specified services.
- `BEFORE`: Lists services which depend on this service. This file will run *before* the specified services.

By carefully setting these keywords for each startup script, an administrator has a very fine-grained level of control of the startup order of the scripts, without the hassle of “runlevels” like some other UNIX operating systems.

Additional information about the `rc.d` system can be found in the rc(8) and rc.subr(8) manual pages. If you are interested in writing your own `rc.d` scripts or improving the existing ones, you may find this article (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/rc-scripting) also useful.

11.8 Setting Up Network Interface Cards

Nowadays we can not think about a computer without thinking about a network connection. Adding and configuring a network card is a common task for any FreeBSD administrator.

11.8.1 Locating the Correct Driver

Before you begin, you should know the model of the card you have, the chip it uses, and whether it is a PCI or ISA card. FreeBSD supports a wide variety of both PCI and ISA cards. Check the Hardware Compatibility List for your release to see if your card is supported.

Once you are sure your card is supported, you need to determine the proper driver for the card. `/usr/src/sys/conf/NOTES` and `/usr/src/sys/`*`arch`*`/conf/NOTES` will give you the list of network interface drivers with some information about the supported chipsets/cards. If you have doubts about which driver is the correct one, read the manual page of the driver. The manual page will give you more information about the supported hardware and even the possible problems that could occur.

If you own a common card, most of the time you will not have to look very hard for a driver. Drivers for common network cards are present in the `GENERIC` kernel, so your card should show up during boot, like so:

```
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]
```

In this example, we see that two cards using the dc(4) driver are present on the system.

If the driver for your NIC is not present in `GENERIC`, you will need to load the proper driver to use your NIC. This may be accomplished in one of two ways:

- The easiest way is to simply load a kernel module for your network card with kldload(8), or automatically at boot time by adding the appropriate line to the file `/boot/loader.conf`. Not all NIC drivers are available as modules; notable examples of devices for which modules do not exist are ISA cards.
- Alternatively, you may statically compile the support for your card into your kernel. Check `/usr/src/sys/conf/NOTES`, `/usr/src/sys/`*`arch`*`/conf/NOTES` and the manual page of the driver to know what to add in your kernel configuration file. For more information about recompiling your kernel, please see Chapter 8. If your card was detected at boot by your kernel (`GENERIC`) you do not have to build a new kernel.

11.8.1.1 Using Windows NDIS Drivers

Unfortunately, there are still many vendors that do not provide schematics for their drivers to the open source community because they regard such information as trade secrets. Consequently, the developers of FreeBSD and other operating systems are left two choices: develop the drivers by a long and pain-staking process of reverse engineering or using the existing driver binaries available for the Microsoft Windows platforms. Most developers, including those involved with FreeBSD, have taken the latter approach.

Thanks to the contributions of Bill Paul (wpaul) there is "native" support for the Network Driver Interface Specification (NDIS). The FreeBSD NDISulator (otherwise known as Project Evil) takes a Windows driver binary and basically tricks it into thinking it is running on Windows. Because the ndis(4) driver is using a Windows binary, it is only usable on i386 and amd64 systems.

> **Note:** The ndis(4) driver is designed to support mainly PCI, CardBus and PCMCIA devices, USB devices are not yet supported.

In order to use the NDISulator, you need three things:

1. Kernel sources
2. Windows XP driver binary (`.SYS` extension)
3. Windows XP driver configuration file (`.INF` extension)

Locate the files for your specific card. Generally, they can be found on the included CDs or at the vendors' websites. In the following examples, we will use `W32DRIVER.SYS` and `W32DRIVER.INF`.

> **Note:** You can not use a Windows/i386 driver with FreeBSD/amd64, you must get a Windows/amd64 driver to make it work properly.

The next step is to compile the driver binary into a loadable kernel module. To accomplish this, as `root`, use ndisgen(8):

```
# ndisgen /path/to/W32DRIVER.INF /path/to/W32DRIVER.SYS
```

The ndisgen(8) utility is interactive and will prompt for any extra information it requires; it will produce a kernel module in the current directory which can be loaded as follows:

```
# kldload ./W32DRIVER_SYS.ko
```

In addition to the generated kernel module, you must load the `ndis.ko` and `if_ndis.ko` modules. This should be automatically done when you load any module that depends on ndis(4). If you want to load them manually, use the following commands:

```
# kldload ndis
# kldload if_ndis
```

The first command loads the NDIS miniport driver wrapper, the second loads the actual network interface.

Now, check dmesg(8) to see if there were any errors loading. If all went well, you should get output resembling the following:

```
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
```

From here you can treat the `ndis0` device like any other network interface (e.g., `dc0`).

You can configure the system to load the NDIS modules at boot time in the same way as with any other module. First, copy the generated module, `W32DRIVER_SYS.ko`, to the `/boot/modules` directory. Then, add the following line to `/boot/loader.conf`:

```
W32DRIVER_SYS_load="YES"
```

11.8.2 Configuring the Network Card

Once the right driver is loaded for the network card, the card needs to be configured. As with many other things, the network card may have been configured at installation time by **sysinstall**.

To display the configuration for the network interfaces on your system, enter the following command:

```
% 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 0xffffff00 broadcast 192.168.1.255
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
dc1: flags=8802<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 0xffffff00 broadcast 10.0.0.255
        media: Ethernet 10baseT/UTP
        status: no carrier
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> metric 0 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=3<PERFORMNUD,ACCEPT_RTADV>
```

In this example, the following devices were displayed:

- `dc0`: The first Ethernet interface
- `dc1`: The second Ethernet interface
- `plip0`: The parallel port interface (if a parallel port is present on the machine)
- `lo0`: The loopback device

FreeBSD uses the driver name followed by the order in which one the card is detected at the kernel boot to name the network card. For example `sis2` would be the third network card on the system using the sis(4) driver.

In this example, the `dc0` device is up and running. The key indicators are:

1. `UP` means that the card is configured and ready.
2. The card has an Internet (`inet`) address (in this case `192.168.1.3`).

3. It has a valid subnet mask (`netmask`; `0xffffff00` is the same as `255.255.255.0`).
4. It has a valid broadcast address (in this case, `192.168.1.255`).
5. The MAC address of the card (`ether`) is `00:a0:cc:da:da:da`
6. The physical media selection is on autoselection mode (`media: Ethernet autoselect (100baseTX <full-duplex>)`). We see that `dc1` was configured to run with `10baseT/UTP` media. For more information on available media types for a driver, please refer to its manual page.
7. The status of the link (`status`) is `active`, i.e. the carrier is detected. For `dc1`, we see `status: no carrier`. This is normal when an Ethernet cable is not plugged into the card.

If the ifconfig(8) output had shown something similar to:

```
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
```

it would indicate the card has not been configured.

To configure your card, you need `root` privileges. The network card configuration can be done from the command line with ifconfig(8) but you would have to do it after each reboot of the system. The file `/etc/rc.conf` is where to add the network card's configuration.

Open `/etc/rc.conf` in your favorite editor. You need to add a line for each network card present on the system, for example in our case, we added these lines:

```
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"
```

You have to replace `dc0`, `dc1`, and so on, with the correct device for your cards, and the addresses with the proper ones. You should read the card driver and ifconfig(8) manual pages for more details about the allowed options and also rc.conf(5) manual page for more information on the syntax of `/etc/rc.conf`.

If you configured the network during installation, some lines about the network card(s) may be already present. Double check `/etc/rc.conf` before adding any lines.

You will also have to edit the file `/etc/hosts` to add the names and the IP addresses of various machines of the LAN, if they are not already there. For more information please refer to hosts(5) and to `/usr/share/examples/etc/hosts`.

> **Note:** If access to the Internet is planned with the machine, you also have to manually set up the default gateway and the nameserver:
>
> ```
> # echo 'defaultrouter="your_default_router"' >> /etc/rc.conf
> # echo 'nameserver your_DNS_server' >> /etc/resolv.conf
> ```

11.8.3 Testing and Troubleshooting

Once you have made the necessary changes in `/etc/rc.conf`, you should reboot your system. This will allow the change(s) to the interface(s) to be applied, and verify that the system restarts without any configuration errors. Alternatively you can just relaunch the networking system:

```
# /etc/rc.d/netif restart
```

> **Note:** If a default gateway has been set in `/etc/rc.conf`, use also this command:
>
> ```
> # /etc/rc.d/routing restart
> ```

Once the networking system has been relaunched, you should test the network interfaces.

11.8.3.1 Testing the Ethernet Card

To verify that an Ethernet card is configured correctly, you have to try two things. First, ping the interface itself, and then ping another machine on the LAN.

First test the local interface:

```
% 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
```

Now we have to ping another machine on the LAN:

```
% 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
```

You could also use the machine name instead of `192.168.1.2` if you have set up the `/etc/hosts` file.

11.8.3.2 Troubleshooting

Troubleshooting hardware and software configurations is always a pain, and a pain which can be alleviated by checking the simple things first. Is your network cable plugged in? Have you properly configured the network services? Did you configure the firewall correctly? Is the card you are using supported by FreeBSD? Always check the hardware notes before sending off a bug report. Update your version of FreeBSD to the latest STABLE version. Check the mailing list archives, or perhaps search the Internet.

If the card works, yet performance is poor, it would be worthwhile to read over the tuning(7) manual page. You can also check the network configuration as incorrect network settings can cause slow connections.

Some users experience one or two `device timeout` messages, which is normal for some cards. If they continue, or are bothersome, you may wish to be sure the device is not conflicting with another device. Double check the cable connections. Perhaps you may just need to get another card.

At times, users see a few `watchdog timeout` errors. The first thing to do here is to check your network cable. Many cards require a PCI slot which supports Bus Mastering. On some old motherboards, only one PCI slot allows it (usually slot 0). Check the network card and the motherboard documentation to determine if that may be the problem.

`No route to host` messages occur if the system is unable to route a packet to the destination host. This can happen if no default route is specified, or if a cable is unplugged. Check the output of `netstat -rn` and make sure there is a valid route to the host you are trying to reach. If there is not, read on to Chapter 31.

`ping: sendto: Permission denied` error messages are often caused by a misconfigured firewall. If `ipfw` is enabled in the kernel but no rules have been defined, then the default policy is to deny all traffic, even ping requests! Read on to Chapter 30 for more information.

Sometimes performance of the card is poor, or below average. In these cases it is best to set the media selection mode from `autoselect` to the correct media selection. While this usually works for most hardware, it may not resolve this issue for everyone. Again, check all the network settings, and read over the tuning(7) manual page.

11.9 Virtual Hosts

A very common use of FreeBSD is virtual site hosting, where one server appears to the network as many servers. This is achieved by assigning multiple network addresses to a single interface.

A given network interface has one “real” address, and may have any number of “alias” addresses. These aliases are normally added by placing alias entries in `/etc/rc.conf`.

An alias entry for the interface `fxp0` looks like:

```
ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

Note that alias entries must start with `alias0` and proceed upwards in order, (for example, `_alias1`, `_alias2`, and so on). The configuration process will stop at the first missing number.

The calculation of alias netmasks is important, but fortunately quite simple. For a given interface, there must be one address which correctly represents the network’s netmask. Any other addresses which fall within this network must have a netmask of all `1`s (expressed as either `255.255.255.255` or `0xffffffff`).

For example, consider the case where the `fxp0` interface is connected to two networks, the `10.1.1.0` network with a netmask of `255.255.255.0` and the `202.0.75.16` network with a netmask of `255.255.255.240`. We want the system to appear at `10.1.1.1` through `10.1.1.5` and at `202.0.75.17` through `202.0.75.20`. As noted above,

only the first address in a given network range (in this case, 10.0.1.1 and 202.0.75.17) should have a real netmask; all the rest (10.1.1.2 through 10.1.1.5 and 202.0.75.18 through 202.0.75.20) must be configured with a netmask of 255.255.255.255.

The following /etc/rc.conf entries configure the adapter correctly for this arrangement:

```
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 Configuration Files

11.10.1 /etc Layout

There are a number of directories in which configuration information is kept. These include:

/etc	Generic system configuration information; data here is system-specific.
/etc/defaults	Default versions of system configuration files.
/etc/mail	Extra sendmail(8) configuration, other MTA configuration files.
/etc/ppp	Configuration for both user- and kernel-ppp programs.
/etc/namedb	Default location for named(8) data. Normally named.conf and zone files are stored here.
/usr/local/etc	Configuration files for installed applications. May contain per-application subdirectories.
/usr/local/etc/rc.d	Start/stop scripts for installed applications.
/var/db	Automatically generated system-specific database files, such as the package database, the locate database, and so on

11.10.2 Hostnames

11.10.2.1 /etc/resolv.conf

/etc/resolv.conf dictates how FreeBSD's resolver accesses the Internet Domain Name System (DNS).

The most common entries to resolv.conf are:

nameserver	The IP address of a name server the resolver should query. The servers are queried in the order listed with a maximum of three.

`search`	Search list for hostname lookup. This is normally determined by the domain of the local hostname.
`domain`	The local domain name.

A typical `resolv.conf`:

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```

Note: Only one of the `search` and `domain` options should be used.

If you are using DHCP, dhclient(8) usually rewrites `resolv.conf` with information received from the DHCP server.

11.10.2.2 `/etc/hosts`

`/etc/hosts` is a simple text database reminiscent of the old Internet. It works in conjunction with DNS and NIS providing name to IP address mappings. Local computers connected via a LAN can be placed in here for simplistic naming purposes instead of setting up a named(8) server. Additionally, `/etc/hosts` can be used to provide a local record of Internet names, reducing the need to query externally for commonly accessed names.

```
# $FreeBSD$
#
#
# Host Database
#
# This file should contain the addresses and aliases for local hosts that
# share this file.  Replace 'my.domain' below with the domainname of your
# machine.
#
# In the presence of the domain name service or NIS, this file may
# not be consulted at all; see /etc/nsswitch.conf for the resolution order.
#
#
::1   localhost localhost.my.domain
127.0.0.1  localhost localhost.my.domain
#
# Imaginary network.
#10.0.0.2  myname.my.domain myname
#10.0.0.3  myfriend.my.domain myfriend
#
# According to RFC 1918, you can use the following IP networks for
# private nets which will never be connected to the Internet:
#
# 10.0.0.0 -   10.255.255.255
# 172.16.0.0 -   172.31.255.255
# 192.168.0.0 -   192.168.255.255
#
```

```
# In case you want to be able to connect to the Internet, you need
# real official assigned numbers.  Do not try to invent your own network
# numbers but instead get one from your network provider (if any) or
# from your regional registry (ARIN, APNIC, LACNIC, RIPE NCC, or AfriNIC.)
#
```

`/etc/hosts` takes on the simple format of:

```
[Internet address] [official hostname] [alias1] [alias2] ...
```

For example:

```
10.0.0.1 myRealHostname.example.com myRealHostname foobar1 foobar2
```

Consult hosts(5) for more information.

11.10.3 Log File Configuration

11.10.3.1 `syslog.conf`

`syslog.conf` is the configuration file for the syslogd(8) program. It indicates which types of `syslog` messages are logged to particular log files.

```
# $FreeBSD$
#
#       Spaces ARE valid field separators in this file. However,
#       other *nix-like systems still insist on using tabs as field
#       separators. If you are sharing this file between systems, you
#       may want to use only tabs as field separators here.
#       Consult the syslog.conf(5) manual page.
*.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                                         *
# uncomment this to log all writes to /dev/console to /var/log/console.log
#console.info                                   /var/log/console.log
# uncomment this to enable logging of all log messages to /var/log/all.log
#*.*                                            /var/log/all.log
# uncomment this to enable logging to a remote log host named loghost
#*.*                                            @loghost
# uncomment these if you’re running inn
# 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
```

Consult the syslog.conf(5) manual page for more information.

11.10.3.2 `newsyslog.conf`

`newsyslog.conf` is the configuration file for newsyslog(8), a program that is normally scheduled to run by cron(8). newsyslog(8) determines when log files require archiving or rearranging. `logfile` is moved to `logfile.0`, `logfile.0` is moved to `logfile.1`, and so on. Alternatively, the log files may be archived in gzip(1) format causing them to be named: `logfile.0.gz`, `logfile.1.gz`, and so on.

`newsyslog.conf` indicates which log files are to be managed, how many are to be kept, and when they are to be touched. Log files can be rearranged and/or archived when they have either reached a certain size, or at a certain periodic time/date.

```
# configuration file for newsyslog
# $FreeBSD$
#
# filename          [owner:group]    mode count size when [ZB] [/pid_file] [sig_num]
/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
```

Consult the newsyslog(8) manual page for more information.

11.10.4 `sysctl.conf`

`sysctl.conf` looks much like `rc.conf`. Values are set in a `variable=value` form. The specified values are set after the system goes into multi-user mode. Not all variables are settable in this mode.

To turn off logging of fatal signal exits and prevent users from seeing processes started from other users, the following tunables can be set in `sysctl.conf`:

```
# Do not log fatal signal exits (e.g. sig 11)
kern.logsigexit=0
```

```
# Prevent users from seeing information about processes that
# are being run under another UID.
security.bsd.see_other_uids=0
```

11.11 Tuning with sysctl

sysctl(8) is an interface that allows you to make changes to a running FreeBSD system. This includes many advanced options of the TCP/IP stack and virtual memory system that can dramatically improve performance for an experienced system administrator. Over five hundred system variables can be read and set using sysctl(8).

At its core, sysctl(8) serves two functions: to read and to modify system settings.

To view all readable variables:

```
% sysctl -a
```

To read a particular variable, for example, `kern.maxproc`:

```
% sysctl kern.maxproc
kern.maxproc: 1044
```

To set a particular variable, use the intuitive *`variable=value`* syntax:

```
# sysctl kern.maxfiles=5000
kern.maxfiles: 2088 -> 5000
```

Settings of sysctl variables are usually either strings, numbers, or booleans (a boolean being `1` for yes or a `0` for no).

If you want to set automatically some variables each time the machine boots, add them to the `/etc/sysctl.conf` file. For more information see the sysctl.conf(5) manual page and the Section 11.10.4.

11.11.1 sysctl(8) Read-only

In some cases it may be desirable to modify read-only sysctl(8) values. While this is sometimes unavoidable, it can only be done on (re)boot.

For instance on some laptop models the cardbus(4) device will not probe memory ranges, and fail with errors which look similar to:

```
cbb0: Could not map register memory
device_probe_and_attach: cbb0 attach returned 12
```

Cases like the one above usually require the modification of some default sysctl(8) settings which are set read only. To overcome these situations a user can put sysctl(8) “OIDs” in their local `/boot/loader.conf`. Default settings are located in the `/boot/defaults/loader.conf` file.

Fixing the problem mentioned above would require a user to set `hw.pci.allow_unsupported_io_range=1` in the aforementioned file. Now cardbus(4) will work properly.

11.12 Tuning Disks

11.12.1 Sysctl Variables

11.12.1.1 `vfs.vmiodirenable`

The `vfs.vmiodirenable` sysctl variable may be set to either 0 (off) or 1 (on); it is 1 by default. This variable controls how directories are cached by the system. Most directories are small, using just a single fragment (typically 1 K) in the file system and less (typically 512 bytes) in the buffer cache. With this variable turned off (to 0), the buffer cache will only cache a fixed number of directories even if you have a huge amount of memory. When turned on (to 1), this sysctl allows the buffer cache to use the VM Page Cache to cache the directories, making all the memory available for caching directories. However, the minimum in-core memory used to cache a directory is the physical page size (typically 4 K) rather than 512 bytes. We recommend keeping this option on if you are running any services which manipulate large numbers of files. Such services can include web caches, large mail systems, and news systems. Keeping this option on will generally not reduce performance even with the wasted memory but you should experiment to find out.

11.12.1.2 `vfs.write_behind`

The `vfs.write_behind` sysctl variable defaults to `1` (on). This tells the file system to issue media writes as full clusters are collected, which typically occurs when writing large sequential files. The idea is to avoid saturating the buffer cache with dirty buffers when it would not benefit I/O performance. However, this may stall processes and under certain circumstances you may wish to turn it off.

11.12.1.3 `vfs.hirunningspace`

The `vfs.hirunningspace` sysctl variable determines how much outstanding write I/O may be queued to disk controllers system-wide at any given instance. The default is usually sufficient but on machines with lots of disks you may want to bump it up to four or five *megabytes*. Note that setting too high a value (exceeding the buffer cache's write threshold) can lead to extremely bad clustering performance. Do not set this value arbitrarily high! Higher write values may add latency to reads occurring at the same time.

There are various other buffer-cache and VM page cache related sysctls. We do not recommend modifying these values, the VM system does an extremely good job of automatically tuning itself.

11.12.1.4 `vm.swap_idle_enabled`

The `vm.swap_idle_enabled` sysctl variable is useful in large multi-user systems where you have lots of users entering and leaving the system and lots of idle processes. Such systems tend to generate a great deal of continuous pressure on free memory reserves. Turning this feature on and tweaking the swapout hysteresis (in idle seconds) via `vm.swap_idle_threshold1` and `vm.swap_idle_threshold2` allows you to depress the priority of memory pages associated with idle processes more quickly then the normal pageout algorithm. This gives a helping hand to the pageout daemon. Do not turn this option on unless you need it, because the tradeoff you are making is essentially pre-page memory sooner rather than later; thus eating more swap and disk bandwidth. In a small system this option will have a determinable effect but in a large system that is already doing moderate paging this option allows the VM system to stage whole processes into and out of memory easily.

11.12.1.5 `hw.ata.wc`

FreeBSD 4.3 flirted with turning off IDE write caching. This reduced write bandwidth to IDE disks but was considered necessary due to serious data consistency issues introduced by hard drive vendors. The problem is that IDE drives lie about when a write completes. With IDE write caching turned on, IDE hard drives not only write data to disk out of order, but will sometimes delay writing some blocks indefinitely when under heavy disk loads. A crash or power failure may cause serious file system corruption. FreeBSD's default was changed to be safe. Unfortunately, the result was such a huge performance loss that we changed write caching back to on by default after the release. You should check the default on your system by observing the `hw.ata.wc` sysctl variable. If IDE write caching is turned off, you can turn it back on by setting the kernel variable back to 1. This must be done from the boot loader at boot time. Attempting to do it after the kernel boots will have no effect.

For more information, please see ata(4).

11.12.1.6 `SCSI_DELAY` (`kern.cam.scsi_delay`)

The `SCSI_DELAY` kernel config may be used to reduce system boot times. The defaults are fairly high and can be responsible for `15` seconds of delay in the boot process. Reducing it to `5` seconds usually works (especially with modern drives). The `kern.cam.scsi_delay` boot time tunable should be used. The tunable, and kernel config option accept values in terms of *milliseconds* and *not seconds*.

11.12.2 Soft Updates

The tunefs(8) program can be used to fine-tune a file system. This program has many different options, but for now we are only concerned with toggling Soft Updates on and off, which is done by:

```
# tunefs -n enable /filesystem
# tunefs -n disable /filesystem
```

A filesystem cannot be modified with tunefs(8) while it is mounted. A good time to enable Soft Updates is before any partitions have been mounted, in single-user mode.

Soft Updates drastically improves meta-data performance, mainly file creation and deletion, through the use of a memory cache. We recommend to use Soft Updates on all of your file systems. There are two downsides to Soft Updates that you should be aware of: First, Soft Updates guarantees filesystem consistency in the case of a crash but could very easily be several seconds (even a minute!) behind updating the physical disk. If your system crashes you may lose more work than otherwise. Secondly, Soft Updates delays the freeing of filesystem blocks. If you have a filesystem (such as the root filesystem) which is almost full, performing a major update, such as `make installworld`, can cause the filesystem to run out of space and the update to fail.

11.12.2.1 More Details about Soft Updates

There are two traditional approaches to writing a file systems meta-data back to disk. (Meta-data updates are updates to non-content data like inodes or directories.)

Historically, the default behavior was to write out meta-data updates synchronously. If a directory had been changed, the system waited until the change was actually written to disk. The file data buffers (file contents) were passed through the buffer cache and backed up to disk later on asynchronously. The advantage of this implementation is that it operates safely. If there is a failure during an update, the meta-data are always in a consistent state. A file is either

created completely or not at all. If the data blocks of a file did not find their way out of the buffer cache onto the disk by the time of the crash, fsck(8) is able to recognize this and repair the filesystem by setting the file length to 0. Additionally, the implementation is clear and simple. The disadvantage is that meta-data changes are slow. An `rm -r`, for instance, touches all the files in a directory sequentially, but each directory change (deletion of a file) will be written synchronously to the disk. This includes updates to the directory itself, to the inode table, and possibly to indirect blocks allocated by the file. Similar considerations apply for unrolling large hierarchies (`tar -x`).

The second case is asynchronous meta-data updates. This is the default for Linux/ext2fs and `mount -o async` for *BSD ufs. All meta-data updates are simply being passed through the buffer cache too, that is, they will be intermixed with the updates of the file content data. The advantage of this implementation is there is no need to wait until each meta-data update has been written to disk, so all operations which cause huge amounts of meta-data updates work much faster than in the synchronous case. Also, the implementation is still clear and simple, so there is a low risk for bugs creeping into the code. The disadvantage is that there is no guarantee at all for a consistent state of the filesystem. If there is a failure during an operation that updated large amounts of meta-data (like a power failure, or someone pressing the reset button), the filesystem will be left in an unpredictable state. There is no opportunity to examine the state of the filesystem when the system comes up again; the data blocks of a file could already have been written to the disk while the updates of the inode table or the associated directory were not. It is actually impossible to implement a `fsck` which is able to clean up the resulting chaos (because the necessary information is not available on the disk). If the filesystem has been damaged beyond repair, the only choice is to use newfs(8) on it and restore it from backup.

The usual solution for this problem was to implement *dirty region logging*, which is also referred to as *journaling*, although that term is not used consistently and is occasionally applied to other forms of transaction logging as well. Meta-data updates are still written synchronously, but only into a small region of the disk. Later on they will be moved to their proper location. Because the logging area is a small, contiguous region on the disk, there are no long distances for the disk heads to move, even during heavy operations, so these operations are quicker than synchronous updates. Additionally the complexity of the implementation is fairly limited, so the risk of bugs being present is low. A disadvantage is that all meta-data are written twice (once into the logging region and once to the proper location) so for normal work, a performance “pessimization” might result. On the other hand, in case of a crash, all pending meta-data operations can be quickly either rolled-back or completed from the logging area after the system comes up again, resulting in a fast filesystem startup.

Kirk McKusick, the developer of Berkeley FFS, solved this problem with Soft Updates: all pending meta-data updates are kept in memory and written out to disk in a sorted sequence (“ordered meta-data updates”). This has the effect that, in case of heavy meta-data operations, later updates to an item “catch” the earlier ones if the earlier ones are still in memory and have not already been written to disk. So all operations on, say, a directory are generally performed in memory before the update is written to disk (the data blocks are sorted according to their position so that they will not be on the disk ahead of their meta-data). If the system crashes, this causes an implicit “log rewind”: all operations which did not find their way to the disk appear as if they had never happened. A consistent filesystem state is maintained that appears to be the one of 30 to 60 seconds earlier. The algorithm used guarantees that all resources in use are marked as such in their appropriate bitmaps: blocks and inodes. After a crash, the only resource allocation error that occurs is that resources are marked as “used” which are actually “free”. fsck(8) recognizes this situation, and frees the resources that are no longer used. It is safe to ignore the dirty state of the filesystem after a crash by forcibly mounting it with `mount -f`. In order to free resources that may be unused, fsck(8) needs to be run at a later time. This is the idea behind the *background fsck*: at system startup time, only a *snapshot* of the filesystem is recorded. The `fsck` can be run later on. All file systems can then be mounted “dirty”, so the system startup proceeds in multiuser mode. Then, background `fsck`s will be scheduled for all file systems where this is required, to free resources that may be unused. (File systems that do not use Soft Updates still need the usual foreground `fsck` though.)

The advantage is that meta-data operations are nearly as fast as asynchronous updates (i.e. faster than with *logging*, which has to write the meta-data twice). The disadvantages are the complexity of the code (implying a higher risk for bugs in an area that is highly sensitive regarding loss of user data), and a higher memory consumption. Additionally there are some idiosyncrasies one has to get used to. After a crash, the state of the filesystem appears to be somewhat "older". In situations where the standard synchronous approach would have caused some zero-length files to remain after the `fsck`, these files do not exist at all with a Soft Updates filesystem because neither the meta-data nor the file contents have ever been written to disk. Disk space is not released until the updates have been written to disk, which may take place some time after running `rm`. This may cause problems when installing large amounts of data on a filesystem that does not have enough free space to hold all the files twice.

11.13 Tuning Kernel Limits

11.13.1 File/Process Limits

11.13.1.1 `kern.maxfiles`

`kern.maxfiles` can be raised or lowered based upon your system requirements. This variable indicates the maximum number of file descriptors on your system. When the file descriptor table is full, `file: table is full` will show up repeatedly in the system message buffer, which can be viewed with the `dmesg` command.

Each open file, socket, or fifo uses one file descriptor. A large-scale production server may easily require many thousands of file descriptors, depending on the kind and number of services running concurrently.

In older FreeBSD releases, the default value of `kern.maxfiles` is derived from the `maxusers` option in your kernel configuration file. `kern.maxfiles` grows proportionally to the value of `maxusers`. When compiling a custom kernel, it is a good idea to set this kernel configuration option according to the uses of your system. From this number, the kernel is given most of its pre-defined limits. Even though a production machine may not actually have 256 users connected at once, the resources needed may be similar to a high-scale web server.

The variable `kern.maxusers` is automatically sized at boot based on the amount of memory available in the system, and may be determined at run-time by inspecting the value of the read-only `kern.maxusers` sysctl. Some sites will require larger or smaller values of `kern.maxusers` and may set it as a loader tunable; values of 64, 128, and 256 are not uncommon. We do not recommend going above 256 unless you need a huge number of file descriptors; many of the tunable values set to their defaults by `kern.maxusers` may be individually overridden at boot-time or run-time in `/boot/loader.conf` (see the loader.conf(5) manual page or the `/boot/defaults/loader.conf` file for some hints) or as described elsewhere in this document.

In older releases, the system will auto-tune `maxusers` for you if you explicitly set it to `0`[1]. When setting this option, you will want to set `maxusers` to at least 4, especially if you are using the X Window System or compiling software. The reason is that the most important table set by `maxusers` is the maximum number of processes, which is set to `20 + 16 * maxusers`, so if you set `maxusers` to 1, then you can only have 36 simultaneous processes, including the 18 or so that the system starts up at boot time and the 15 or so you will probably create when you start the X Window System. Even a simple task like reading a manual page will start up nine processes to filter, decompress, and view it. Setting `maxusers` to 64 will allow you to have up to 1044 simultaneous processes, which should be enough for nearly all uses. If, however, you see the dreaded proc table full error when trying to start another program, or are running a server with a large number of simultaneous users (like `ftp.FreeBSD.org`), you can always increase the number and rebuild.

Note: `maxusers` does *not* limit the number of users which can log into your machine. It simply sets various table sizes to reasonable values considering the maximum number of users you will likely have on your system and how many processes each of them will be running.

11.13.1.2 `kern.ipc.somaxconn`

The `kern.ipc.somaxconn` sysctl variable limits the size of the listen queue for accepting new TCP connections. The default value of `128` is typically too low for robust handling of new connections in a heavily loaded web server environment. For such environments, it is recommended to increase this value to `1024` or higher. The service daemon may itself limit the listen queue size (e.g. sendmail(8), or **Apache**) but will often have a directive in its configuration file to adjust the queue size. Large listen queues also do a better job of avoiding Denial of Service (DoS) attacks.

11.13.2 Network Limits

The `NMBCLUSTERS` kernel configuration option dictates the amount of network Mbufs available to the system. A heavily-trafficked server with a low number of Mbufs will hinder FreeBSD's ability. Each cluster represents approximately 2 K of memory, so a value of 1024 represents 2 megabytes of kernel memory reserved for network buffers. A simple calculation can be done to figure out how many are needed. If you have a web server which maxes out at 1000 simultaneous connections, and each connection eats a 16 K receive and 16 K send buffer, you need approximately 32 MB worth of network buffers to cover the web server. A good rule of thumb is to multiply by 2, so 2x32 MB / 2 KB = 64 MB / 2 kB = 32768. We recommend values between 4096 and 32768 for machines with greater amounts of memory. Under no circumstances should you specify an arbitrarily high value for this parameter as it could lead to a boot time crash. The `-m` option to netstat(1) may be used to observe network cluster use.

`kern.ipc.nmbclusters` loader tunable should be used to tune this at boot time. Only older versions of FreeBSD will require you to use the `NMBCLUSTERS` kernel config(8) option.

For busy servers that make extensive use of the sendfile(2) system call, it may be necessary to increase the number of sendfile(2) buffers via the `NSFBUFS` kernel configuration option or by setting its value in `/boot/loader.conf` (see loader(8) for details). A common indicator that this parameter needs to be adjusted is when processes are seen in the `sfbufa` state. The sysctl variable `kern.ipc.nsfbufs` is a read-only glimpse at the kernel configured variable. This parameter nominally scales with `kern.maxusers`, however it may be necessary to tune accordingly.

Important: Even though a socket has been marked as non-blocking, calling sendfile(2) on the non-blocking socket may result in the sendfile(2) call blocking until enough `struct sf_buf`'s are made available.

11.13.2.1 `net.inet.ip.portrange.*`

The `net.inet.ip.portrange.*` sysctl variables control the port number ranges automatically bound to TCP and UDP sockets. There are three ranges: a low range, a default range, and a high range. Most network programs use the default range which is controlled by the `net.inet.ip.portrange.first` and `net.inet.ip.portrange.last`, which default to 1024 and 5000, respectively. Bound port ranges are used for outgoing connections, and it is possible to run the system out of ports under certain circumstances. This most commonly occurs when you are running a heavily loaded web proxy. The port range is not an issue when running

servers which handle mainly incoming connections, such as a normal web server, or has a limited number of outgoing connections, such as a mail relay. For situations where you may run yourself out of ports, it is recommended to increase `net.inet.ip.portrange.last` modestly. A value of `10000`, `20000` or `30000` may be reasonable. You should also consider firewall effects when changing the port range. Some firewalls may block large ranges of ports (usually low-numbered ports) and expect systems to use higher ranges of ports for outgoing connections — for this reason it is not recommended that `net.inet.ip.portrange.first` be lowered.

11.13.2.2 TCP Bandwidth Delay Product

The TCP Bandwidth Delay Product Limiting is similar to TCP/Vegas in NetBSD. It can be enabled by setting `net.inet.tcp.inflight.enable` sysctl variable to `1`. The system will attempt to calculate the bandwidth delay product for each connection and limit the amount of data queued to the network to just the amount required to maintain optimum throughput.

This feature is useful if you are serving data over modems, Gigabit Ethernet, or even high speed WAN links (or any other link with a high bandwidth delay product), especially if you are also using window scaling or have configured a large send window. If you enable this option, you should also be sure to set `net.inet.tcp.inflight.debug` to `0` (disable debugging), and for production use setting `net.inet.tcp.inflight.min` to at least `6144` may be beneficial. However, note that setting high minimums may effectively disable bandwidth limiting depending on the link. The limiting feature reduces the amount of data built up in intermediate route and switch packet queues as well as reduces the amount of data built up in the local host's interface queue. With fewer packets queued up, interactive connections, especially over slow modems, will also be able to operate with lower *Round Trip Times*. However, note that this feature only effects data transmission (uploading / server side). It has no effect on data reception (downloading).

Adjusting `net.inet.tcp.inflight.stab` is *not* recommended. This parameter defaults to 20, representing 2 maximal packets added to the bandwidth delay product window calculation. The additional window is required to stabilize the algorithm and improve responsiveness to changing conditions, but it can also result in higher ping times over slow links (though still much lower than you would get without the inflight algorithm). In such cases, you may wish to try reducing this parameter to 15, 10, or 5; and may also have to reduce `net.inet.tcp.inflight.min` (for example, to 3500) to get the desired effect. Reducing these parameters should be done as a last resort only.

11.13.3 Virtual Memory

11.13.3.1 `kern.maxvnodes`

A vnode is the internal representation of a file or directory. So increasing the number of vnodes available to the operating system cuts down on disk I/O. Normally this is handled by the operating system and does not need to be changed. In some cases where disk I/O is a bottleneck and the system is running out of vnodes, this setting will need to be increased. The amount of inactive and free RAM will need to be taken into account.

To see the current number of vnodes in use:

```
# sysctl vfs.numvnodes
vfs.numvnodes: 91349
```

To see the maximum vnodes:

```
# sysctl kern.maxvnodes
```

```
kern.maxvnodes: 100000
```

If the current vnode usage is near the maximum, increasing `kern.maxvnodes` by a value of 1,000 is probably a good idea. Keep an eye on the number of `vfs.numvnodes`. If it climbs up to the maximum again, `kern.maxvnodes` will need to be increased further. A shift in your memory usage as reported by top(1) should be visible. More memory should be active.

11.14 Adding Swap Space

No matter how well you plan, sometimes a system does not run as you expect. If you find you need more swap space, it is simple enough to add. You have three ways to increase swap space: adding a new hard drive, enabling swap over NFS, and creating a swap file on an existing partition.

For information on how to encrypt swap space, what options for this task exist and why it should be done, please refer to Section 18.17 of the Handbook.

11.14.1 Swap on a New Hard Drive

The best way to add swap, of course, is to use this as an excuse to add another hard drive. You can always use another hard drive, after all. If you can do this, go reread the discussion of swap space in Section 11.2 of the Handbook for some suggestions on how to best arrange your swap.

11.14.2 Swapping over NFS

Swapping over NFS is only recommended if you do not have a local hard disk to swap to; NFS swapping will be limited by the available network bandwidth and puts an additional burden on the NFS server.

11.14.3 Swapfiles

You can create a file of a specified size to use as a swap file. In our example here we will use a 64MB file called `/usr/swap0`. You can use any name you want, of course.

Example 11-1. Creating a Swapfile on FreeBSD

1. Be certain that your kernel configuration includes the memory disk driver (md(4)). It is default in `GENERIC` kernel.
   ```
   device   md   # Memory "disks"
   ```
2. Create a swapfile (`/usr/swap0`):
   ```
   # dd if=/dev/zero of=/usr/swap0 bs=1024k count=64
   ```
3. Set proper permissions on (`/usr/swap0`):
   ```
   # chmod 0600 /usr/swap0
   ```
4. Enable the swap file in `/etc/rc.conf`:
   ```
   swapfile="/usr/swap0"   # Set to name of swapfile if aux swapfile desired.
   ```

5. Reboot the machine or to enable the swap file immediately, type:
   ```
   # mdconfig -a -t vnode -f /usr/swap0 -u 0 && swapon /dev/md0
   ```

11.15 Power and Resource Management

It is important to utilize hardware resources in an efficient manner. Before ACPI was introduced, it was difficult and inflexible for operating systems to manage the power usage and thermal properties of a system. The hardware was managed by the BIOS and thus the user had less control and visibility into the power management settings. Some limited configurability was available via *Advanced Power Management (APM)*. Power and resource management is one of the key components of a modern operating system. For example, you may want an operating system to monitor system limits (and possibly alert you) in case your system temperature increased unexpectedly.

In this section of the FreeBSD Handbook, we will provide comprehensive information about ACPI. References will be provided for further reading at the end.

11.15.1 What Is ACPI?

Advanced Configuration and Power Interface (ACPI) is a standard written by an alliance of vendors to provide a standard interface for hardware resources and power management (hence the name). It is a key element in *Operating System-directed configuration and Power Management*, i.e.: it provides more control and flexibility to the operating system (OS). Modern systems “stretched” the limits of the current Plug and Play interfaces prior to the introduction of ACPI. ACPI is the direct successor to APM (Advanced Power Management).

11.15.2 Shortcomings of Advanced Power Management (APM)

The *Advanced Power Management (APM)* facility controls the power usage of a system based on its activity. The APM BIOS is supplied by the (system) vendor and it is specific to the hardware platform. An APM driver in the OS mediates access to the *APM Software Interface*, which allows management of power levels. APM should still be used for systems manufactured at or before the year 2000.

There are four major problems in APM. Firstly, power management is done by the (vendor-specific) BIOS, and the OS does not have any knowledge of it. One example of this, is when the user sets idle-time values for a hard drive in the APM BIOS, that when exceeded, it (BIOS) would spin down the hard drive, without the consent of the OS. Secondly, the APM logic is embedded in the BIOS, and it operates outside the scope of the OS. This means users can only fix problems in their APM BIOS by flashing a new one into the ROM; which is a very dangerous procedure with the potential to leave the system in an unrecoverable state if it fails. Thirdly, APM is a vendor-specific technology, which means that there is a lot of parity (duplication of efforts) and bugs found in one vendor’s BIOS, may not be solved in others. Last but not the least, the APM BIOS did not have enough room to implement a sophisticated power policy, or one that can adapt very well to the purpose of the machine.

Plug and Play BIOS (PNPBIOS) was unreliable in many situations. PNPBIOS is 16-bit technology, so the OS has to use 16-bit emulation in order to “interface” with PNPBIOS methods.

The FreeBSD APM driver is documented in the apm(4) manual page.

11.15.3 Configuring ACPI

The `acpi.ko` driver is loaded by default at start up by the loader(8) and should *not* be compiled into the kernel. The reasoning behind this is that modules are easier to work with, say if switching to another `acpi.ko` without doing a kernel rebuild. This has the advantage of making testing easier. Another reason is that starting ACPI after a system has been brought up often doesn't work well. If you are experiencing problems, you can disable ACPI altogether. This driver should not and can not be unloaded because the system bus uses it for various hardware interactions. ACPI can be disabled by setting `hint.acpi.0.disabled="1"` in `/boot/loader.conf` or at the loader(8) prompt.

> **Note:** ACPI and APM cannot coexist and should be used separately. The last one to load will terminate if the driver notices the other running.

ACPI can be used to put the system into a sleep mode with acpiconf(8), the `-s` flag, and a `1-5` option. Most users will only need `1` or `3` (suspend to RAM). Option `5` will do a soft-off which is the same action as:

```
# halt -p
```

Other options are available via sysctl(8). Check out the acpi(4) and acpiconf(8) manual pages for more information.

11.16 Using and Debugging FreeBSD ACPI

ACPI is a fundamentally new way of discovering devices, managing power usage, and providing standardized access to various hardware previously managed by the BIOS. Progress is being made toward ACPI working on all systems, but bugs in some motherboards' *ACPI Machine Language* (AML) bytecode, incompleteness in FreeBSD's kernel subsystems, and bugs in the Intel ACPI-CA interpreter continue to appear.

This document is intended to help you assist the FreeBSD ACPI maintainers in identifying the root cause of problems you observe and debugging and developing a solution. Thanks for reading this and we hope we can solve your system's problems.

11.16.1 Submitting Debugging Information

> **Note:** Before submitting a problem, be sure you are running the latest BIOS version and, if available, embedded controller firmware version.

For those of you that want to submit a problem right away, please send the following information to freebsd-acpi@FreeBSD.org (mailto:freebsd-acpi@FreeBSD.org):

- Description of the buggy behavior, including system type and model and anything that causes the bug to appear. Also, please note as accurately as possible when the bug began occurring if it is new for you.
- The dmesg(8) output after `boot -v`, including any error messages generated by you exercising the bug.
- The dmesg(8) output from `boot -v` with ACPI disabled, if disabling it helps fix the problem.

- Output from `sysctl hw.acpi`. This is also a good way of figuring out what features your system offers.
- URL where your *ACPI Source Language* (ASL) can be found. Do *not* send the ASL directly to the list as it can be very large. Generate a copy of your ASL by running this command:

  ```
  # acpidump -dt > name-system.asl
  ```

  (Substitute your login name for *name* and manufacturer/model for *system*. Example: `njl-FooCo6000.asl`)

Most of the developers watch the FreeBSD-CURRENT mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current) but please submit problems to freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi) to be sure it is seen. Please be patient, all of us have full-time jobs elsewhere. If your bug is not immediately apparent, we will probably ask you to submit a PR via send-pr(1). When entering a PR, please include the same information as requested above. This will help us track the problem and resolve it. Do not send a PR without emailing freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi) first as we use PRs as reminders of existing problems, not a reporting mechanism. It is likely that your problem has been reported by someone before.

11.16.2 Background

ACPI is present in all modern computers that conform to the ia32 (x86), ia64 (Itanium), and amd64 (AMD) architectures. The full standard has many features including CPU performance management, power planes control, thermal zones, various battery systems, embedded controllers, and bus enumeration. Most systems implement less than the full standard. For instance, a desktop system usually only implements the bus enumeration parts while a laptop might have cooling and battery management support as well. Laptops also have suspend and resume, with their own associated complexity.

An ACPI-compliant system has various components. The BIOS and chipset vendors provide various fixed tables (e.g., FADT) in memory that specify things like the APIC map (used for SMP), config registers, and simple configuration values. Additionally, a table of bytecode (the *Differentiated System Description Table* DSDT) is provided that specifies a tree-like name space of devices and methods.

The ACPI driver must parse the fixed tables, implement an interpreter for the bytecode, and modify device drivers and the kernel to accept information from the ACPI subsystem. For FreeBSD, Intel has provided an interpreter (ACPI-CA) that is shared with Linux and NetBSD. The path to the ACPI-CA source code is `src/sys/contrib/dev/acpica`. The glue code that allows ACPI-CA to work on FreeBSD is in `src/sys/dev/acpica/Osd`. Finally, drivers that implement various ACPI devices are found in `src/sys/dev/acpica`.

11.16.3 Common Problems

For ACPI to work correctly, all the parts have to work correctly. Here are some common problems, in order of frequency of appearance, and some possible workarounds or fixes.

11.16.3.1 Mouse Issues

In some cases, resuming from a suspend operation will cause the mouse to fail. A known work around is to add `hint.psm.0.flags="0x3000"` to the `/boot/loader.conf` file. If this does not work then please consider sending a bug report as described above.

11.16.3.2 Suspend/Resume

ACPI has three suspend to RAM (STR) states, S1-S3, and one suspend to disk state (STD), called S4. S5 is "soft off" and is the normal state your system is in when plugged in but not powered up. S4 can actually be implemented two separate ways. S4BIOS is a BIOS-assisted suspend to disk. S4OS is implemented entirely by the operating system.

Start by checking `sysctl hw.acpi` for the suspend-related items. Here are the results for a Thinkpad:

```
hw.acpi.supported_sleep_state: S3 S4 S5
hw.acpi.s4bios: 0
```

This means that we can use `acpiconf -s` to test S3, S4OS, and S5. If `s4bios` was one (1), we would have S4BIOS support instead of S4 OS.

When testing suspend/resume, start with S1, if supported. This state is most likely to work since it does not require much driver support. No one has implemented S2 but if you have it, it is similar to S1. The next thing to try is S3. This is the deepest STR state and requires a lot of driver support to properly reinitialize your hardware. If you have problems resuming, feel free to email the freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi) list but do not expect the problem to be resolved since there are a lot of drivers/hardware that need more testing and work.

A common problem with suspend/resume is that many device drivers do not save, restore, or reinitialize their firmware, registers, or device memory properly. As a first attempt at debugging the problem, try:

```
# sysctl debug.bootverbose=1
# sysctl debug.acpi.suspend_bounce=1
# acpiconf -s 3
```

This test emulates suspend/resume cycle of all device drivers without actually going into S3 state. In some cases, you can easily catch problems with this method (e.g., losing firmware state, device watchdog time out, and retrying forever). Note that the system will not really enter S3 state, which means devices may not lose power, and many will work fine even if suspend/resume methods are totally missing, unlike real S3 state.

Harder cases require additional hardware, i.e., serial port/cable for serial console or Firewire port/cable for dcons(4), and kernel debugging skills.

To help isolate the problem, remove as many drivers from your kernel as possible. If it works, you can narrow down which driver is the problem by loading drivers until it fails again. Typically binary drivers like `nvidia.ko`, X11 display drivers, and USB will have the most problems while Ethernet interfaces usually work fine. If you can properly load/unload the drivers, you can automate this by putting the appropriate commands in `/etc/rc.suspend` and `/etc/rc.resume`. There is a commented-out example for unloading and loading a driver. Try setting `hw.acpi.reset_video` to zero (0) if your display is messed up after resume. Try setting longer or shorter values for `hw.acpi.sleep_delay` to see if that helps.

Another thing to try is load a recent Linux distribution with ACPI support and test their suspend/resume support on the same hardware. If it works on Linux, it is likely a FreeBSD driver problem and narrowing down which driver causes the problems will help us fix the problem. Note that the ACPI maintainers do not usually maintain other drivers (e.g sound, ATA, etc.) so any work done on tracking down a driver problem should probably eventually be posted to the freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current) list and mailed to the driver maintainer. If you are feeling adventurous, go ahead and start putting some debugging printf(3)s in a problematic driver to track down where in its resume function it hangs.

Finally, try disabling ACPI and enabling APM instead. If suspend/resume works with APM, you may be better off sticking with APM, especially on older hardware (pre-2000). It took vendors a while to get ACPI support correct and older hardware is more likely to have BIOS problems with ACPI.

11.16.3.3 System Hangs (temporary or permanent)

Most system hangs are a result of lost interrupts or an interrupt storm. Chipsets have a lot of problems based on how the BIOS configures interrupts before boot, correctness of the APIC (MADT) table, and routing of the *System Control Interrupt* (SCI).

Interrupt storms can be distinguished from lost interrupts by checking the output of `vmstat -i` and looking at the line that has `acpi0`. If the counter is increasing at more than a couple per second, you have an interrupt storm. If the system appears hung, try breaking to DDB (**CTRL**+**ALT**+**ESC** on console) and type `show interrupts`.

Your best hope when dealing with interrupt problems is to try disabling APIC support with `hint.apic.0.disabled="1"` in `loader.conf`.

11.16.3.4 Panics

Panics are relatively rare for ACPI and are the top priority to be fixed. The first step is to isolate the steps to reproduce the panic (if possible) and get a backtrace. Follow the advice for enabling `options DDB` and setting up a serial console (see Section 26.6.5.3) or setting up a dump(8) partition. You can get a backtrace in DDB with `tr`. If you have to handwrite the backtrace, be sure to at least get the lowest five (5) and top five (5) lines in the trace.

Then, try to isolate the problem by booting with ACPI disabled. If that works, you can isolate the ACPI subsystem by using various values of `debug.acpi.disable`. See the acpi(4) manual page for some examples.

11.16.3.5 System Powers Up After Suspend or Shutdown

First, try setting `hw.acpi.disable_on_poweroff="0"` in loader.conf(5). This keeps ACPI from disabling various events during the shutdown process. Some systems need this value set to `1` (the default) for the same reason. This usually fixes the problem of a system powering up spontaneously after a suspend or poweroff.

11.16.3.6 Other Problems

If you have other problems with ACPI (working with a docking station, devices not detected, etc.), please email a description to the mailing list as well; however, some of these issues may be related to unfinished parts of the ACPI subsystem so they might take a while to be implemented. Please be patient and prepared to test patches we may send you.

11.16.4 ASL, `acpidump`, and IASL

The most common problem is the BIOS vendors providing incorrect (or outright buggy!) bytecode. This is usually manifested by kernel console messages like this:

```
ACPI-1287: *** Error: Method execution failed [\\_SB_.PCI0.LPC0.FIGD._STA] \\
(Node 0xc3f6d160), AE_NOT_FOUND
```

Often, you can resolve these problems by updating your BIOS to the latest revision. Most console messages are harmless but if you have other problems like battery status not working, they are a good place to start looking for problems in the AML. The bytecode, known as AML, is compiled from a source language called ASL. The AML is found in the table known as the DSDT. To get a copy of your ASL, use acpidump(8). You should use both the `-t`

(show contents of the fixed tables) and `-d` (disassemble AML to ASL) options. See the Submitting Debugging Information section for an example syntax.

The simplest first check you can do is to recompile your ASL to check for errors. Warnings can usually be ignored but errors are bugs that will usually prevent ACPI from working correctly. To recompile your ASL, issue the following command:

```
# iasl your.asl
```

11.16.5 Fixing Your ASL

In the long run, our goal is for almost everyone to have ACPI work without any user intervention. At this point, however, we are still developing workarounds for common mistakes made by the BIOS vendors. The Microsoft interpreter (`acpi.sys` and `acpiec.sys`) does not strictly check for adherence to the standard, and thus many BIOS vendors who only test ACPI under Windows never fix their ASL. We hope to continue to identify and document exactly what non-standard behavior is allowed by Microsoft's interpreter and replicate it so FreeBSD can work without forcing users to fix the ASL. As a workaround and to help us identify behavior, you can fix the ASL manually. If this works for you, please send a diff(1) of the old and new ASL so we can possibly work around the buggy behavior in ACPI-CA and thus make your fix unnecessary.

Here is a list of common error messages, their cause, and how to fix them:

11.16.5.1 _OS dependencies

Some AML assumes the world consists of various Windows versions. You can tell FreeBSD to claim it is any OS to see if this fixes problems you may have. An easy way to override this is to set `hw.acpi.osname="Windows 2001"` in `/boot/loader.conf` or other similar strings you find in the ASL.

11.16.5.2 Missing Return statements

Some methods do not explicitly return a value as the standard requires. While ACPI-CA does not handle this, FreeBSD has a workaround that allows it to return the value implicitly. You can also add explicit Return statements where required if you know what value should be returned. To force `iasl` to compile the ASL, use the `-f` flag.

11.16.5.3 Overriding the Default AML

After you customize `your.asl`, you will want to compile it, run:

```
# iasl your.asl
```

You can add the `-f` flag to force creation of the AML, even if there are errors during compilation. Remember that some errors (e.g., missing Return statements) are automatically worked around by the interpreter.

`DSDT.aml` is the default output filename for `iasl`. You can load this instead of your BIOS's buggy copy (which is still present in flash memory) by editing `/boot/loader.conf` as follows:

```
acpi_dsdt_load="YES"
acpi_dsdt_name="/boot/DSDT.aml"
```

Be sure to copy your `DSDT.aml` to the `/boot` directory.

11.16.6 Getting Debugging Output From ACPI

The ACPI driver has a very flexible debugging facility. It allows you to specify a set of subsystems as well as the level of verbosity. The subsystems you wish to debug are specified as "layers" and are broken down into ACPI-CA components (ACPI_ALL_COMPONENTS) and ACPI hardware support (ACPI_ALL_DRIVERS). The verbosity of debugging output is specified as the "level" and ranges from ACPI_LV_ERROR (just report errors) to ACPI_LV_VERBOSE (everything). The "level" is a bitmask so multiple options can be set at once, separated by spaces. In practice, you will want to use a serial console to log the output if it is so long it flushes the console message buffer. A full list of the individual layers and levels is found in the acpi(4) manual page.

Debugging output is not enabled by default. To enable it, add `options ACPI_DEBUG` to your kernel configuration file if ACPI is compiled into the kernel. You can add `ACPI_DEBUG=1` to your `/etc/make.conf` to enable it globally. If it is a module, you can recompile just your `acpi.ko` module as follows:

```
# cd /sys/modules/acpi/acpi
&& make clean &&
make ACPI_DEBUG=1
```

Install `acpi.ko` in `/boot/kernel` and add your desired level and layer to `loader.conf`. This example enables debug messages for all ACPI-CA components and all ACPI hardware drivers (CPU, LID, etc.). It will only output error messages, the least verbose level.

```
debug.acpi.layer="ACPI_ALL_COMPONENTS ACPI_ALL_DRIVERS"
debug.acpi.level="ACPI_LV_ERROR"
```

If the information you want is triggered by a specific event (say, a suspend and then resume), you can leave out changes to `loader.conf` and instead use `sysctl` to specify the layer and level after booting and preparing your system for the specific event. The `sysctl`s are named the same as the tunables in `loader.conf`.

11.16.7 References

More information about ACPI may be found in the following locations:

- The FreeBSD ACPI mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi)
- The ACPI Mailing List Archives http://lists.freebsd.org/pipermail/freebsd-acpi/
- The old ACPI Mailing List Archives http://home.jp.FreeBSD.org/mail-list/acpi-jp/
- The ACPI 2.0 Specification http://acpi.info/spec.htm
- FreeBSD Manual pages: acpi(4), acpi_thermal(4), acpidump(8), iasl(8), acpidb(8)
- DSDT debugging resource (http://www.cpqlinux.com/acpi-howto.html#fix_broken_dsdt). (Uses Compaq as an example but generally useful.)

Notes

1. The auto-tuning algorithm sets `maxusers` equal to the amount of memory in the system, with a minimum of 32, and a maximum of 384.

Chapter 12
The FreeBSD Booting Process

12.1 Synopsis

The process of starting a computer and loading the operating system is referred to as “the bootstrap process”, or simply “booting”. FreeBSD’s boot process provides a great deal of flexibility in customizing what happens when you start the system, allowing you to select from different operating systems installed on the same computer, or even different versions of the same operating system or installed kernel.

This chapter details the configuration options you can set and how to customize the FreeBSD boot process. This includes everything that happens until the FreeBSD kernel has started, probed for devices, and started init(8). If you are not quite sure when this happens, it occurs when the text color changes from bright white to grey.

After reading this chapter, you will know:

- What the components of the FreeBSD bootstrap system are, and how they interact.
- The options you can give to the components in the FreeBSD bootstrap to control the boot process.
- The basics of device.hints(5).

> **x86 Only:** This chapter only describes the boot process for FreeBSD running on Intel x86 systems.

12.2 The Booting Problem

Turning on a computer and starting the operating system poses an interesting dilemma. By definition, the computer does not know how to do anything until the operating system is started. This includes running programs from the disk. So if the computer can not run a program from the disk without the operating system, and the operating system programs are on the disk, how is the operating system started?

This problem parallels one in the book *The Adventures of Baron Munchausen*. A character had fallen part way down a manhole, and pulled himself out by grabbing his bootstraps, and lifting. In the early days of computing the term *bootstrap* was applied to the mechanism used to load the operating system, which has become shortened to “booting”.

On x86 hardware the Basic Input/Output System (BIOS) is responsible for loading the operating system. To do this, the BIOS looks on the hard disk for the Master Boot Record (MBR), which must be located on a specific place on the disk. The BIOS has enough knowledge to load and run the MBR, and assumes that the MBR can then carry out the rest of the tasks involved in loading the operating system, possibly with the help of the BIOS.

The code within the MBR is usually referred to as a *boot manager*, especially when it interacts with the user. In this case the boot manager usually has more code in the first *track* of the disk or within some OS’s file system. (A boot

manager is sometimes also called a *boot loader*, but FreeBSD uses that term for a later stage of booting.) Popular boot managers include **boot0** (a.k.a. **Boot Easy**, the standard FreeBSD boot manager), **Grub**, **GAG**, and **LILO**. (Only **boot0** fits within the MBR.)

If you have only one operating system installed on your disks then a standard PC MBR will suffice. This MBR searches for the first bootable (a.k.a. active) slice on the disk, and then runs the code on that slice to load the remainder of the operating system. The MBR installed by fdisk(8), by default, is such an MBR. It is based on `/boot/mbr`.

If you have installed multiple operating systems on your disks then you can install a different boot manager, one that can display a list of different operating systems, and allows you to choose the one to boot from. Two of these are discussed in the next subsection.

The remainder of the FreeBSD bootstrap system is divided into three stages. The first stage is run by the MBR, which knows just enough to get the computer into a specific state and run the second stage. The second stage can do a little bit more, before running the third stage. The third stage finishes the task of loading the operating system. The work is split into these three stages because the PC standards put limits on the size of the programs that can be run at stages one and two. Chaining the tasks together allows FreeBSD to provide a more flexible loader.

The kernel is then started and it begins to probe for devices and initialize them for use. Once the kernel boot process is finished, the kernel passes control to the user process init(8), which then makes sure the disks are in a usable state. init(8) then starts the user-level resource configuration which mounts file systems, sets up network cards to communicate on the network, and generally starts all the processes that usually are run on a FreeBSD system at startup.

12.3 The Boot Manager and Boot Stages

12.3.1 The Boot Manager

The code in the MBR or boot manager is sometimes referred to as *stage zero* of the boot process. This subsection discusses two of the boot managers previously mentioned: **boot0** and **LILO**.

The boot0 Boot Manager: The MBR installed by FreeBSD's installer or boot0cfg(8), by default, is based on `/boot/boot0`. (The **boot0** program is very simple, since the program in the MBR can only be 446 bytes long because of the slice table and `0x55AA` identifier at the end of the MBR.) If you have installed **boot0** and multiple operating systems on your hard disks, then you will see a display similar to this one at boot time:

Example 12-1. `boot0` Screenshot

```
F1 DOS
F2 FreeBSD
F3 Linux
F4 ??
F5 Drive 1

Default: F2
```

Other operating systems, in particular Windows, have been known to overwrite an existing MBR with their own. If this happens to you, or you want to replace your existing MBR with the FreeBSD MBR then use the following command:

```
# fdisk -B -b /boot/boot0 device
```

where *device* is the device that you boot from, such as `ad0` for the first IDE disk, `ad2` for the first IDE disk on a second IDE controller, `da0` for the first SCSI disk, and so on. Or, if you want a custom configuration of the MBR, use boot0cfg(8).

The LILO Boot Manager: To install this boot manager so it will also boot FreeBSD, first start Linux and add the following to your existing `/etc/lilo.conf` configuration file:

```
other=/dev/hdXY
table=/dev/hdX
loader=/boot/chain.b
label=FreeBSD
```

In the above, specify FreeBSD's primary partition and drive using Linux specifiers, replacing *X* with the Linux drive letter and *Y* with the Linux primary partition number. If you are using a SCSI drive, you will need to change */dev/hd* to read something similar to */dev/sd*. The `loader=/boot/chain.b` line can be omitted if you have both operating systems on the same drive. Now run `/sbin/lilo -v` to commit your new changes to the system; this should be verified by checking its screen messages.

12.3.2 Stage One, `/boot/boot1`, and Stage Two, `/boot/boot2`

Conceptually the first and second stages are part of the same program, on the same area of the disk. Because of space constraints they have been split into two, but you would always install them together. They are copied from the combined file `/boot/boot` by the installer or **bsdlabel** (see below).

They are located outside file systems, in the first track of the boot slice, starting with the first sector. This is where boot0, or any other boot manager, expects to find a program to run which will continue the boot process. The number of sectors used is easily determined from the size of `/boot/boot`.

`boot1` is very simple, since it can only be 512 bytes in size, and knows just enough about the FreeBSD *bsdlabel*, which stores information about the slice, to find and execute `boot2`.

`boot2` is slightly more sophisticated, and understands the FreeBSD file system enough to find files on it, and can provide a simple interface to choose the kernel or loader to run.

Since the loader is much more sophisticated, and provides a nice easy-to-use boot configuration, `boot2` usually runs it, but previously it was tasked to run the kernel directly.

Example 12-2. `boot2` Screenshot

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

If you ever need to replace the installed `boot1` and `boot2` use bsdlabel(8):

```
# bsdlabel -B dskslice
```

where *dskslice* is the disk and slice you boot from, such as `ad0s1` for the first slice on the first IDE disk.

Dangerously Dedicated Mode: If you use just the disk name, such as `ad0`, in the bsdlabel(8) command you will create a dangerously dedicated disk, without slices. This is almost certainly not what you want to do, so make sure you double check the bsdlabel(8) command before you press **Return**.

12.3.3 Stage Three, `/boot/loader`

The loader is the final stage of the three-stage bootstrap, and is located on the file system, usually as `/boot/loader`.

The loader is intended as a user-friendly method for configuration, using an easy-to-use built-in command set, backed up by a more powerful interpreter, with a more complex command set.

12.3.3.1 Loader Program Flow

During initialization, the loader will probe for a console and for disks, and figure out what disk it is booting from. It will set variables accordingly, and an interpreter is started where user commands can be passed from a script or interactively.

The loader will then read `/boot/loader.rc`, which by default reads in `/boot/defaults/loader.conf` which sets reasonable defaults for variables and reads `/boot/loader.conf` for local changes to those variables. `loader.rc` then acts on these variables, loading whichever modules and kernel are selected.

Finally, by default, the loader issues a 10 second wait for key presses, and boots the kernel if it is not interrupted. If interrupted, the user is presented with a prompt which understands the easy-to-use command set, where the user may adjust variables, unload all modules, load modules, and then finally boot or reboot.

12.3.3.2 Loader Built-In Commands

These are the most commonly used loader commands. For a complete discussion of all available commands, please see loader(8).

autoboot *`seconds`*

> Proceeds to boot the kernel if not interrupted within the time span given, in seconds. It displays a countdown, and the default time span is 10 seconds.

boot [*`-options`*] [*`kernelname`*]

> Immediately proceeds to boot the kernel, with the given options, if any, and with the kernel name given, if it is. Providing a kernel name on the command-line is only applicable after an *unload* command has been issued, otherwise the previously-loaded kernel will be used.

boot-conf

> Goes through the same automatic configuration of modules based on variables as what happens at boot. This only makes sense if you use `unload` first, and change some variables, most commonly `kernel`.

help [*`topic`*]

> Shows help messages read from `/boot/loader.help`. If the topic given is `index`, then the list of available topics is given.

include *filename* ...

Processes the file with the given filename. The file is read in, and interpreted line by line. An error immediately stops the include command.

load [-t *type*] *filename*

Loads the kernel, kernel module, or file of the type given, with the filename given. Any arguments after filename are passed to the file.

ls [-l] [*path*]

Displays a listing of files in the given path, or the root directory, if the path is not specified. If -l is specified, file sizes will be shown too.

lsdev [-v]

Lists all of the devices from which it may be possible to load modules. If -v is specified, more details are printed.

lsmod [-v]

Displays loaded modules. If -v is specified, more details are shown.

more *filename*

Displays the files specified, with a pause at each LINES displayed.

reboot

Immediately reboots the system.

set *variable*
set *variable*=*value*

Sets the loader's environment variables.

unload

Removes all loaded modules.

12.3.3.3 Loader Examples

Here are some practical examples of loader usage:

- To simply boot your usual kernel, but in single-user mode:

  ```
  boot -s
  ```

- To unload your usual kernel and modules, and then load just your old (or another) kernel:

  ```
  unload
  load kernel.old
  ```

  You can use kernel.GENERIC to refer to the generic kernel that comes on the install disk, or kernel.old to refer to your previously installed kernel (when you have upgraded or configured your own kernel, for example).

Note: Use the following to load your usual modules with another kernel:

```
unload
set kernel="kernel.old"
boot-conf
```

- To load a kernel configuration script (an automated script which does the things you would normally do in the kernel boot-time configurator):

  ```
  load -t userconfig_script /boot/kernel.conf
  ```

12.3.3.4 Boot Time Splash Screens

The splash screen creates a more visually appealing boot screen compared to the original boot messages. This screen will be displayed until a console login prompt or an X display manager offers a login prompt.

There are two basic environments available in FreeBSD. The first is the default legacy virtual console command line environment. After the system finishes booting, a console login prompt is presented. The second environment is the X11 Desktop graphical environment. After X11 and one of the graphical desktop environments, such as **GNOME**, **KDE**, or **XFce** are installed, the X11 desktop can be launched by using the `startx` command.

Some users prefer the X11 graphical login screen over the traditional text based login prompt. Display managers like **XDM** for Xorg, **gdm** for **GNOME**, and **kdm** for **KDE** (and any other from the Ports Collection) basically provide a graphical login screen in place of the console login prompt. After a successful login, they present the user with a graphical desktop.

In the command line environment, the splash screen would hide all the boot probe messages and task startup messages before displaying the login prompt. In X11 environment, the users would get a visually clearer system start up experience resembling something closer to what a (Microsoft Windows or non-unix type system) user would experience.

12.3.3.4.1 Splash Screen Function

The splash screen function only supports 256-color bitmap (`.bmp`) or ZSoft PCX (`.pcx`) files. In addition, the splash image files must have a resolution of 320 by 200 pixels or less to work on standard VGA adapters.

To use larger images, up to the maximum resolution of 1024 by 768 pixels, activate the VESA support included in FreeBSD. This can be enabled by loading the VESA module during system boot, or adding a `VESA` kernel configuration option and building a custom kernel (see Chapter 8). The VESA support gives users the ability to display a splash screen image that fills the whole display screen.

While the splash screen is being displayed during the booting process, it can be turned off any time by hitting any key on the keyboard.

The splash screen also defaults to being a screen saver outside of X11. After a time period of non-use the screen will change to the splash screen and cycle through steps of changing intensity of the image, from bright to a very dark and over again. This default splash screen (screen saver) behavior could be overridden by adding a `saver=` line to `/etc/rc.conf`. Option `saver=` has several built-in screen savers to choose from, the full list can be found in the splash(4) manual page. The default screen saver is called “warp”. Note that the `saver=` option specified in `/etc/rc.conf` only applies to virtual consoles. It has no effect on X11 display managers.

A few boot loader messages, including the boot options menu and a timed wait count down prompt are displayed at boot time, even when the splash screen is enabled.

Sample splash screen files can be downloaded from the gallery at http://artwork.freebsdgr.org (http://artwork.freebsdgr.org/node/3/). By installing the `sysutils/bsd-splash-changer` port, splash images can be chosen from a collection randomly at each boot.

12.3.3.4.2 Enabling the Splash Screen Function

The splash screen (`.bmp`) or (`.pcx`) file has to be placed on the root partition, for example in the `/boot` directory.

For default boot display resolution (256-color, 320 by 200 pixels, or less), edit `/boot/loader.conf`, so it contains the following:

```
splash_bmp_load="YES"
bitmap_load="YES"
bitmap_name="/boot/splash.bmp"
```

For larger video resolutions up to the maximum of 1024 by 768 pixels, edit `/boot/loader.conf`, so it contains the following:

```
vesa_load="YES"
splash_bmp_load="YES"
bitmap_load="YES"
bitmap_name="/boot/splash.bmp"
```

The above assumes that */boot/splash.bmp* is used for splash screen. When a PCX file is desired, use the following statements, plus the `vesa_load="YES"` line depending on the resolution.

```
splash_pcx_load="YES"
bitmap_load="YES"
bitmap_name="/boot/splash.pcx"
```

The file name is not restricted to “splash” as shown in the above example. It can be anything as long as it has type of BMP or PCX, such as *splash_640x400*`.bmp` or *blue_wave*`.pcx`.

Some other interesting `loader.conf` options:

`beastie_disable="YES"`

> This will stop the boot options menu from being displayed, but the timed wait count down prompt will still be present. Even with the display of the boot options menu disabled, entering an option selection at the timed wait count down prompt will enact the corresponding boot option.

`loader_logo="beastie"`

> This will replace the default words “FreeBSD”, which are displayed to the right of the boot options menu with the colored beastie logo like releases in the past had.

For more information, please see the splash(4), loader.conf(5), and vga(4) manual pages.

12.4 Kernel Interaction During Boot

Once the kernel is loaded by either loader (as usual) or boot2 (bypassing the loader), it examines its boot flags, if any, and adjusts its behavior as necessary.

12.4.1 Kernel Boot Flags

Here are the more common boot flags:

`-a`

during kernel initialization, ask for the device to mount as the root file system.

`-C`

boot from CDROM.

`-c`

run UserConfig, the boot-time kernel configurator

`-s`

boot into single-user mode

`-v`

be more verbose during kernel startup

Note: There are other boot flags, read boot(8) for more information on them.

12.5 Device Hints

During initial system startup, the boot loader(8) will read the device.hints(5) file. This file stores kernel boot information known as variables, sometimes referred to as “device hints”. These “device hints” are used by device drivers for device configuration.

Device hints may also be specified at the Stage 3 boot loader prompt. Variables can be added using `set`, removed with `unset`, and viewed with the `show` commands. Variables set in the `/boot/device.hints` file can be overridden here also. Device hints entered at the boot loader are not permanent and will be forgotten on the next reboot.

Once the system is booted, the kenv(1) command can be used to dump all of the variables.

The syntax for the `/boot/device.hints` file is one variable per line, using the standard hash “#” as comment markers. Lines are constructed as follows:

```
hint.driver.unit.keyword="value"
```

The syntax for the Stage 3 boot loader is:

```
set hint.driver.unit.keyword=value
```

`driver` is the device driver name, `unit` is the device driver unit number, and `keyword` is the hint keyword. The keyword may consist of the following options:

- `at`: specifies the bus which the device is attached to.
- `port`: specifies the start address of the I/O to be used.
- `irq`: specifies the interrupt request number to be used.
- `drq`: specifies the DMA channel number.
- `maddr`: specifies the physical memory address occupied by the device.
- `flags`: sets various flag bits for the device.
- `disabled`: if set to `1` the device is disabled.

Device drivers may accept (or require) more hints not listed here, viewing their manual page is recommended. For more information, consult the device.hints(5), kenv(1), loader.conf(5), and loader(8) manual pages.

12.6 Init: Process Control Initialization

Once the kernel has finished booting, it passes control to the user process init(8), which is located at `/sbin/init`, or the program path specified in the `init_path` variable in `loader`.

12.6.1 Automatic Reboot Sequence

The automatic reboot sequence makes sure that the file systems available on the system are consistent. If they are not, and fsck(8) cannot fix the inconsistencies, init(8) drops the system into single-user mode for the system administrator to take care of the problems directly.

12.6.2 Single-User Mode

This mode can be reached through the automatic reboot sequence, or by the user booting with the `-s` option or setting the `boot_single` variable in `loader`.

It can also be reached by calling shutdown(8) without the reboot (`-r`) or halt (`-h`) options, from multi-user mode.

If the system `console` is set to `insecure` in `/etc/ttys`, then the system prompts for the `root` password before initiating single-user mode.

Example 12-3. An Insecure Console in `/etc/ttys`

```
# 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
```

Note: An `insecure` console means that you consider your physical security to the console to be insecure, and want to make sure only someone who knows the `root` password may use single-user mode, and it does not mean that you want to run your console insecurely. Thus, if you want security, choose `insecure`, not `secure`.

12.6.3 Multi-User Mode

If init(8) finds your file systems to be in order, or once the user has finished in single-user mode, the system enters multi-user mode, in which it starts the resource configuration of the system.

12.6.3.1 Resource Configuration (rc)

The resource configuration system reads in configuration defaults from `/etc/defaults/rc.conf`, and system-specific details from `/etc/rc.conf`, and then proceeds to mount the system file systems mentioned in `/etc/fstab`, start up networking services, start up miscellaneous system daemons, and finally runs the startup scripts of locally installed packages.

The rc(8) manual page is a good reference to the resource configuration system, as is examining the scripts themselves.

12.7 Shutdown Sequence

Upon controlled shutdown, via shutdown(8), init(8) will attempt to run the script `/etc/rc.shutdown`, and then proceed to send all processes the `TERM` signal, and subsequently the `KILL` signal to any that do not terminate timely.

To power down a FreeBSD machine on architectures and systems that support power management, simply use the command `shutdown -p now` to turn the power off immediately. To just reboot a FreeBSD system, just use `shutdown -r now`. You need to be `root` or a member of `operator` group to run shutdown(8). The halt(8) and reboot(8) commands can also be used, please refer to their manual pages and to shutdown(8)'s one for more information.

Note: Power management requires acpi(4) support in the kernel or loaded as module for.

Chapter 13
Users and Basic Account Management

13.1 Synopsis

FreeBSD allows multiple users to use the computer at the same time. Obviously, only one of those users can be sitting in front of the screen and keyboard at any one time [1], but any number of users can log in through the network to get their work done. To use the system every user must have an account.

After reading this chapter, you will know:

- The differences between the various user accounts on a FreeBSD system.
- How to add user accounts.
- How to remove user accounts.
- How to change account details, such as the user's full name, or preferred shell.
- How to set limits on a per-account basis, to control the resources such as memory and CPU time that accounts and groups of accounts are allowed to access.
- How to use groups to make account management easier.

Before reading this chapter, you should:

- Understand the basics of UNIX and FreeBSD (Chapter 3).

13.2 Introduction

All access to the system is achieved via accounts, and all processes are run by users, so user and account management are of integral importance on FreeBSD systems.

Every account on a FreeBSD system has certain information associated with it to identify the account.

User name

The user name as it would be typed at the `login:` prompt. User names must be unique across the computer; you may not have two users with the same user name. There are a number of rules for creating valid user names, documented in passwd(5); you would typically use user names that consist of eight or fewer all lower case characters.

Password

Each account has a password associated with it. The password may be blank, in which case no password will be required to access the system. This is normally a very bad idea; every account should have a password.

User ID (UID)

The UID is a number, traditionally from 0 to 65535[2], used to uniquely identify the user to the system. Internally, FreeBSD uses the UID to identify users—any FreeBSD commands that allow you to specify a user name will convert it to the UID before working with it. This means that you can have several accounts with different user names but the same UID. As far as FreeBSD is concerned these accounts are one user. It is unlikely you will ever need to do this.

Group ID (GID)

The GID is a number, traditionally from 0 to 65535[2], used to uniquely identify the primary group that the user belongs to. Groups are a mechanism for controlling access to resources based on a user's GID rather than their UID. This can significantly reduce the size of some configuration files. A user may also be in more than one group.

Login class

Login classes are an extension to the group mechanism that provide additional flexibility when tailoring the system to different users.

Password change time

By default FreeBSD does not force users to change their passwords periodically. You can enforce this on a per-user basis, forcing some or all of your users to change their passwords after a certain amount of time has elapsed.

Account expiry time

By default FreeBSD does not expire accounts. If you are creating accounts that you know have a limited lifespan, for example, in a school where you have accounts for the students, then you can specify when the account expires. After the expiry time has elapsed the account cannot be used to log in to the system, although the account's directories and files will remain.

User's full name

The user name uniquely identifies the account to FreeBSD, but does not necessarily reflect the user's real name. This information can be associated with the account.

Home directory

The home directory is the full path to a directory on the system in which the user will start when logging on to the system. A common convention is to put all user home directories under `/home/`*`username`* or `/usr/home/`*`username`*. The user would store their personal files in their home directory, and any directories they may create in there.

User shell

The shell provides the default environment users use to interact with the system. There are many different kinds of shells, and experienced users will have their own preferences, which can be reflected in their account settings.

There are three main types of accounts: the Superuser, system users, and user accounts. The Superuser account, usually called `root`, is used to manage the system with no limitations on privileges. System users run services. Finally, user accounts are used by real people, who log on, read mail, and so forth.

13.3 The Superuser Account

The superuser account, usually called `root`, comes preconfigured to facilitate system administration, and should not be used for day-to-day tasks like sending and receiving mail, general exploration of the system, or programming.

This is because the superuser, unlike normal user accounts, can operate without limits, and misuse of the superuser account may result in spectacular disasters. User accounts are unable to destroy the system by mistake, so it is generally best to use normal user accounts whenever possible, unless you especially need the extra privilege.

You should always double and triple-check commands you issue as the superuser, since an extra space or missing character can mean irreparable data loss.

So, the first thing you should do after reading this chapter is to create an unprivileged user account for yourself for general usage if you have not already. This applies equally whether you are running a multi-user or single-user machine. Later in this chapter, we discuss how to create additional accounts, and how to change between the normal user and superuser.

13.4 System Accounts

System users are those used to run services such as DNS, mail, web servers, and so forth. The reason for this is security; if all services ran as the superuser, they could act without restriction.

Examples of system users are `daemon`, `operator`, `bind` (for the Domain Name Service), `news`, and `www`.

`nobody` is the generic unprivileged system user. However, it is important to keep in mind that the more services that use `nobody`, the more files and processes that user will become associated with, and hence the more privileged that user becomes.

13.5 User Accounts

User accounts are the primary means of access for real people to the system, and these accounts insulate the user and the environment, preventing the users from damaging the system or other users, and allowing users to customize their environment without affecting others.

Every person accessing your system should have a unique user account. This allows you to find out who is doing what, prevent people from clobbering each others' settings or reading each others' mail, and so forth.

Each user can set up their own environment to accommodate their use of the system, by using alternate shells, editors, key bindings, and language.

13.6 Modifying Accounts

There are a variety of different commands available in the UNIX environment to manipulate user accounts. The most common commands are summarized below, followed by more detailed examples of their usage.

Command	Summary
adduser(8)	The recommended command-line application for adding new users.
rmuser(8)	The recommended command-line application for removing users.
chpass(1)	A flexible tool to change user database information.
passwd(1)	The simple command-line tool to change user passwords.
pw(8)	A powerful and flexible tool to modify all aspects of user accounts.

13.6.1 adduser

adduser(8) is a simple program for adding new users. It creates entries in the system `passwd` and `group` files. It will also create a home directory for the new user, copy in the default configuration files ("dotfiles") from `/usr/share/skel`, and can optionally mail the new user a welcome message.

Example 13-1. Adding a user on FreeBSD

```
# adduser
Username: jru
Full name: J. Random User
Uid (Leave empty for default):
Login group [jru]:
Login group is jru. Invite jru into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh zsh nologin) [sh]: zsh
Home directory [/home/jru]:
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   : jru
Password   : ****
Full Name  : J. Random User
Uid        : 1001
Class      :
Groups     : jru wheel
Home       : /home/jru
Shell      : /usr/local/bin/zsh
Locked     : no
OK? (yes/no): yes
adduser: INFO: Successfully added (jru) to the user database.
Add another user? (yes/no): no
Goodbye!
#
```

> **Note:** The password you type in is not echoed, nor are asterisks displayed. Make sure that you do not mistype the password.

13.6.2 rmuser

You can use rmuser(8) to completely remove a user from the system. rmuser(8) performs the following steps:

1. Removes the user's crontab(1) entry (if any).
2. Removes any at(1) jobs belonging to the user.
3. Kills all processes owned by the user.
4. Removes the user from the system's local password file.
5. Removes the user's home directory (if it is owned by the user).
6. Removes the incoming mail files belonging to the user from `/var/mail`.
7. Removes all files owned by the user from temporary file storage areas such as `/tmp`.
8. Finally, removes the username from all groups to which it belongs in `/etc/group`.

> **Note:** If a group becomes empty and the group name is the same as the username, the group is removed; this complements the per-user unique groups created by adduser(8).

rmuser(8) cannot be used to remove superuser accounts, since that is almost always an indication of massive destruction.

By default, an interactive mode is used, which attempts to make sure you know what you are doing.

Example 13-2. `rmuser` Interactive Account Removal

```
# rmuser jru
Matching password entry:
jru:*:1001:1001::0:0:J. Random User:/home/jru:/usr/local/bin/zsh
Is this the entry you wish to remove? y
Remove user's home directory (/home/jru)? y
Updating password file, updating databases, done.
Updating group file: trusted (removing group jru -- personal group is empty) done.
Removing user's incoming mail file /var/mail/jru: done.
Removing files belonging to jru from /tmp: done.
Removing files belonging to jru from /var/tmp: done.
Removing files belonging to jru from /var/tmp/vi.recover: done.
#
```

13.6.3 chpass

chpass(1) changes user database information such as passwords, shells, and personal information.

Only system administrators, as the superuser, may change other users' information and passwords with chpass(1).

When passed no options, aside from an optional username, chpass(1) displays an editor containing user information. When the user exists from the editor, the user database is updated with the new information.

> **Note:** You will be asked for your password after exiting the editor if you are not the superuser.

Example 13-3. Interactive chpass by Superuser

```
#Changing user database information for jru.
Login: jru
Password: *
Uid [#]: 1001
Gid [# or name]: 1001
Change [month day year]:
Expire [month day year]:
Class:
Home directory: /home/jru
Shell: /usr/local/bin/zsh
Full Name: J. Random User
Office Location:
Office Phone:
Home Phone:
Other information:
```

The normal user can change only a small subset of this information, and only for themselves.

Example 13-4. Interactive chpass by Normal User

```
#Changing user database information for jru.
Shell: /usr/local/bin/zsh
Full Name: J. Random User
Office Location:
Office Phone:
Home Phone:
Other information:
```

> **Note:** chfn(1) and chsh(1) are just links to chpass(1), as are ypchpass(1), ypchfn(1), and ypchsh(1). NIS support is automatic, so specifying the yp before the command is not necessary. If this is confusing to you, do not worry, NIS will be covered in Chapter 29.

13.6.4 passwd

passwd(1) is the usual way to change your own password as a user, or another user's password as the superuser.

> **Note:** To prevent accidental or unauthorized changes, the original password must be entered before a new password can be set.

Example 13-5. Changing Your Password

```
% passwd
Changing local password for jru.
Old password:
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

Example 13-6. Changing Another User's Password as the Superuser

```
# passwd jru
Changing local password for jru.
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

> **Note:** As with chpass(1), yppasswd(1) is just a link to passwd(1), so NIS works with either command.

13.6.5 pw

pw(8) is a command line utility to create, remove, modify, and display users and groups. It functions as a front end to the system user and group files. pw(8) has a very powerful set of command line options that make it suitable for use in shell scripts, but new users may find it more complicated than the other commands presented here.

13.7 Limiting Users

If you have users, the ability to limit their system use may have come to mind. FreeBSD provides several ways an administrator can limit the amount of system resources an individual may use. These limits are divided into two sections: disk quotas, and other resource limits.

Disk quotas limit disk usage to users, and they provide a way to quickly check that usage without calculating it every time. Quotas are discussed in Section 18.15.

The other resource limits include ways to limit the amount of CPU, memory, and other resources a user may consume. These are defined using login classes and are discussed here.

Login classes are defined in `/etc/login.conf`. The precise semantics are beyond the scope of this section, but are described in detail in the login.conf(5) manual page. It is sufficient to say that each user is assigned to a login class (`default` by default), and that each login class has a set of login capabilities associated with it. A login capability is a *`name=value`* pair, where *`name`* is a well-known identifier and *`value`* is an arbitrary string processed accordingly depending on the name. Setting up login classes and capabilities is rather straight-forward and is also described in login.conf(5).

Note: The system does not normally read the configuration in `/etc/login.conf` directly, but reads the database file `/etc/login.conf.db` which provides faster lookups. To generate `/etc/login.conf.db` from `/etc/login.conf`, execute the following command:

```
# cap_mkdb /etc/login.conf
```

Resource limits are different from plain vanilla login capabilities in two ways. First, for every limit, there is a soft (current) and hard limit. A soft limit may be adjusted by the user or application, but may be no higher than the hard limit. The latter may be lowered by the user, but never raised. Second, most resource limits apply per process to a specific user, not the user as a whole. Note, however, that these differences are mandated by the specific handling of the limits, not by the implementation of the login capability framework (i.e., they are not *really* a special case of login capabilities).

And so, without further ado, below are the most commonly used resource limits (the rest, along with all the other login capabilities, may be found in login.conf(5)).

`coredumpsize`

The limit on the size of a core file generated by a program is, for obvious reasons, subordinate to other limits on disk usage (e.g., `filesize`, or disk quotas). Nevertheless, it is often used as a less-severe method of controlling disk space consumption: since users do not generate core files themselves, and often do not delete them, setting this may save them from running out of disk space should a large program (e.g., **emacs**) crash.

`cputime`

This is the maximum amount of CPU time a user's process may consume. Offending processes will be killed by the kernel.

Note: This is a limit on CPU *time* consumed, not percentage of the CPU as displayed in some fields by top(1) and ps(1). A limit on the latter is, at the time of this writing, not possible, and would be rather useless: a compiler—probably a legitimate task—can easily use almost 100% of a CPU for some time.

`filesize`

This is the maximum size of a file the user may possess. Unlike disk quotas, this limit is enforced on individual files, not the set of all files a user owns.

`maxproc`

This is the maximum number of processes a user may be running. This includes foreground and background processes alike. For obvious reasons, this may not be larger than the system limit specified by the `kern.maxproc` sysctl(8). Also note that setting this too small may hinder a user's productivity: it is often

useful to be logged in multiple times or execute pipelines. Some tasks, such as compiling a large program, also spawn multiple processes (e.g., make(1), cc(1), and other intermediate preprocessors).

`memorylocked`

This is the maximum amount a memory a process may have requested to be locked into main memory (e.g., see mlock(2)). Some system-critical programs, such as amd(8), lock into main memory such that in the event of being swapped out, they do not contribute to a system's trashing in time of trouble.

`memoryuse`

This is the maximum amount of memory a process may consume at any given time. It includes both core memory and swap usage. This is not a catch-all limit for restricting memory consumption, but it is a good start.

`openfiles`

This is the maximum amount of files a process may have open. In FreeBSD, files are also used to represent sockets and IPC channels; thus, be careful not to set this too low. The system-wide limit for this is defined by the `kern.maxfiles` sysctl(8).

`sbsize`

This is the limit on the amount of network memory, and thus mbufs, a user may consume. This originated as a response to an old DoS attack by creating a lot of sockets, but can be generally used to limit network communications.

`stacksize`

This is the maximum size a process' stack may grow to. This alone is not sufficient to limit the amount of memory a program may use; consequently, it should be used in conjunction with other limits.

There are a few other things to remember when setting resource limits. Following are some general tips, suggestions, and miscellaneous comments.

- Processes started at system startup by `/etc/rc` are assigned to the `daemon` login class.
- Although the `/etc/login.conf` that comes with the system is a good source of reasonable values for most limits, only you, the administrator, can know what is appropriate for your system. Setting a limit too high may open your system up to abuse, while setting it too low may put a strain on productivity.
- Users of the X Window System (X11) should probably be granted more resources than other users. X11 by itself takes a lot of resources, but it also encourages users to run more programs simultaneously.
- Remember that many limits apply to individual processes, not the user as a whole. For example, setting `openfiles` to 50 means that each process the user runs may open up to 50 files. Thus, the gross amount of files a user may open is the value of `openfiles` multiplied by the value of `maxproc`. This also applies to memory consumption.

For further information on resource limits and login classes and capabilities in general, please consult the relevant manual pages: cap_mkdb(1), getrlimit(2), login.conf(5).

13.8 Groups

A group is simply a list of users. Groups are identified by their group name and GID (Group ID). In FreeBSD (and most other UNIX like systems), the two factors the kernel uses to decide whether a process is allowed to do something is its user ID and list of groups it belongs to. Unlike a user ID, a process has a list of groups associated with it. You may hear some things refer to the "group ID" of a user or process; most of the time, this just means the first group in the list.

The group name to group ID map is in `/etc/group`. This is a plain text file with four colon-delimited fields. The first field is the group name, the second is the encrypted password, the third the group ID, and the fourth the comma-delimited list of members. It can safely be edited by hand (assuming, of course, that you do not make any syntax errors!). For a more complete description of the syntax, see the group(5) manual page.

If you do not want to edit `/etc/group` manually, you can use the pw(8) command to add and edit groups. For example, to add a group called `teamtwo` and then confirm that it exists you can use:

Example 13-7. Adding a Group Using pw(8)

```
# pw groupadd teamtwo
# pw groupshow teamtwo
teamtwo:*:1100:
```

The number `1100` above is the group ID of the group `teamtwo`. Right now, `teamtwo` has no members, and is thus rather useless. Let's change that by inviting `jru` to the `teamtwo` group.

Example 13-8. Setting the List of Members of a Group Using pw(8)

```
# pw groupmod teamtwo -M jru
# pw groupshow teamtwo
teamtwo:*:1100:jru
```

The argument to the `-M` option is a comma-delimited list of users who are to be in the group. From the preceding sections, we know that the password file also contains a group for each user. The latter (the user) is automatically added to the group list by the system; the user will not show up as a member when using the `groupshow` command to pw(8), but will show up when the information is queried via id(1) or similar tool. In other words, pw(8) only manipulates the `/etc/group` file; it will never attempt to read additionally data from `/etc/passwd`.

Example 13-9. Adding a New Member to a Group Using pw(8)

```
# pw groupmod teamtwo -m db
# pw groupshow teamtwo
teamtwo:*:1100:jru,db
```

The argument to the `-m` option is a comma-delimited list of users who are to be added to the group. Unlike the previous example, these users are added to the group and do not replace the list of users in the group.

Example 13-10. Using id(1) to Determine Group Membership

```
% id jru
uid=1001(jru) gid=1001(jru) groups=1001(jru), 1100(teamtwo)
```

As you can see, `jru` is a member of the groups `jru` and `teamtwo`.

For more information about pw(8), see its manual page, and for more information on the format of `/etc/group`, consult the group(5) manual page.

Notes

1. Well, unless you hook up multiple terminals, but we will save that for Chapter 26.
2. It is possible to use UID/GIDs as large as 4294967295, but such IDs can cause serious problems with software that makes assumptions about the values of IDs.

Chapter 14
Security

14.1 Synopsis

This chapter will provide a basic introduction to system security concepts, some general good rules of thumb, and some advanced topics under FreeBSD. A lot of the topics covered here can be applied to system and Internet security in general as well. The Internet is no longer a “friendly” place in which everyone wants to be your kind neighbor. Securing your system is imperative to protect your data, intellectual property, time, and much more from the hands of hackers and the like.

FreeBSD provides an array of utilities and mechanisms to ensure the integrity and security of your system and network.

After reading this chapter, you will know:

- Basic system security concepts, in respect to FreeBSD.
- About the various crypt mechanisms available in FreeBSD, such as DES and MD5.
- How to set up one-time password authentication.
- How to configure TCP Wrappers for use with **inetd**.
- How to set up **Kerberos5** on FreeBSD.
- How to configure IPsec and create a VPN between FreeBSD/Windows machines.
- How to configure and use **OpenSSH**, FreeBSD’s SSH implementation.
- What file system ACLs are and how to use them.
- How to use the **Portaudit** utility to audit third party software packages installed from the Ports Collection.
- How to utilize the FreeBSD security advisories publications.
- Have an idea of what Process Accounting is and how to enable it on FreeBSD.

Before reading this chapter, you should:

- Understand basic FreeBSD and Internet concepts.

Additional security topics are covered throughout this book. For example, Mandatory Access Control is discussed in Chapter 16 and Internet Firewalls are discussed in Chapter 30.

14.2 Introduction

Security is a function that begins and ends with the system administrator. While all BSD UNIX multi-user systems have some inherent security, the job of building and maintaining additional security mechanisms to keep those users

"honest" is probably one of the single largest undertakings of the sysadmin. Machines are only as secure as you make them, and security concerns are ever competing with the human necessity for convenience. UNIX systems, in general, are capable of running a huge number of simultaneous processes and many of these processes operate as servers — meaning that external entities can connect and talk to them. As yesterday's mini-computers and mainframes become today's desktops, and as computers become networked and inter-networked, security becomes an even bigger issue.

System security also pertains to dealing with various forms of attack, including attacks that attempt to crash, or otherwise make a system unusable, but do not attempt to compromise the `root` account ("break root"). Security concerns can be split up into several categories:

1. Denial of service attacks.
2. User account compromises.
3. Root compromise through accessible servers.
4. Root compromise via user accounts.
5. Backdoor creation.

A denial of service attack is an action that deprives the machine of needed resources. Typically, DoS attacks are brute-force mechanisms that attempt to crash or otherwise make a machine unusable by overwhelming its servers or network stack. Some DoS attacks try to take advantage of bugs in the networking stack to crash a machine with a single packet. The latter can only be fixed by applying a bug fix to the kernel. Attacks on servers can often be fixed by properly specifying options to limit the load the servers incur on the system under adverse conditions. Brute-force network attacks are harder to deal with. A spoofed-packet attack, for example, is nearly impossible to stop, short of cutting your system off from the Internet. It may not be able to take your machine down, but it can saturate your Internet connection.

A user account compromise is even more common than a DoS attack. Many sysadmins still run standard **telnetd**, **rlogind**, **rshd**, and **ftpd** servers on their machines. These servers, by default, do not operate over encrypted connections. The result is that if you have any moderate-sized user base, one or more of your users logging into your system from a remote location (which is the most common and convenient way to login to a system) will have his or her password sniffed. The attentive system admin will analyze his remote access logs looking for suspicious source addresses even for successful logins.

One must always assume that once an attacker has access to a user account, the attacker can break `root`. However, the reality is that in a well secured and maintained system, access to a user account does not necessarily give the attacker access to `root`. The distinction is important because without access to `root` the attacker cannot generally hide his tracks and may, at best, be able to do nothing more than mess with the user's files, or crash the machine. User account compromises are very common because users tend not to take the precautions that sysadmins take.

System administrators must keep in mind that there are potentially many ways to break `root` on a machine. The attacker may know the `root` password, the attacker may find a bug in a root-run server and be able to break `root` over a network connection to that server, or the attacker may know of a bug in a suid-root program that allows the attacker to break `root` once he has broken into a user's account. If an attacker has found a way to break `root` on a machine, the attacker may not have a need to install a backdoor. Many of the `root` holes found and closed to date involve a considerable amount of work by the attacker to cleanup after himself, so most attackers install backdoors. A backdoor provides the attacker with a way to easily regain `root` access to the system, but it also gives the smart system administrator a convenient way to detect the intrusion. Making it impossible for an attacker to install a backdoor may actually be detrimental to your security, because it will not close off the hole the attacker found to break in the first place.

Security remedies should always be implemented with a multi-layered “onion peel” approach and can be categorized as follows:

1. Securing `root` and staff accounts.
2. Securing `root`–run servers and suid/sgid binaries.
3. Securing user accounts.
4. Securing the password file.
5. Securing the kernel core, raw devices, and file systems.
6. Quick detection of inappropriate changes made to the system.
7. Paranoia.

The next section of this chapter will cover the above bullet items in greater depth.

14.3 Securing FreeBSD

> **Command vs. Protocol:** Throughout this document, we will use **bold** text to refer to an application, and a `monospaced` font to refer to specific commands. Protocols will use a normal font. This typographical distinction is useful for instances such as ssh, since it is a protocol as well as command.

The sections that follow will cover the methods of securing your FreeBSD system that were mentioned in the last section of this chapter.

14.3.1 Securing the `root` Account and Staff Accounts

First off, do not bother securing staff accounts if you have not secured the `root` account. Most systems have a password assigned to the `root` account. The first thing you do is assume that the password is *always* compromised. This does not mean that you should remove the password. The password is almost always necessary for console access to the machine. What it does mean is that you should not make it possible to use the password outside of the console or possibly even with the su(1) command. For example, make sure that your ptys are specified as being insecure in the `/etc/ttys` file so that direct `root` logins via `telnet` or `rlogin` are disallowed. If using other login services such as **sshd**, make sure that direct `root` logins are disabled there as well. You can do this by editing your `/etc/ssh/sshd_config` file, and making sure that `PermitRootLogin` is set to `no`. Consider every access method — services such as FTP often fall through the cracks. Direct `root` logins should only be allowed via the system console.

Of course, as a sysadmin you have to be able to get to `root`, so we open up a few holes. But we make sure these holes require additional password verification to operate. One way to make `root` accessible is to add appropriate staff accounts to the `wheel` group (in `/etc/group`). The staff members placed in the `wheel` group are allowed to `su` to `root`. You should never give staff members native `wheel` access by putting them in the `wheel` group in their password entry. Staff accounts should be placed in a `staff` group, and then added to the `wheel` group via the `/etc/group` file. Only those staff members who actually need to have `root` access should be placed in the `wheel` group. It is also possible, when using an authentication method such as Kerberos, to use Kerberos’ `.k5login` file in the `root` account to allow a ksu(1) to `root` without having to place anyone at all in the `wheel` group. This may be the better solution since the `wheel` mechanism still allows an intruder to break `root` if the intruder has gotten hold

of your password file and can break into a staff account. While having the `wheel` mechanism is better than having nothing at all, it is not necessarily the safest option.

To lock an account completely, the pw(8) command should be used:

```
#pw lock staff
```

This will prevent the user from logging in using any mechanism, including ssh(1).

Another method of blocking access to accounts would be to replace the encrypted password with a single "`*`" character. This character would never match the encrypted password and thus block user access. For example, the following staff account:

```
foobar:R9DT/Fa1/LV9U:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

Should be changed to this:

```
foobar:*:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

This will prevent the `foobar` user from logging in using conventional methods. This method for access restriction is flawed on sites using **Kerberos** or in situations where the user has set up keys with ssh(1).

These security mechanisms also assume that you are logging in from a more restrictive server to a less restrictive server. For example, if your main box is running all sorts of servers, your workstation should not be running any. In order for your workstation to be reasonably secure you should run as few servers as possible, up to and including no servers at all, and you should run a password-protected screen blanker. Of course, given physical access to a workstation an attacker can break any sort of security you put on it. This is definitely a problem that you should consider, but you should also consider the fact that the vast majority of break-ins occur remotely, over a network, from people who do not have physical access to your workstation or servers.

Using something like Kerberos also gives you the ability to disable or change the password for a staff account in one place, and have it immediately affect all the machines on which the staff member may have an account. If a staff member's account gets compromised, the ability to instantly change his password on all machines should not be underrated. With discrete passwords, changing a password on N machines can be a mess. You can also impose re-passwording restrictions with Kerberos: not only can a Kerberos ticket be made to timeout after a while, but the Kerberos system can require that the user choose a new password after a certain period of time (say, once a month).

14.3.2 Securing Root-run Servers and SUID/SGID Binaries

The prudent sysadmin only runs the servers he needs to, no more, no less. Be aware that third party servers are often the most bug-prone. For example, running an old version of **imapd** or **popper** is like giving a universal `root` ticket out to the entire world. Never run a server that you have not checked out carefully. Many servers do not need to be run as `root`. For example, the **ntalk**, **comsat**, and **finger** daemons can be run in special user *sandboxes*. A sandbox is not perfect, unless you go through a large amount of trouble, but the onion approach to security still stands: If someone is able to break in through a server running in a sandbox, they still have to break out of the sandbox. The more layers the attacker must break through, the lower the likelihood of his success. Root holes have historically been found in virtually every server ever run as `root`, including basic system servers. If you are running a machine through which people only login via **sshd** and never login via **telnetd** or **rshd** or **rlogind**, then turn off those services!

FreeBSD now defaults to running **ntalkd**, **comsat**, and **finger** in a sandbox. Another program which may be a candidate for running in a sandbox is named(8). `/etc/defaults/rc.conf` includes the arguments necessary to run **named** in a sandbox in a commented-out form. Depending on whether you are installing a new system or

upgrading an existing system, the special user accounts used by these sandboxes may not be installed. The prudent sysadmin would research and implement sandboxes for servers whenever possible.

There are a number of other servers that typically do not run in sandboxes: **sendmail**, **popper**, **imapd**, **ftpd**, and others. There are alternatives to some of these, but installing them may require more work than you are willing to perform (the convenience factor strikes again). You may have to run these servers as `root` and rely on other mechanisms to detect break-ins that might occur through them.

The other big potential `root` holes in a system are the suid-root and sgid binaries installed on the system. Most of these binaries, such as **rlogin**, reside in `/bin`, `/sbin`, `/usr/bin`, or `/usr/sbin`. While nothing is 100% safe, the system-default suid and sgid binaries can be considered reasonably safe. Still, `root` holes are occasionally found in these binaries. A `root` hole was found in `Xlib` in 1998 that made **xterm** (which is typically suid) vulnerable. It is better to be safe than sorry and the prudent sysadmin will restrict suid binaries, that only staff should run, to a special group that only staff can access, and get rid of (`chmod 000`) any suid binaries that nobody uses. A server with no display generally does not need an **xterm** binary. Sgid binaries can be almost as dangerous. If an intruder can break an sgid-kmem binary, the intruder might be able to read `/dev/kmem` and thus read the encrypted password file, potentially compromising any passworded account. Alternatively an intruder who breaks group `kmem` can monitor keystrokes sent through ptys, including ptys used by users who login through secure methods. An intruder that breaks the `tty` group can write to almost any user's tty. If a user is running a terminal program or emulator with a keyboard-simulation feature, the intruder can potentially generate a data stream that causes the user's terminal to echo a command, which is then run as that user.

14.3.3 Securing User Accounts

User accounts are usually the most difficult to secure. While you can impose draconian access restrictions on your staff and "star" out their passwords, you may not be able to do so with any general user accounts you might have. If you do have sufficient control, then you may win out and be able to secure the user accounts properly. If not, you simply have to be more vigilant in your monitoring of those accounts. Use of ssh and Kerberos for user accounts is more problematic, due to the extra administration and technical support required, but still a very good solution compared to a encrypted password file.

14.3.4 Securing the Password File

The only sure fire way is to star out as many passwords as you can and use ssh or Kerberos for access to those accounts. Even though the encrypted password file (`/etc/spwd.db`) can only be read by `root`, it may be possible for an intruder to obtain read access to that file even if the attacker cannot obtain root-write access.

Your security scripts should always check for and report changes to the password file (see the Checking file integrity section below).

14.3.5 Securing the Kernel Core, Raw Devices, and File systems

If an attacker breaks `root` he can do just about anything, but there are certain conveniences. For example, most modern kernels have a packet sniffing device driver built in. Under FreeBSD it is called the `bpf` device. An intruder will commonly attempt to run a packet sniffer on a compromised machine. You do not need to give the intruder the capability and most systems do not have the need for the `bpf` device compiled in.

But even if you turn off the bpf device, you still have /dev/mem and /dev/kmem to worry about. For that matter, the intruder can still write to raw disk devices. Also, there is another kernel feature called the module loader, kldload(8). An enterprising intruder can use a KLD module to install his own bpf device, or other sniffing device, on a running kernel. To avoid these problems you have to run the kernel at a higher secure level, at least securelevel 1.

The secure level of the kernel can be set in a variety of ways. The simplest way of raising the secure level of a running kernel is through a sysctl on the kern.securelevel kernel variable:

```
# sysctl kern.securelevel=1
```

By default, the FreeBSD kernel boots with a secure level of -1. The secure level will remain at -1 unless it is altered, either by the administrator or by init(8) because of a setting in the start up scripts. The secure level may be raised during system startup by setting the kern_securelevel_enable variable to YES in the /etc/rc.conf file, and the value of the kern_securelevel variable to the desired secure level.

The default secure level of an FreeBSD system right after the startup scripts are done is -1. This is called “insecure mode” because immutable file flags may be turned off, all devices may be read from or written to, and so on.

Once the secure level is set to 1 or a higher value, the append-only and immutable files are honored, they cannot be turned off, and access to raw devices will be denied. Higher levels restrict even more operations. For a full description of the effect of various secure levels, please read the security(7) manual page (or the manual page of init(8) in releases older than FreeBSD 7.0).

> **Note:** Bumping the secure level to 1 or higher may cause a few problems to X11 (access to /dev/io will be blocked), or to the installation of FreeBSD build from source (the installworld part of the process needs to temporarily reset the append-only and immutable flags of some files), and in a few other cases. Sometimes, as in the case of X11, it may be possible to work around this by starting xdm(1) pretty early in the boot process, when the securelevel is still low enough. Workarounds like this may not be possible for all secure levels or for all the potential restrictions they enforce. A bit of forward planning is a good idea. Understanding the restrictions imposed by each secure level is important as they severly diminish the ease of system use. It will also make choosing a default setting much simpler and prevent any surprises.

If the kernel’s secure level is raised to 1 or a higher value, it may be useful to set the schg flag on critical startup binaries, directories, and script files (i.e. everything that gets run up to the point where the securelevel is set). This might be overdoing it, and upgrading the system is much more difficult when it operates at a high secure level. A less strict compromise is to run the system at a higher secure level but skip setting the schg flag for every system file and directory under the sun. Another possibility is to simply mount / and /usr read-only. It should be noted that being too draconian about what is permitted may prevent the all-important detection of an intrusion.

14.3.6 Checking File Integrity: Binaries, Configuration Files, Etc.

When it comes right down to it, you can only protect your core system configuration and control files so much before the convenience factor rears its ugly head. For example, using chflags to set the schg bit on most of the files in / and /usr is probably counterproductive, because while it may protect the files, it also closes a detection window. The last layer of your security onion is perhaps the most important — detection. The rest of your security is pretty much useless (or, worse, presents you with a false sense of security) if you cannot detect potential intrusions. Half the job of the onion is to slow down the attacker, rather than stop him, in order to be able to catch him in the act.

The best way to detect an intrusion is to look for modified, missing, or unexpected files. The best way to look for modified files is from another (often centralized) limited-access system. Writing your security scripts on the

extra-secure limited-access system makes them mostly invisible to potential attackers, and this is important. In order to take maximum advantage you generally have to give the limited-access box significant access to the other machines in the business, usually either by doing a read-only NFS export of the other machines to the limited-access box, or by setting up ssh key-pairs to allow the limited-access box to ssh to the other machines. Except for its network traffic, NFS is the least visible method — allowing you to monitor the file systems on each client box virtually undetected. If your limited-access server is connected to the client boxes through a switch, the NFS method is often the better choice. If your limited-access server is connected to the client boxes through a hub, or through several layers of routing, the NFS method may be too insecure (network-wise) and using ssh may be the better choice even with the audit-trail tracks that ssh lays.

Once you have given a limited-access box at least read access to the client systems it is supposed to monitor, you must write scripts to do the actual monitoring. Given an NFS mount, you can write scripts out of simple system utilities such as find(1) and md5(1). It is best to physically md5 the client-box files at least once a day, and to test control files such as those found in `/etc` and `/usr/local/etc` even more often. When mismatches are found, relative to the base md5 information the limited-access machine knows is valid, it should scream at a sysadmin to go check it out. A good security script will also check for inappropriate suid binaries and for new or deleted files on system partitions such as `/` and `/usr`.

When using ssh rather than NFS, writing the security script is much more difficult. You essentially have to `scp` the scripts to the client box in order to run them, making them visible, and for safety you also need to `scp` the binaries (such as find) that those scripts use. The **ssh** client on the client box may already be compromised. All in all, using ssh may be necessary when running over insecure links, but it is also a lot harder to deal with.

A good security script will also check for changes to user and staff members access configuration files: `.rhosts`, `.shosts`, `.ssh/authorized_keys` and so forth, files that might fall outside the purview of the `MD5` check.

If you have a huge amount of user disk space, it may take too long to run through every file on those partitions. In this case, setting mount flags to disallow suid binaries is a good idea. The `nosuid` option (see mount(8)) is what you want to look into. You should probably scan them anyway, at least once a week, since the object of this layer is to detect a break-in attempt, whether or not the attempt succeeds.

Process accounting (see accton(8)) is a relatively low-overhead feature of the operating system which might help as a post-break-in evaluation mechanism. It is especially useful in tracking down how an intruder has actually broken into a system, assuming the file is still intact after the break-in has occured.

Finally, security scripts should process the log files, and the logs themselves should be generated in as secure a manner as possible — remote syslog can be very useful. An intruder will try to cover his tracks, and log files are critical to the sysadmin trying to track down the time and method of the initial break-in. One way to keep a permanent record of the log files is to run the system console to a serial port and collect the information to a secure machine monitoring the consoles.

14.3.7 Paranoia

A little paranoia never hurts. As a rule, a sysadmin can add any number of security features, as long as they do not affect convenience, and can add security features that *do* affect convenience with some added thought. Even more importantly, a security administrator should mix it up a bit — if you use recommendations such as those given by this document verbatim, you give away your methodologies to the prospective attacker who also has access to this document.

14.3.8 Denial of Service Attacks

This section covers Denial of Service attacks. A DoS attack is typically a packet attack. While there is not much you can do about modern spoofed packet attacks that saturate your network, you can generally limit the damage by ensuring that the attacks cannot take down your servers by:

1. Limiting server forks.
2. Limiting springboard attacks (ICMP response attacks, ping broadcast, etc.).
3. Overloading the Kernel Route Cache.

A common DoS attack scenario is attacking a forking server and making it spawning so many child processes that the host system eventually runs out of memory, file descriptors, etc. and then grinds to a halt. **inetd** (see inetd(8)) has several options to limit this sort of attack. It should be noted that while it is possible to prevent a machine from going down, it is not generally possible to prevent a service from being disrupted by the attack. Read the **inetd** manual page carefully and pay specific attention to the `-c`, `-C`, and `-R` options. Note that spoofed-IP attacks will circumvent the `-C` option to **inetd**, so typically a combination of options must be used. Some standalone servers have self-fork-limitation parameters.

Sendmail has its `-OMaxDaemonChildren` option, which tends to work much better than trying to use **Sendmail**'s load limiting options due to the load lag. You should specify a `MaxDaemonChildren` parameter, when you start **sendmail**; high enough to handle your expected load, but not so high that the computer cannot handle that number of **Sendmail** instances without falling on its face. It is also prudent to run **Sendmail** in queued mode (`-ODeliveryMode=queued`) and to run the daemon (`sendmail -bd`) separate from the queue-runs (`sendmail -q15m`). If you still want real-time delivery you can run the queue at a much lower interval, such as `-q1m`, but be sure to specify a reasonable `MaxDaemonChildren` option for *that* **Sendmail** to prevent cascade failures.

Syslogd can be attacked directly and it is strongly recommended that you use the `-s` option whenever possible, and the `-a` option otherwise.

You should also be fairly careful with connect-back services such as **TCP Wrapper**'s reverse-identd, which can be attacked directly. You generally do not want to use the reverse-ident feature of **TCP Wrapper** for this reason.

It is a very good idea to protect internal services from external access by firewalling them off at your border routers. The idea here is to prevent saturation attacks from outside your LAN, not so much to protect internal services from network-based `root` compromise. Always configure an exclusive firewall, i.e., "firewall everything *except* ports A, B, C, D, and M-Z". This way you can firewall off all of your low ports except for certain specific services such as **named** (if you are primary for a zone), **ntalkd**, **sendmail**, and other Internet-accessible services. If you try to configure the firewall the other way — as an inclusive or permissive firewall, there is a good chance that you will forget to "close" a couple of services, or that you will add a new internal service and forget to update the firewall. You can still open up the high-numbered port range on the firewall, to allow permissive-like operation, without compromising your low ports. Also take note that FreeBSD allows you to control the range of port numbers used for dynamic binding, via the various `net.inet.ip.portrange` `sysctl`'s (`sysctl -a | fgrep portrange`), which can also ease the complexity of your firewall's configuration. For example, you might use a normal first/last range of 4000 to 5000, and a hiport range of 49152 to 65535, then block off everything under 4000 in your firewall (except for certain specific Internet-accessible ports, of course).

Another common DoS attack is called a springboard attack — to attack a server in a manner that causes the server to generate responses which overloads the server, the local network, or some other machine. The most common attack of this nature is the *ICMP ping broadcast attack*. The attacker spoofs ping packets sent to your LAN's broadcast address with the source IP address set to the actual machine they wish to attack. If your border routers are not configured to stomp on ping packets to broadcast addresses, your LAN winds up generating sufficient responses to

the spoofed source address to saturate the victim, especially when the attacker uses the same trick on several dozen broadcast addresses over several dozen different networks at once. Broadcast attacks of over a hundred and twenty megabits have been measured. A second common springboard attack is against the ICMP error reporting system. By constructing packets that generate ICMP error responses, an attacker can saturate a server's incoming network and cause the server to saturate its outgoing network with ICMP responses. This type of attack can also crash the server by running it out of memory, especially if the server cannot drain the ICMP responses it generates fast enough. Use the **sysctl** variable `net.inet.icmp.icmplim` to limit these attacks. The last major class of springboard attacks is related to certain internal **inetd** services such as the udp echo service. An attacker simply spoofs a UDP packet with the source address being server A's echo port, and the destination address being server B's echo port, where server A and B are both on your LAN. The two servers then bounce this one packet back and forth between each other. The attacker can overload both servers and their LANs simply by injecting a few packets in this manner. Similar problems exist with the internal **chargen** port. A competent sysadmin will turn off all of these inetd-internal test services.

Spoofed packet attacks may also be used to overload the kernel route cache. Refer to the `net.inet.ip.rtexpire`, `rtminexpire`, and `rtmaxcache` `sysctl` parameters. A spoofed packet attack that uses a random source IP will cause the kernel to generate a temporary cached route in the route table, viewable with `netstat -rna | fgrep W3`. These routes typically timeout in 1600 seconds or so. If the kernel detects that the cached route table has gotten too big it will dynamically reduce the `rtexpire` but will never decrease it to less than `rtminexpire`. There are two problems:

1. The kernel does not react quickly enough when a lightly loaded server is suddenly attacked.
2. The `rtminexpire` is not low enough for the kernel to survive a sustained attack.

If your servers are connected to the Internet via a T3 or better, it may be prudent to manually override both `rtexpire` and `rtminexpire` via sysctl(8). Never set either parameter to zero (unless you want to crash the machine). Setting both parameters to 2 seconds should be sufficient to protect the route table from attack.

14.3.9 Access Issues with Kerberos and SSH

There are a few issues with both Kerberos and ssh that need to be addressed if you intend to use them. Kerberos 5 is an excellent authentication protocol, but there are bugs in the kerberized **telnet** and **rlogin** applications that make them unsuitable for dealing with binary streams. Also, by default Kerberos does not encrypt a session unless you use the `-x` option. **ssh** encrypts everything by default.

Ssh works quite well in every respect except that it forwards encryption keys by default. What this means is that if you have a secure workstation holding keys that give you access to the rest of the system, and you ssh to an insecure machine, your keys are usable. The actual keys themselves are not exposed, but ssh installs a forwarding port for the duration of your login, and if an attacker has broken `root` on the insecure machine he can utilize that port to use your keys to gain access to any other machine that your keys unlock.

We recommend that you use ssh in combination with Kerberos whenever possible for staff logins. **Ssh** can be compiled with Kerberos support. This reduces your reliance on potentially exposed ssh keys while at the same time protecting passwords via Kerberos. Ssh keys should only be used for automated tasks from secure machines (something that Kerberos is unsuited to do). We also recommend that you either turn off key-forwarding in the ssh configuration, or that you make use of the `from=IP/DOMAIN` option that ssh allows in its `authorized_keys` file to make the key only usable to entities logging in from specific machines.

14.4 DES, Blowfish, MD5, and Crypt

Every user on a UNIX system has a password associated with their account. It seems obvious that these passwords need to be known only to the user and the actual operating system. In order to keep these passwords secret, they are encrypted with what is known as a "one-way hash", that is, they can only be easily encrypted but not decrypted. In other words, what we told you a moment ago was obvious is not even true: the operating system itself does not *really* know the password. It only knows the *encrypted* form of the password. The only way to get the "plain-text" password is by a brute force search of the space of possible passwords.

Unfortunately the only secure way to encrypt passwords when UNIX came into being was based on DES, the Data Encryption Standard. This was not such a problem for users resident in the US, but since the source code for DES could not be exported outside the US, FreeBSD had to find a way to both comply with US law and retain compatibility with all the other UNIX variants that still used DES.

The solution was to divide up the encryption libraries so that US users could install the DES libraries and use DES but international users still had an encryption method that could be exported abroad. This is how FreeBSD came to use MD5 as its default encryption method. MD5 is believed to be more secure than DES, so installing DES is offered primarily for compatibility reasons.

14.4.1 Recognizing Your Crypt Mechanism

Currently the library supports DES, MD5 and Blowfish hash functions. By default FreeBSD uses MD5 to encrypt passwords.

It is pretty easy to identify which encryption method FreeBSD is set up to use. Examining the encrypted passwords in the `/etc/master.passwd` file is one way. Passwords encrypted with the MD5 hash are longer than those encrypted with the DES hash and also begin with the characters `$1$`. Passwords starting with `$2a$` are encrypted with the Blowfish hash function. DES password strings do not have any particular identifying characteristics, but they are shorter than MD5 passwords, and are coded in a 64-character alphabet which does not include the `$` character, so a relatively short string which does not begin with a dollar sign is very likely a DES password.

The password format used for new passwords is controlled by the `passwd_format` login capability in `/etc/login.conf`, which takes values of `des`, `md5` or `blf`. See the login.conf(5) manual page for more information about login capabilities.

14.5 One-time Passwords

By default, FreeBSD includes support for OPIE (One-time Passwords In Everything), which uses the MD5 hash by default.

There are three different sorts of passwords which we will discuss below. The first is your usual UNIX style or Kerberos password; we will call this a "UNIX password". The second sort is the one-time password which is generated by the OPIE opiekey(1) program and accepted by the opiepasswd(1) program and the login prompt; we will call this a "one-time password". The final sort of password is the secret password which you give to the `opiekey` program (and sometimes the `opiepasswd` programs) which it uses to generate one-time passwords; we will call it a "secret password" or just unqualified "password".

The secret password does not have anything to do with your UNIX password; they can be the same but this is not recommended. OPIE secret passwords are not limited to 8 characters like old UNIX passwords[1], they can be as long

as you like. Passwords of six or seven word long phrases are fairly common. For the most part, the OPIE system operates completely independently of the UNIX password system.

Besides the password, there are two other pieces of data that are important to OPIE. One is what is known as the "seed" or "key", consisting of two letters and five digits. The other is what is called the "iteration count", a number between 1 and 100. OPIE creates the one-time password by concatenating the seed and the secret password, then applying the MD5 hash as many times as specified by the iteration count and turning the result into six short English words. These six English words are your one-time password. The authentication system (primarily PAM) keeps track of the last one-time password used, and the user is authenticated if the hash of the user-provided password is equal to the previous password. Because a one-way hash is used it is impossible to generate future one-time passwords if a successfully used password is captured; the iteration count is decremented after each successful login to keep the user and the login program in sync. When the iteration count gets down to 1, OPIE must be reinitialized.

There are a few programs involved in each system which we will discuss below. The `opiekey` program accepts an iteration count, a seed, and a secret password, and generates a one-time password or a consecutive list of one-time passwords. The `opiepasswd` program is used to initialize OPIE, and to change passwords, iteration counts, or seeds; it takes either a secret passphrase, or an iteration count, seed, and a one-time password. The `opieinfo` program will examine the relevant credentials files (`/etc/opiekeys`) and print out the invoking user's current iteration count and seed.

There are four different sorts of operations we will cover. The first is using `opiepasswd` over a secure connection to set up one-time-passwords for the first time, or to change your password or seed. The second operation is using `opiepasswd` over an insecure connection, in conjunction with `opiekey` over a secure connection, to do the same. The third is using `opiekey` to log in over an insecure connection. The fourth is using `opiekey` to generate a number of keys which can be written down or printed out to carry with you when going to some location without secure connections to anywhere.

14.5.1 Secure Connection Initialization

To initialize OPIE for the first time, execute the `opiepasswd` command:

```
% 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
```

At the `Enter new secret pass phrase:` or `Enter secret password:` prompts, you should enter a password or phrase. Remember, this is not the password that you will use to login with, this is used to generate your one-time login keys. The "ID" line gives the parameters of your particular instance: your login name, the iteration count, and seed. When logging in the system will remember these parameters and present them back to you so you do not have to remember them. The last line gives the particular one-time password which corresponds to those parameters and your secret password; if you were to re-login immediately, this one-time password is the one you would use.

14.5.2 Insecure Connection Initialization

To initialize or change your secret password over an insecure connection, you will need to already have a secure connection to some place where you can run `opiekey`; this might be in the form of a shell prompt on a machine you trust. You will also need to make up an iteration count (100 is probably a good value), and you may make up your own seed or use a randomly-generated one. Over on the insecure connection (to the machine you are initializing), use `opiepasswd`:

```
% 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
```

To accept the default seed press **Return**. Then before entering an access password, move over to your secure connection and give it the same parameters:

```
% 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
```

Now switch back over to the insecure connection, and copy the one-time password generated over to the relevant program.

14.5.3 Generating a Single One-time Password

Once you have initialized OPIE and login, you will be presented with a prompt like this:

```
% telnet example.com
Trying 10.0.0.1...
Connected to example.com
Escape character is '^]'.

FreeBSD/i386 (example.com) (ttypa)

login: <username>
otp-md5 498 gr4269 ext
Password:
```

As a side note, the OPIE prompts have a useful feature (not shown here): if you press **Return** at the password prompt, the prompter will turn echo on, so you can see what you are typing. This can be extremely useful if you are attempting to type in a password by hand, such as from a printout.

At this point you need to generate your one-time password to answer this login prompt. This must be done on a trusted system that you can run `opiekey` on. (There are versions of these for DOS, Windows and Mac OS as well.) They need the iteration count and the seed as command line options. You can cut-and-paste these right from the login prompt on the machine that you are logging in to.

On the trusted system:

```
% 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
```

Now that you have your one-time password you can continue logging in.

14.5.4 Generating Multiple One-time Passwords

Sometimes you have to go places where you do not have access to a trusted machine or secure connection. In this case, it is possible to use the `opiekey` command to generate a number of one-time passwords beforehand to be printed out and taken with you. For example:

```
% 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
```

The `-n 5` requests five keys in sequence, the `30` specifies what the last iteration number should be. Note that these are printed out in *reverse* order of eventual use. If you are really paranoid, you might want to write the results down by hand; otherwise you can cut-and-paste into `lpr`. Note that each line shows both the iteration count and the one-time password; you may still find it handy to scratch off passwords as you use them.

14.5.5 Restricting Use of UNIX® Passwords

OPIE can restrict the use of UNIX passwords based on the IP address of a login session. The relevant file is `/etc/opieaccess`, which is present by default. Please check opieaccess(5) for more information on this file and which security considerations you should be aware of when using it.

Here is a sample `opieaccess` file:

```
permit 192.168.0.0 255.255.0.0
```

This line allows users whose IP source address (which is vulnerable to spoofing) matches the specified value and mask, to use UNIX passwords at any time.

If no rules in `opieaccess` are matched, the default is to deny non-OPIE logins.

14.6 TCP Wrappers

Anyone familiar with inetd(8) has probably heard of TCP Wrappers at some point. But few individuals seem to fully comprehend its usefulness in a network environment. It seems that everyone wants to install a firewall to handle network connections. While a firewall has a wide variety of uses, there are some things that a firewall will not handle, such as sending text back to the connection originator. The TCP Wrappers software does this and much more. In the next few sections many of the TCP Wrappers features will be discussed, and, when applicable, example configuration lines will be provided.

The TCP Wrappers software extends the abilities of **inetd** to provide support for every server daemon under its control. Using this method it is possible to provide logging support, return messages to connections, permit a daemon to only accept internal connections, etc. While some of these features can be provided by implementing a firewall, this will add not only an extra layer of protection but go beyond the amount of control a firewall can provide.

The added functionality of TCP Wrappers should not be considered a replacement for a good firewall. TCP Wrappers can be used in conjunction with a firewall or other security enhancements though and it can serve nicely as an extra layer of protection for the system.

Since this is an extension to the configuration of **inetd**, the reader is expected have read the inetd configuration section.

> **Note:** While programs run by inetd(8) are not exactly “daemons”, they have traditionally been called daemons. This is the term we will use in this section too.

14.6.1 Initial Configuration

The only requirement of using TCP Wrappers in FreeBSD is to ensure the **inetd** server is started from `rc.conf` with the `-Ww` option; this is the default setting. Of course, proper configuration of `/etc/hosts.allow` is also expected, but syslogd(8) will throw messages in the system logs in these cases.

> **Note:** Unlike other implementations of TCP Wrappers, the use of `hosts.deny` has been deprecated. All configuration options should be placed in `/etc/hosts.allow`.

In the simplest configuration, daemon connection policies are set to either be permitted or blocked depending on the options in `/etc/hosts.allow`. The default configuration in FreeBSD is to allow a connection to every daemon started with **inetd**. Changing this will be discussed only after the basic configuration is covered.

Basic configuration usually takes the form of `daemon : address : action`. Where `daemon` is the daemon name which `inetd` started. The `address` can be a valid hostname, an IP address or an IPv6 address enclosed in brackets ([ ]). The `action` field can be either `allow` or `deny` to grant or deny access appropriately. Keep in mind that configuration works off a first rule match semantic, meaning that the configuration file is scanned in ascending order for a matching rule. When a match is found the rule is applied and the search process will halt.

Several other options exist but they will be explained in a later section. A simple configuration line may easily be constructed from that information alone. For example, to allow POP3 connections via the `mail/qpopper` daemon, the following lines should be appended to `hosts.allow`:

```
# This line is required for POP3 connections:
qpopper : ALL : allow
```

After adding this line, **inetd** will need to be restarted. This can be accomplished by use of the kill(1) command, or with the *restart* parameter with /etc/rc.d/inetd.

14.6.2 Advanced Configuration

TCP Wrappers has advanced options too; they will allow for more control over the way connections are handled. In some cases it may be a good idea to return a comment to certain hosts or daemon connections. In other cases, perhaps a log file should be recorded or an email sent to the administrator. Other situations may require the use of a service for local connections only. This is all possible through the use of configuration options known as wildcards, expansion characters and external command execution. The next two sections are written to cover these situations.

14.6.2.1 External Commands

Suppose that a situation occurs where a connection should be denied yet a reason should be sent to the individual who attempted to establish that connection. How could it be done? That action can be made possible by using the twist option. When a connection attempt is made, twist will be called to execute a shell command or script. An example already exists in the hosts.allow file:

```
# The rest of the daemons are protected.
ALL : ALL \
        : severity auth.info \
        : twist /bin/echo "You are not welcome to use %d from %h."
```

This example shows that the message, “You are not allowed to use daemon from hostname.” will be returned for any daemon not previously configured in the access file. This is extremely useful for sending a reply back to the connection initiator right after the established connection is dropped. Note that any message returned *must* be wrapped in quote " characters; there are no exceptions to this rule.

> **Warning:** It may be possible to launch a denial of service attack on the server if an attacker, or group of attackers could flood these daemons with connection requests.

Another possibility is to use the spawn option in these cases. Like twist, the spawn option implicitly denies the connection and may be used to run external shell commands or scripts. Unlike twist, spawn will not send a reply back to the individual who established the connection. For an example, consider the following configuration line:

```
# 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
```

This will deny all connection attempts from the *.example.com domain; simultaneously logging the hostname, IP address and the daemon which they attempted to access in the /var/log/connections.log file.

Aside from the already explained substitution characters above, e.g. %a, a few others exist. See the hosts_access(5) manual page for the complete list.

14.6.2.2 Wildcard Options

Thus far the `ALL` option has been used continuously throughout the examples. Other options exist which could extend the functionality a bit further. For instance, `ALL` may be used to match every instance of either a daemon, domain or an IP address. Another wildcard available is `PARANOID` which may be used to match any host which provides an IP address that may be forged. In other words, `PARANOID` may be used to define an action to be taken whenever a connection is made from an IP address that differs from its hostname. The following example may shed some more light on this discussion:

```
# Block possibly spoofed requests to sendmail:
sendmail : PARANOID : deny
```

In that example all connection requests to `sendmail` which have an IP address that varies from its hostname will be denied.

> **Caution:** Using the `PARANOID` wildcard may severely cripple servers if the client or server has a broken DNS setup. Administrator discretion is advised.

To learn more about wildcards and their associated functionality, see the hosts_access(5) manual page.

Before any of the specific configuration lines above will work, the first configuration line should be commented out in `hosts.allow`. This was noted at the beginning of this section.

14.7 Kerberos5

Kerberos is a network add-on system/protocol that allows users to authenticate themselves through the services of a secure server. Services such as remote login, remote copy, secure inter-system file copying and other high-risk tasks are made considerably safer and more controllable.

Kerberos can be described as an identity-verifying proxy system. It can also be described as a trusted third-party authentication system. **Kerberos** provides only one function — the secure authentication of users on the network. It does not provide authorization functions (what users are allowed to do) or auditing functions (what those users did). After a client and server have used **Kerberos** to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

Therefore it is highly recommended that **Kerberos** be used with other security methods which provide authorization and audit services.

The following instructions can be used as a guide on how to set up **Kerberos** as distributed for FreeBSD. However, you should refer to the relevant manual pages for a complete description.

For purposes of demonstrating a **Kerberos** installation, the various name spaces will be handled as follows:

- The DNS domain (“zone”) will be example.org.
- The **Kerberos** realm will be EXAMPLE.ORG.

> **Note:** Please use real domain names when setting up **Kerberos** even if you intend to run it internally. This avoids DNS problems and assures inter-operation with other **Kerberos** realms.

14.7.1 History

Kerberos was created by MIT as a solution to network security problems. The **Kerberos** protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection.

Kerberos is both the name of a network authentication protocol and an adjective to describe programs that implement the program (**Kerberos** telnet, for example). The current version of the protocol is version 5, described in RFC 1510.

Several free implementations of this protocol are available, covering a wide range of operating systems. The Massachusetts Institute of Technology (MIT), where **Kerberos** was originally developed, continues to develop their **Kerberos** package. It is commonly used in the US as a cryptography product, as such it has historically been affected by US export regulations. The MIT **Kerberos** is available as a port (`security/krb5`). Heimdal **Kerberos** is another version 5 implementation, and was explicitly developed outside of the US to avoid export regulations (and is thus often included in non-commercial UNIX variants). The Heimdal **Kerberos** distribution is available as a port (`security/heimdal`), and a minimal installation of it is included in the base FreeBSD install.

In order to reach the widest audience, these instructions assume the use of the Heimdal distribution included in FreeBSD.

14.7.2 Setting up a Heimdal KDC

The Key Distribution Center (KDC) is the centralized authentication service that **Kerberos** provides — it is the computer that issues **Kerberos** tickets. The KDC is considered “trusted” by all other computers in the **Kerberos** realm, and thus has heightened security concerns.

Note that while running the **Kerberos** server requires very few computing resources, a dedicated machine acting only as a KDC is recommended for security reasons.

To begin setting up a KDC, ensure that your `/etc/rc.conf` file contains the correct settings to act as a KDC (you may need to adjust paths to reflect your own system):

```
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

Next we will set up your **Kerberos** config file, `/etc/krb5.conf`:

```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
        kdc = kerberos.example.org
        admin_server = kerberos.example.org
    }
[domain_realm]
    .example.org = EXAMPLE.ORG
```

Note that this `/etc/krb5.conf` file implies that your KDC will have the fully-qualified hostname of `kerberos.example.org`. You will need to add a CNAME (alias) entry to your zone file to accomplish this if your KDC has a different hostname.

Note: For large networks with a properly configured BIND DNS server, the above example could be trimmed to:

```
[libdefaults]
      default_realm = EXAMPLE.ORG
```

With the following lines being appended to the `example.org` zonefile:

```
_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
```

Note: For clients to be able to find the **Kerberos** services, you *must* have either a fully configured `/etc/krb5.conf` or a minimally configured `/etc/krb5.conf` *and* a properly configured DNS server.

Next we will create the **Kerberos** database. This database contains the keys of all principals encrypted with a master password. You are not required to remember this password, it will be stored in a file (`/var/heimdal/m-key`). To create the master key, run `kstash` and enter a password.

Once the master key has been created, you can initialize the database using the `kadmin` program with the `-l` option (standing for "local"). This option instructs `kadmin` to modify the database files directly rather than going through the `kadmind` network service. This handles the chicken-and-egg problem of trying to connect to the database before it is created. Once you have the `kadmin` prompt, use the `init` command to create your realms initial database.

Lastly, while still in `kadmin`, create your first principal using the `add` command. Stick to the defaults options for the principal for now, you can always change them later with the `modify` command. Note that you can use the `?` command at any prompt to see the available options.

A sample database creation session is shown below:

```
# 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
```

Now it is time to start up the KDC services. Run `/etc/rc.d/kerberos start` and `/etc/rc.d/kadmind start` to bring up the services. Note that you will not have any kerberized daemons running at this point but you should be able to confirm the that the KDC is functioning by obtaining and listing a ticket for the principal (user) that you just created from the command-line of the KDC itself:

```
% 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
```

The ticket can then be revoked when you have finished:

```
% kdestroy
```

14.7.3 Kerberos enabling a server with Heimdal services

First, we need a copy of the **Kerberos** configuration file, `/etc/krb5.conf`. To do so, simply copy it over to the client computer from the KDC in a secure fashion (using network utilities, such as scp(1), or physically via a floppy disk).

Next you need a `/etc/krb5.keytab` file. This is the major difference between a server providing **Kerberos** enabled daemons and a workstation — the server must have a `keytab` file. This file contains the server's host key, which allows it and the KDC to verify each others identity. It must be transmitted to the server in a secure fashion, as the security of the server can be broken if the key is made public. This explicitly means that transferring it via a clear text channel, such as FTP, is a very bad idea.

Typically, you transfer to the `keytab` to the server using the `kadmin` program. This is handy because you also need to create the host principal (the KDC end of the `krb5.keytab`) using `kadmin`.

Note that you must have already obtained a ticket and that this ticket must be allowed to use the `kadmin` interface in the `kadmind.acl`. See the section titled "Remote administration" in the Heimdal info pages (`info heimdal`) for details on designing access control lists. If you do not want to enable remote `kadmin` access, you can simply securely connect to the KDC (via local console, ssh(1) or **Kerberos** telnet(1)) and perform administration locally using `kadmin -l`.

After installing the `/etc/krb5.conf` file, you can use `kadmin` from the **Kerberos** server. The `add --random-key` command will let you add the server's host principal, and the `ext` command will allow you to extract the server's host principal to its own keytab. For example:

```
# 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
```

Note that the `ext` command (short for "extract") stores the extracted key in `/etc/krb5.keytab` by default.

If you do not have `kadmind` running on the KDC (possibly for security reasons) and thus do not have access to `kadmin` remotely, you can add the host principal (`host/myserver.EXAMPLE.ORG`) directly on the KDC and then extract it to a temporary file (to avoid over-writing the `/etc/krb5.keytab` on the KDC) using something like this:

```
# kadmin
```

```
kadmin> ext --keytab=/tmp/example.keytab host/myserver.example.org
kadmin> exit
```

You can then securely copy the keytab to the server computer (using `scp` or a floppy, for example). Be sure to specify a non-default keytab name to avoid over-writing the keytab on the KDC.

At this point your server can communicate with the KDC (due to its `krb5.conf` file) and it can prove its own identity (due to the `krb5.keytab` file). It is now ready for you to enable some **Kerberos** services. For this example we will enable the `telnet` service by putting a line like this into your `/etc/inetd.conf` and then restarting the inetd(8) service with `/etc/rc.d/inetd restart`:

```
telnet    stream  tcp     nowait  root    /usr/libexec/telnetd  telnetd -a user
```

The critical bit is that the `-a` (for authentication) type is set to user. Consult the telnetd(8) manual page for more details.

14.7.4 Kerberos enabling a client with Heimdal

Setting up a client computer is almost trivially easy. As far as **Kerberos** configuration goes, you only need the **Kerberos** configuration file, located at `/etc/krb5.conf`. Simply securely copy it over to the client computer from the KDC.

Test your client computer by attempting to use `kinit`, `klist`, and `kdestroy` from the client to obtain, show, and then delete a ticket for the principal you created above. You should also be able to use **Kerberos** applications to connect to **Kerberos** enabled servers, though if that does not work and obtaining a ticket does the problem is likely with the server and not with the client or the KDC.

When testing an application like `telnet`, try using a packet sniffer (such as tcpdump(1)) to confirm that your password is not sent in the clear. Try using `telnet` with the `-x` option, which encrypts the entire data stream (similar to `ssh`).

Various non-core **Kerberos** client applications are also installed by default. This is where the “minimal” nature of the base Heimdal installation is felt: `telnet` is the only **Kerberos** enabled service.

The Heimdal port adds some of the missing client applications: **Kerberos** enabled versions of `ftp`, `rsh`, `rcp`, `rlogin`, and a few other less common programs. The MIT port also contains a full suite of **Kerberos** client applications.

14.7.5 User configuration files: `.k5login` and `.k5users`

Users within a realm typically have their **Kerberos** principal (such as `tillman@EXAMPLE.ORG`) mapped to a local user account (such as a local account named `tillman`). Client applications such as `telnet` usually do not require a user name or a principal.

Occasionally, however, you want to grant access to a local user account to someone who does not have a matching **Kerberos** principal. For example, `tillman@EXAMPLE.ORG` may need access to the local user account `webdevelopers`. Other principals may also need access to that local account.

The `.k5login` and `.k5users` files, placed in a users home directory, can be used similar to a powerful combination of `.hosts` and `.rhosts`, solving this problem. For example, if a `.k5login` with the following contents:

```
tillman@example.org
```

```
jdoe@example.org
```

Were to be placed into the home directory of the local user `webdevelopers` then both principals listed would have access to that account without requiring a shared password.

Reading the manual pages for these commands is recommended. Note that the `ksu` manual page covers `.k5users`.

14.7.6 Kerberos Tips, Tricks, and Troubleshooting

- When using either the Heimdal or MIT **Kerberos** ports ensure that your `PATH` environment variable lists the **Kerberos** versions of the client applications before the system versions.
- Do all the computers in your realm have synchronized time settings? If not, authentication may fail. Section 29.10 describes how to synchronize clocks using NTP.
- MIT and Heimdal inter-operate nicely. Except for `kadmin`, the protocol for which is not standardized.
- If you change your hostname, you also need to change your `host/` principal and update your keytab. This also applies to special keytab entries like the `www/` principal used for Apache's `www/mod_auth_kerb`.
- All hosts in your realm must be resolvable (both forwards and reverse) in DNS (or `/etc/hosts` as a minimum). CNAMEs will work, but the A and PTR records must be correct and in place. The error message is not very intuitive: `Kerberos5 refuses authentication because Read req failed: Key table entry not found`.
- Some operating systems that may being acting as clients to your KDC do not set the permissions for `ksu` to be setuid `root`. This means that `ksu` does not work, which is a good security idea but annoying. This is not a KDC error.
- With MIT **Kerberos**, if you want to allow a principal to have a ticket life longer than the default ten hours, you must use `modify_principal` in `kadmin` to change the maxlife of both the principal in question and the `krbtgt` principal. Then the principal can use the `-l` option with `kinit` to request a ticket with a longer lifetime.
-

  Note: If you run a packet sniffer on your KDC to add in troubleshooting and then run `kinit` from a workstation, you will notice that your TGT is sent immediately upon running `kinit` — even before you type your password! The explanation is that the **Kerberos** server freely transmits a TGT (Ticket Granting Ticket) to any unauthorized request; however, every TGT is encrypted in a key derived from the user's password. Therefore, when a user types their password it is not being sent to the KDC, it is being used to decrypt the TGT that `kinit` already obtained. If the decryption process results in a valid ticket with a valid time stamp, the user has valid **Kerberos** credentials. These credentials include a session key for establishing secure communications with the **Kerberos** server in the future, as well as the actual ticket-granting ticket, which is actually encrypted with the **Kerberos** server's own key. This second layer of encryption is unknown to the user, but it is what allows the **Kerberos** server to verify the authenticity of each TGT.

- If you want to use long ticket lifetimes (a week, for example) and you are using **OpenSSH** to connect to the machine where your ticket is stored, make sure that **Kerberos** `TicketCleanup` is set to `no` in your `sshd_config` or else your tickets will be deleted when you log out.

- Remember that host principals can have a longer ticket lifetime as well. If your user principal has a lifetime of a week but the host you are connecting to has a lifetime of nine hours, you will have an expired host principal in your cache and the ticket cache will not work as expected.
- When setting up a `krb5.dict` file to prevent specific bad passwords from being used (the manual page for `kadmind` covers this briefly), remember that it only applies to principals that have a password policy assigned to them. The `krb5.dict` files format is simple: one string per line. Creating a symbolic link to `/usr/share/dict/words` might be useful.

14.7.7 Differences with the MIT port

The major difference between the MIT and Heimdal installs relates to the `kadmin` program which has a different (but equivalent) set of commands and uses a different protocol. This has a large implications if your KDC is MIT as you will not be able to use the Heimdal `kadmin` program to administer your KDC remotely (or vice versa, for that matter).

The client applications may also take slightly different command line options to accomplish the same tasks. Following the instructions on the MIT **Kerberos** web site (http://web.mit.edu/Kerberos/www/) is recommended. Be careful of path issues: the MIT port installs into `/usr/local/` by default, and the "normal" system applications may be run instead of MIT if your `PATH` environment variable lists the system directories first.

> **Note:** With the MIT `security/krb5` port that is provided by FreeBSD, be sure to read the `/usr/local/share/doc/krb5/README.FreeBSD` file installed by the port if you want to understand why logins via `telnetd` and `klogind` behave somewhat oddly. Most importantly, correcting the "incorrect permissions on cache file" behavior requires that the `login.krb5` binary be used for authentication so that it can properly change ownership for the forwarded credentials.

The `rc.conf` must also be modified to contain the following configuration:

```
kerberos5_server="/usr/local/sbin/krb5kdc"
kadmind5_server="/usr/local/sbin/kadmind"
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

This is done because the applications for MIT kerberos installs binaries in the `/usr/local` hierarchy.

14.7.8 Mitigating limitations found in Kerberos

14.7.8.1 Kerberos is an all-or-nothing approach

Every service enabled on the network must be modified to work with **Kerberos** (or be otherwise secured against network attacks) or else the users credentials could be stolen and re-used. An example of this would be **Kerberos** enabling all remote shells (via `rsh` and `telnet`, for example) but not converting the POP3 mail server which sends passwords in plain text.

14.7.8.2 Kerberos is intended for single-user workstations

In a multi-user environment, **Kerberos** is less secure. This is because it stores the tickets in the `/tmp` directory, which is readable by all users. If a user is sharing a computer with several other people simultaneously (i.e. multi-user), it is possible that the user's tickets can be stolen (copied) by another user.

This can be overcome with the `-c` filename command-line option or (preferably) the `KRB5CCNAME` environment variable, but this is rarely done. In principal, storing the ticket in the users home directory and using simple file permissions can mitigate this problem.

14.7.8.3 The KDC is a single point of failure

By design, the KDC must be as secure as the master password database is contained on it. The KDC should have absolutely no other services running on it and should be physically secured. The danger is high because **Kerberos** stores all passwords encrypted with the same key (the "master" key), which in turn is stored as a file on the KDC.

As a side note, a compromised master key is not quite as bad as one might normally fear. The master key is only used to encrypt the **Kerberos** database and as a seed for the random number generator. As long as access to your KDC is secure, an attacker cannot do much with the master key.

Additionally, if the KDC is unavailable (perhaps due to a denial of service attack or network problems) the network services are unusable as authentication can not be performed, a recipe for a denial-of-service attack. This can alleviated with multiple KDCs (a single master and one or more slaves) and with careful implementation of secondary or fall-back authentication (PAM is excellent for this).

14.7.8.4 Kerberos Shortcomings

Kerberos allows users, hosts and services to authenticate between themselves. It does not have a mechanism to authenticate the KDC to the users, hosts or services. This means that a trojanned `kinit` (for example) could record all user names and passwords. Something like `security/tripwire` or other file system integrity checking tools can alleviate this.

14.7.9 Resources and further information

- The **Kerberos** FAQ (http://www.faqs.org/faqs/Kerberos-faq/general/preamble.html)
- Designing an Authentication System: a Dialog in Four Scenes (http://web.mit.edu/Kerberos/www/dialogue.html)
- RFC 1510, The **Kerberos** Network Authentication Service (V5) (http://www.ietf.org/rfc/rfc1510.txt?number=1510)
- MIT **Kerberos** home page (http://web.mit.edu/Kerberos/www/)
- Heimdal **Kerberos** home page (http://www.pdc.kth.se/heimdal/)

14.8 OpenSSL

One feature that many users overlook is the **OpenSSL** toolkit included in FreeBSD. **OpenSSL** provides an encryption transport layer on top of the normal communications layer; thus allowing it to be intertwined with many network applications and services.

Some uses of **OpenSSL** may include encrypted authentication of mail clients, web based transactions such as credit card payments and more. Many ports such as `www/apache13-ssl`, and `mail/claws-mail` will offer compilation support for building with **OpenSSL**.

> **Note:** In most cases the Ports Collection will attempt to build the `security/openssl` port unless the `WITH_OPENSSL_BASE` make variable is explicitly set to "yes".

The version of **OpenSSL** included in FreeBSD supports Secure Sockets Layer v2/v3 (SSLv2/SSLv3), Transport Layer Security v1 (TLSv1) network security protocols and can be used as a general cryptographic library.

> **Note:** While **OpenSSL** supports the IDEA algorithm, it is disabled by default due to United States patents. To use it, the license should be reviewed and, if the restrictions are acceptable, the `MAKE_IDEA` variable must be set in `make.conf`.

One of the most common uses of **OpenSSL** is to provide certificates for use with software applications. These certificates ensure that the credentials of the company or individual are valid and not fraudulent. If the certificate in question has not been verified by one of the several "Certificate Authorities", or CAs, a warning is usually produced. A Certificate Authority is a company, such as VeriSign (http://www.verisign.com), which will sign certificates in order to validate credentials of individuals or companies. This process has a cost associated with it and is definitely not a requirement for using certificates; however, it can put some of the more paranoid users at ease.

14.8.1 Generating Certificates

To generate a certificate, the following command is available:

```
# 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 '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:PA
Locality Name (eg, city) []:Pittsburgh
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Company
Organizational Unit Name (eg, section) []:Systems Administrator
```

```
Common Name (eg, YOUR name) []:localhost.example.org
Email Address []:trhodes@FreeBSD.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:SOME PASSWORD
An optional company name []:Another Name
```

Notice the response directly after the "Common Name" prompt shows a domain name. This prompt requires a server name to be entered for verification purposes; placing anything but a domain name would yield a useless certificate. Other options, for instance expire time, alternate encryption algorithms, etc. are available. A complete list may be obtained by viewing the openssl(1) manual page.

Two files should now exist in the directory in which the aforementioned command was issued. The certificate request, `req.pem`, may be sent to a certificate authority who will validate the credentials that you entered, sign the request and return the certificate to you. The second file created will be named `cert.pem` and is the private key for the certificate and should be protected at all costs; if this falls in the hands of others it can be used to impersonate you (or your server).

In cases where a signature from a CA is not required, a self signed certificate can be created. First, generate the RSA key:

```
# openssl dsaparam -rand -genkey -out myRSA.key 1024
```

Next, generate the CA key:

```
# openssl gendsa -des3 -out myca.key myRSA.key
```

Use this key to create the certificate:

```
# openssl req -new -x509 -days 365 -key myca.key -out new.crt
```

Two new files should appear in the directory: a certificate authority signature file, `myca.key` and the certificate itself, `new.crt`. These should be placed in a directory, preferably under `/etc`, which is readable only by `root`. Permissions of 0700 should be fine for this and they can be set with the `chmod` utility.

14.8.2 Using Certificates, an Example

So what can these files do? A good use would be to encrypt connections to the **Sendmail** MTA. This would dissolve the use of clear text authentication for users who send mail via the local MTA.

> **Note:** This is not the best use in the world as some MUAs will present the user with an error if they have not installed the certificate locally. Refer to the documentation included with the software for more information on certificate installation.

The following lines should be placed inside the local `.mc` file:

```
dnl SSL Options
define('confCACERT_PATH','/etc/certs')dnl
define('confCACERT','/etc/certs/new.crt')dnl
define('confSERVER_CERT','/etc/certs/new.crt')dnl
```

```
define('confSERVER_KEY','/etc/certs/myca.key')dnl
define('confTLS_SRV_OPTIONS', 'V')dnl
```

Where `/etc/certs/` is the directory to be used for storing the certificate and key files locally. The last few requirements are a rebuild of the local `.cf` file. This is easily achieved by typing `make install` within the `/etc/mail` directory. Follow that up with `make restart` which should start the **Sendmail** daemon.

If all went well there will be no error messages in the `/var/log/maillog` file and **Sendmail** will show up in the process list.

For a simple test, simply connect to the mail server using the telnet(1) utility:

```
# 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.
```

If the "STARTTLS" line appears in the output then everything is working correctly.

14.9 VPN over IPsec

Creating a VPN between two networks, separated by the Internet, using FreeBSD gateways.

14.9.1 Understanding IPsec

This section will guide you through the process of setting up IPsec. In order to set up IPsec, it is necessary that you are familiar with the concepts of building a custom kernel (see Chapter 8).

IPsec is a protocol which sits on top of the Internet Protocol (IP) layer. It allows two or more hosts to communicate in a secure manner (hence the name). The FreeBSD IPsec "network stack" is based on the KAME (http://www.kame.net/) implementation, which has support for both protocol families, IPv4 and IPv6.

IPsec consists of two sub-protocols:

- *Encapsulated Security Payload (ESP)*, protects the IP packet data from third party interference, by encrypting the contents using symmetric cryptography algorithms (like Blowfish, 3DES).
- *Authentication Header (AH)*, protects the IP packet header from third party interference and spoofing, by computing a cryptographic checksum and hashing the IP packet header fields with a secure hashing function. This is then followed by an additional header that contains the hash, to allow the information in the packet to be authenticated.

ESP and AH can either be used together or separately, depending on the environment.

IPsec can either be used to directly encrypt the traffic between two hosts (known as *Transport Mode*); or to build "virtual tunnels" between two subnets, which could be used for secure communication between two corporate networks (known as *Tunnel Mode*). The latter is more commonly known as a *Virtual Private Network (VPN)*. The ipsec(4) manual page should be consulted for detailed information on the IPsec subsystem in FreeBSD.

To add IPsec support to your kernel, add the following options to your kernel configuration file:

```
options   IPSEC        #IP security
device    crypto
```

If IPsec debugging support is desired, the following kernel option should also be added:

```
options   IPSEC_DEBUG  #debug for IP security
```

14.9.2 The Problem

There is no standard for what constitutes a VPN. VPNs can be implemented using a number of different technologies, each of which have their own strengths and weaknesses. This section presents a scenario, and the strategies used for implementing a VPN for this scenario.

14.9.3 The Scenario: Two networks, one home based and one corporate based. Both are connected to the Internet, and expected, via this VPN to behave as one.

The premise is as follows:

- You have at least two sites
- Both sites are using IP internally
- Both sites are connected to the Internet, through a gateway that is running FreeBSD.
- The gateway on each network has at least one public IP address.
- The internal addresses of the two networks can be public or private IP addresses, it does not matter. They just may not collide; e.g.: may not both use `192.168.1.x`.

14.9.4 Configuring IPsec on FreeBSD

To begin, the `security/ipsec-tools` must be installed from the Ports Collection. This third party software package provides a number of applications which will help support the configuration.

The next requirement is to create two gif(4) pseudo-devices which will be used to tunnel packets and allow both networks to communicate properly. As `root`, run the following commands, replacing the *`internal`* and *`external`* items with the real internal and external gateways:

```
# ifconfig gif0 create

# ifconfig gif0 internal1 internal2

# ifconfig gif0 tunnel external1 external2
```

For example, the corporate LAN's public IP is `172.16.5.4` having a private IP of `10.246.38.1`. The home LAN's public IP is `192.168.1.12` with an internal private IP of `10.0.0.5`.

This may seem confusing, so review the following example output from the ifconfig(8) command:

```
Gateway 1:

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 0xffffff00

Gateway 2:

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 0xffffff00
inet6 fe80::250:bfff:fe3a:c1f%gif0 prefixlen 64 scopeid 0x4
```

Once complete, both private IPs should be reachable using the ping(8) command like the following output suggests:

```
priv-net# 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

corp-net# 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
```

As expected, both sides have the ability to send and receive ICMP packets from the privately configured addresses. Next, both gateways must be told how to route packets in order to correctly send traffic from either network. The following command will achieve this goal:

```
# corp-net# route add 10.0.0.0 10.0.0.5 255.255.255.0

# corp-net# route add net 10.0.0.0: gateway 10.0.0.5

# priv-net# route add 10.246.38.0 10.246.38.1 255.255.255.0

# priv-net# route add host 10.246.38.0: gateway 10.246.38.1
```

At this point, internal machines should be reachable from each gateway as well as from machines behind the gateways. This is easily determined from the following example:

```
corp-net# 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

priv-net# 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
```

Setting up the tunnels is the easy part. Configuring a secure link is a much more in depth process. The following configuration uses pre-shared (PSK) RSA keys. Aside from the IP addresses, both `/usr/local/etc/racoon/racoon.conf` files will be identical and look similar to

```
path    pre_shared_key  "/usr/local/etc/racoon/psk.txt"; #location of pre-shared key file
log     debug; #log verbosity setting: set to 'notify' when testing and debugging is complete

padding # options are not to be changed
{
        maximum_length  20;
        randomize       off;
        strict_check    off;
        exclusive_tail  off;
}
```

```
timer # timing options. change as needed
{
        counter         5;
        interval        20 sec;
        persend         1;
#       natt_keepalive  15 sec;
        phase1          30 sec;
        phase2          15 sec;
}

listen # address [port] that racoon will listening on
{
        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 $network/$netmask $type ad
{        # $network must be the two internal networks you are joining.
        pfs_group       1;
        lifetime        time    36000 sec;
        encryption_algorithm    blowfish,3des,des;
        authentication_algorithm        hmac_md5,hmac_sha1;
        compression_algorithm   deflate;
}
```

Explaining every available option, along with those listed in these examples is beyond the scope of this document. There is plenty of relevant information in the **racoon** configuration manual page.

The SPD policies need to be configured so FreeBSD and **racoon** is able to encrypt and decrypt network traffic between hosts.

This task may be undertaken with a simple shell script similar to the following which is on the corporate gateway. This file will be used during system initialization and should be saved as `/usr/local/etc/racoon/setkey.conf`.

```
flush;
spdflush;
# To the home network
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;
```

Once in place, **racoon** may be started on both gateways using the following command:

```
# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf -l /var/log/racoon.log
```

The output should be similar to the following:

```
corp-net# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.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 172.16.5.4[500]-192.168.1.12[500] spi:623b9b3bd2
2006-01-30 01:36:05: INFO: initiate new phase 2 negotiation: 172.16.5.4[0]192.168.1.12[0]
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 192.168.1.12[0]->172.16.5.4[0] spi=28
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
```

To ensure the tunnel is working properly, switch to another console and use tcpdump(1) to view network traffic using the following command. Replace `em0` with the network interface card as required.

```
# tcpdump -i em0 host 172.16.5.4 and dst 192.168.1.12
```

Data similar to the following should appear on the console. If not, there is an issue, and debugging the returned data will be required.

```
01:47:32.021683 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com: ESP(spi=0x02acbf9f,seq
01:47:33.022442 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com: ESP(spi=0x02acbf9f,seq
01:47:34.024218 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com: ESP(spi=0x02acbf9f,seq
```

At this point, both networks should be available and seem to be part of the same network. Most likely both networks are protected by a firewall, as they should be. To allow traffic to flow between them, rules need to be added to pass packets back and forth. For the ipfw(8) firewall, add the following lines to the firewall configuration file:

```
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
```

Note: The rule numbers may need to be altered depending on the current host configuration.

For users of pf(4) or ipf(8), the following rules should do the trick:

```
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
```

Finally, to allow the machine to start support for the VPN during system initialization, add the following lines to `/etc/rc.conf`:

```
ipsec_enable="YES"
ipsec_program="/usr/local/sbin/setkey"
ipsec_file="/usr/local/etc/racoon/setkey.conf" # allows setting up spd policies on boot
racoon_enable="yes"
```

14.10 OpenSSH

OpenSSH is a set of network connectivity tools used to access remote machines securely. It can be used as a direct replacement for `rlogin`, `rsh`, `rcp`, and `telnet`. Additionally, TCP/IP connections can be tunneled/forwarded securely through SSH. **OpenSSH** encrypts all traffic to effectively eliminate eavesdropping, connection hijacking, and other network-level attacks.

OpenSSH is maintained by the OpenBSD project, and is based upon SSH v1.2.12 with all the recent bug fixes and updates. It is compatible with both SSH protocols 1 and 2.

14.10.1 Advantages of Using OpenSSH

Normally, when using telnet(1) or rlogin(1), data is sent over the network in a clear, un-encrypted form. Network sniffers anywhere in between the client and server can steal your user/password information or data transferred in your session. **OpenSSH** offers a variety of authentication and encryption methods to prevent this from happening.

14.10.2 Enabling sshd

The **sshd** is an option presented during a `Standard` install of FreeBSD. To see if **sshd** is enabled, check the `rc.conf` file for:

```
sshd_enable="YES"
```

This will load sshd(8), the daemon program for **OpenSSH**, the next time your system initializes. Alternatively, it is possible to use `/etc/rc.d/sshd` rc(8) script to start **OpenSSH**:

```
/etc/rc.d/sshd start
```

14.10.3 SSH Client

The ssh(1) utility works similarly to rlogin(1).

```
# ssh user@example.com
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'example.com' added to the list of known hosts.
user@example.com's password: *******
```

The login will continue just as it would have if a session was created using `rlogin` or `telnet`. SSH utilizes a key fingerprint system for verifying the authenticity of the server when the client connects. The user is prompted to enter `yes` only when connecting for the first time. Future attempts to login are all verified against the saved fingerprint key. The SSH client will alert you if the saved fingerprint differs from the received fingerprint on future login attempts. The fingerprints are saved in `~/.ssh/known_hosts`, or `~/.ssh/known_hosts2` for SSH v2 fingerprints.

By default, recent versions of the **OpenSSH** servers only accept SSH v2 connections. The client will use version 2 if possible and will fall back to version 1. The client can also be forced to use one or the other by passing it the `-1` or `-2` for version 1 or version 2, respectively. The version 1 compatibility is maintained in the client for backwards compatibility with older versions.

14.10.4 Secure Copy

The scp(1) command works similarly to rcp(1); it copies a file to or from a remote machine, except in a secure fashion.

```
#  scp user@example.com:/COPYRIGHT COPYRIGHT
user@example.com's password: *******
COPYRIGHT            100% |*****************************|  4735
00:00
#
```

Since the fingerprint was already saved for this host in the previous example, it is verified when using scp(1) here.

The arguments passed to scp(1) are similar to cp(1), with the file or files in the first argument, and the destination in the second. Since the file is fetched over the network, through SSH, one or more of the file arguments takes on the form `user@host:`<path_to_remote_file>.

14.10.5 Configuration

The system-wide configuration files for both the **OpenSSH** daemon and client reside within the `/etc/ssh` directory.

`ssh_config` configures the client settings, while `sshd_config` configures the daemon.

Additionally, the `sshd_program` (`/usr/sbin/sshd` by default), and `sshd_flags` `rc.conf` options can provide more levels of configuration.

14.10.6 ssh-keygen

Instead of using passwords, ssh-keygen(1) can be used to generate DSA or RSA keys to authenticate a user:

```
% ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_dsa.
Your public key has been saved in /home/user/.ssh/id_dsa.pub.
The key fingerprint is:
bb:48:db:f2:93:57:80:b6:aa:bc:f5:d5:ba:8f:79:17 user@host.example.com
```

ssh-keygen(1) will create a public and private key pair for use in authentication. The private key is stored in `~/.ssh/id_dsa` or `~/.ssh/id_rsa`, whereas the public key is stored in `~/.ssh/id_dsa.pub` or `~/.ssh/id_rsa.pub`, respectively for DSA and RSA key types. The public key must be placed in the `~/.ssh/authorized_keys` file of the remote machine for both RSA or DSA keys in order for the setup to work.

This will allow connection to the remote machine based upon SSH keys instead of passwords.

If a passphrase is used in ssh-keygen(1), the user will be prompted for a password each time in order to use the private key. ssh-agent(1) can alleviate the strain of repeatedly entering long passphrases, and is explored in the Section 14.10.7 section below.

> **Warning:** The various options and files can be different according to the **OpenSSH** version you have on your system; to avoid problems you should consult the ssh-keygen(1) manual page.

14.10.7 ssh-agent and ssh-add

The ssh-agent(1) and ssh-add(1) utilities provide methods for **SSH** keys to be loaded into memory for use, without needing to type the passphrase each time.

The ssh-agent(1) utility will handle the authentication using the private key(s) that are loaded into it. ssh-agent(1) should be used to launch another application. At the most basic level, it could spawn a shell or at a more advanced level, a window manager.

To use ssh-agent(1) in a shell, first it will need to be spawned with a shell as an argument. Secondly, the identity needs to be added by running ssh-add(1) and providing it the passphrase for the private key. Once these steps have been completed the user will be able to ssh(1) to any host that has the corresponding public key installed. For example:

```
% ssh-agent csh
% ssh-add
Enter passphrase for /home/user/.ssh/id_dsa:
Identity added: /home/user/.ssh/id_dsa (/home/user/.ssh/id_dsa)
%
```

To use ssh-agent(1) in X11, a call to ssh-agent(1) will need to be placed in `~/.xinitrc`. This will provide the ssh-agent(1) services to all programs launched in X11. An example `~/.xinitrc` file might look like this:

```
exec ssh-agent startxfce4
```

This would launch ssh-agent(1), which would in turn launch **XFCE**, every time X11 starts. Then once that is done and X11 has been restarted so that the changes can take effect, simply run ssh-add(1) to load all of your SSH keys.

14.10.8 SSH Tunneling

OpenSSH has the ability to create a tunnel to encapsulate another protocol in an encrypted session.

The following command tells ssh(1) to create a tunnel for **telnet**:

```
% ssh -2 -N -f -L 5023:localhost:23 user@foo.example.com
%
```

The `ssh` command is used with the following options:

`-2`

Forces `ssh` to use version 2 of the protocol. (Do not use if you are working with older SSH servers)

`-N`

Indicates no command, or tunnel only. If omitted, `ssh` would initiate a normal session.

`-f`

Forces `ssh` to run in the background.

`-L`

Indicates a local tunnel in *`localport:remotehost:remoteport`* fashion.

`user@foo.example.com`

The remote SSH server.

An SSH tunnel works by creating a listen socket on `localhost` on the specified port. It then forwards any connection received on the local host/port via the SSH connection to the specified remote host and port.

In the example, port *`5023`* on `localhost` is being forwarded to port *`23`* on `localhost` of the remote machine. Since *`23`* is **telnet**, this would create a secure **telnet** session through an SSH tunnel.

This can be used to wrap any number of insecure TCP protocols such as SMTP, POP3, FTP, etc.

Example 14-1. Using SSH to Create a Secure Tunnel for SMTP

```
% ssh -2 -N -f -L 5025:localhost:25 user@mailserver.example.com
user@mailserver.example.com's password: *****
% telnet localhost 5025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mailserver.example.com ESMTP
```

This can be used in conjunction with an ssh-keygen(1) and additional user accounts to create a more seamless/hassle-free SSH tunneling environment. Keys can be used in place of typing a password, and the tunnels can be run as a separate user.

14.10.8.1 Practical SSH Tunneling Examples

14.10.8.1.1 Secure Access of a POP3 Server

At work, there is an SSH server that accepts connections from the outside. On the same office network resides a mail server running a POP3 server. The network, or network path between your home and office may or may not be completely trustable. Because of this, you need to check your e-mail in a secure manner. The solution is to create an SSH connection to your office's SSH server, and tunnel through to the mail server.

```
% ssh -2 -N -f -L 2110:mail.example.com:110 user@ssh-server.example.com
user@ssh-server.example.com's password: ******
```

When the tunnel is up and running, you can point your mail client to send POP3 requests to `localhost` port 2110. A connection here will be forwarded securely across the tunnel to `mail.example.com`.

14.10.8.1.2 Bypassing a Draconian Firewall

Some network administrators impose extremely draconian firewall rules, filtering not only incoming connections, but outgoing connections. You may be only given access to contact remote machines on ports 22 and 80 for SSH and web surfing.

You may wish to access another (perhaps non-work related) service, such as an Ogg Vorbis server to stream music. If this Ogg Vorbis server is streaming on some other port than 22 or 80, you will not be able to access it.

The solution is to create an SSH connection to a machine outside of your network's firewall, and use it to tunnel to the Ogg Vorbis server.

```
% ssh -2 -N -f -L 8888:music.example.com:8000 user@unfirewalled-system.example.org
user@unfirewalled-system.example.org's password: *******
```

Your streaming client can now be pointed to `localhost` port 8888, which will be forwarded over to `music.example.com` port 8000, successfully evading the firewall.

14.10.9 The `AllowUsers` Users Option

It is often a good idea to limit which users can log in and from where. The `AllowUsers` option is a good way to accomplish this. For example, to only allow the `root` user to log in from `192.168.1.32`, something like this would be appropriate in the `/etc/ssh/sshd_config` file:

```
AllowUsers root@192.168.1.32
```

To allow the user `admin` to log in from anywhere, just list the username by itself:

```
AllowUsers admin
```

Multiple users should be listed on the same line, like so:

```
AllowUsers root@192.168.1.32 admin
```

Note: It is important that you list each user that needs to log in to this machine; otherwise they will be locked out.

After making changes to `/etc/ssh/sshd_config` you must tell sshd(8) to reload its config files, by running:

```
# /etc/rc.d/sshd reload
```

14.10.10 Further Reading

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)

14.11 File System Access Control Lists

In conjunction with file system enhancements like snapshots, FreeBSD offers the security of File System Access Control Lists (ACLs).

Access Control Lists extend the standard UNIX permission model in a highly compatible (POSIX.1e) way. This feature permits an administrator to make use of and take advantage of a more sophisticated security model.

To enable ACL support for UFS file systems, the following:

```
options UFS_ACL
```

must be compiled into the kernel. If this option has not been compiled in, a warning message will be displayed when attempting to mount a file system supporting ACLs. This option is included in the `GENERIC` kernel. ACLs rely on extended attributes being enabled on the file system. Extended attributes are natively supported in the next generation UNIX file system, UFS2.

Note: A higher level of administrative overhead is required to configure extended attributes on UFS1 than on UFS2. The performance of extended attributes on UFS2 is also substantially higher. As a result, UFS2 is generally recommended in preference to UFS1 for use with access control lists.

ACLs are enabled by the mount-time administrative flag, `acls`, which may be added to `/etc/fstab`. The mount-time flag can also be automatically set in a persistent manner using tunefs(8) to modify a superblock ACLs flag in the file system header. In general, it is preferred to use the superblock flag for several reasons:

- The mount-time ACLs flag cannot be changed by a remount (mount(8) `-u`), only by means of a complete umount(8) and fresh mount(8). This means that ACLs cannot be enabled on the root file system after boot. It also means that you cannot change the disposition of a file system once it is in use.

- Setting the superblock flag will cause the file system to always be mounted with ACLs enabled even if there is not an `fstab` entry or if the devices re-order. This prevents accidental mounting of the file system without ACLs enabled, which can result in ACLs being improperly enforced, and hence security problems.

  Note: We may change the ACLs behavior to allow the flag to be enabled without a complete fresh mount(8), but we consider it desirable to discourage accidental mounting without ACLs enabled, because you can shoot your feet quite nastily if you enable ACLs, then disable them, then re-enable them without flushing the extended attributes. In general, once you have enabled ACLs on a file system, they should not be disabled, as the resulting file protections may not be compatible with those intended by the users of the system, and re-enabling ACLs may re-attach the previous ACLs to files that have since had their permissions changed, resulting in other unpredictable behavior.

File systems with ACLs enabled will show a + (plus) sign in their permission settings when viewed. For example:

```
drwx------  2 robert  robert  512 Dec 27 11:54 private
drwxrwx---+ 2 robert  robert  512 Dec 23 10:57 directory1
drwxrwx---+ 2 robert  robert  512 Dec 22 10:20 directory2
drwxrwx---+ 2 robert  robert  512 Dec 27 11:57 directory3
drwxr-xr-x  2 robert  robert  512 Nov 10 11:54 public_html
```

Here we see that the `directory1`, `directory2`, and `directory3` directories are all taking advantage of ACLs. The `public_html` directory is not.

14.11.1 Making Use of ACLs

The file system ACLs can be viewed by the getfacl(1) utility. For instance, to view the ACL settings on the `test` file, one would use the command:

```
% getfacl test
 #file:test
 #owner:1001
 #group:1001
 user::rw-
 group::r--
 other::r--
```

To change the ACL settings on this file, invoke the setfacl(1) utility. Observe:

```
% setfacl -k test
```

The `-k` flag will remove all of the currently defined ACLs from a file or file system. The more preferable method would be to use `-b` as it leaves the basic fields required for ACLs to work.

```
% setfacl -m u:trhodes:rwx,group:web:r--,o::--- test
```

In the aforementioned command, the `-m` option was used to modify the default ACL entries. Since there were no pre-defined entries, as they were removed by the previous command, this will restore the default options and assign the options listed. Take care to notice that if you add a user or group which does not exist on the system, an `Invalid argument` error will be printed to `stdout`.

14.12 Monitoring Third Party Security Issues

In recent years, the security world has made many improvements to how vulnerability assessment is handled. The threat of system intrusion increases as third party utilities are installed and configured for virtually any operating system available today.

Vulnerability assessment is a key factor in security, and while FreeBSD releases advisories for the base system, doing so for every third party utility is beyond the FreeBSD Project's capability. There is a way to mitigate third party vulnerabilities and warn administrators of known security issues. A FreeBSD add on utility known as **Portaudit** exists solely for this purpose.

The `ports-mgmt/portaudit` port polls a database, updated and maintained by the FreeBSD Security Team and ports developers, for known security issues.

To begin using **Portaudit**, one must install it from the Ports Collection:

```
# cd /usr/ports/ports-mgmt/portaudit && make install clean
```

During the install process, the configuration files for periodic(8) will be updated, permitting **Portaudit** output in the daily security runs. Ensure the daily security run emails, which are sent to `root`'s email account, are being read. No more configuration will be required here.

After installation, an administrator can update the database and view known vulnerabilities in installed packages by invoking the following command:

```
# portaudit -Fda
```

> **Note:** The database will automatically be updated during the periodic(8) run; thus, the previous command is completely optional. It is only required for the following examples.

To audit the third party utilities installed as part of the Ports Collection at anytime, an administrator need only run the following command:

```
# portaudit -a
```

Portaudit will produce something like this for vulnerable packages:

```
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.
```

By pointing a web browser to the URL shown, an administrator may obtain more information about the vulnerability in question. This will include versions affected, by FreeBSD Port version, along with other web sites which may contain security advisories.

In short, **Portaudit** is a powerful utility and extremely useful when coupled with the **Portupgrade** port.

14.13 FreeBSD Security Advisories

Like many production quality operating systems, FreeBSD publishes “Security Advisories”. These advisories are usually mailed to the security lists and noted in the Errata only after the appropriate releases have been patched. This section will work to explain what an advisory is, how to understand it, and what measures to take in order to patch a system.

14.13.1 What does an advisory look like?

The FreeBSD security advisories look similar to the one below, taken from the freebsd-security-notifications (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications) mailing list.

```
=============================================================================
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 FreeBSD❻
                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)❼
CVE Name: CVE-XXXX-XXXX❽

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.   Background❾


II.  Problem Description(10)


III. Impact(11)


IV.  Workaround(12)
```

```
V.   Solution(13)

VI.  Correction details(14)

VII. References(15)
```

❶ The `Topic` field indicates exactly what the problem is. It is basically an introduction to the current security advisory and notes the utility with the vulnerability.

❷ The `Category` refers to the affected part of the system which may be one of `core`, `contrib`, or `ports`. The `core` category means that the vulnerability affects a core component of the FreeBSD operating system. The `contrib` category means that the vulnerability affects software contributed to the FreeBSD Project, such as **sendmail**. Finally the `ports` category indicates that the vulnerability affects add on software available as part of the Ports Collection.

❸ The `Module` field refers to the component location, for instance `sys`. In this example, we see that the module, `sys`, is affected; therefore, this vulnerability affects a component used within the kernel.

❹ The `Announced` field reflects the date said security advisory was published, or announced to the world. This means that the security team has verified that the problem does exist and that a patch has been committed to the FreeBSD source code repository.

❺ The `Credits` field gives credit to the individual or organization who noticed the vulnerability and reported it.

❻ The `Affects` field explains which releases of FreeBSD are affected by this vulnerability. For the kernel, a quick look over the output from `ident` on the affected files will help in determining the revision. For ports, the version number is listed after the port name in `/var/db/pkg`. If the system does not sync with the FreeBSD CVS repository and rebuild daily, chances are that it is affected.

❼ The `Corrected` field indicates the date, time, time offset, and release that was corrected.

❽ Reserved for the identification information used to look up vulnerabilities in the Common Vulnerabilities Database system.

❾ The `Background` field gives information on exactly what the affected utility is. Most of the time this is why the utility exists in FreeBSD, what it is used for, and a bit of information on how the utility came to be.

(10) The `Problem Description` field explains the security hole in depth. This can include information on flawed code, or even how the utility could be maliciously used to open a security hole.

(11) The `Impact` field describes what type of impact the problem could have on a system. For example, this could be anything from a denial of service attack, to extra privileges available to users, or even giving the attacker superuser access.

(12) The `Workaround` field offers a feasible workaround to system administrators who may be incapable of upgrading the system. This may be due to time constraints, network availability, or a slew of other reasons. Regardless, security should not be taken lightly, and an affected system should either be patched or the security hole workaround should be implemented.

(13) The `Solution` field offers instructions on patching the affected system. This is a step by step tested and verified method for getting a system patched and working securely.

(14) The `Correction Details` field displays the CVS branch or release name with the periods changed to underscore characters. It also shows the revision number of the affected files within each branch.

(15) The `References` field usually offers sources of other information. This can include web URLs, books, mailing lists, and newsgroups.

14.14 Process Accounting

Process accounting is a security method in which an administrator may keep track of system resources used, their allocation among users, provide for system monitoring, and minimally track a user's commands.

This indeed has its own positive and negative points. One of the positives is that an intrusion may be narrowed down to the point of entry. A negative is the amount of logs generated by process accounting, and the disk space they may require. This section will walk an administrator through the basics of process accounting.

14.14.1 Enable and Utilizing Process Accounting

Before making use of process accounting, it must be enabled. To do this, execute the following commands:

```
# touch /var/account/acct

# accton /var/account/acct

# echo 'accounting_enable="YES"' >> /etc/rc.conf
```

Once enabled, accounting will begin to track CPU stats, commands, etc. All accounting logs are in a non-human readable format and may be viewed using the sa(8) utility. If issued without any options, `sa` will print information relating to the number of per user calls, the total elapsed time in minutes, total CPU and user time in minutes, average number of I/O operations, etc.

To view information about commands being issued, one would use the lastcomm(1) utility. The `lastcomm` command may be used to print out commands issued by users on specific ttys(5), for example:

```
# lastcomm ls
 trhodes ttyp1
```

Would print out all known usage of the `ls` by `trhodes` on the `ttyp1` terminal.

Many other useful options exist and are explained in the lastcomm(1), acct(5) and sa(8) manual pages.

Notes

1. Under FreeBSD the standard login password may be up to 128 characters in length.

Chapter 15
Jails

15.1 Synopsis

This chapter will provide an explanation of what FreeBSD jails are and how to use them. Jails, sometimes referred to as an enhanced replacement of *chroot environments*, are a very powerful tool for system administrators, but their basic usage can also be useful for advanced users.

After reading this chapter, you will know:

- What a jail is, and what purpose it may serve in FreeBSD installations.
- How to build, start, and stop a jail.
- The basics of jail administration, both from inside and outside the jail.

Other sources of useful information about jails are:

- The jail(8) manual page. This is the full reference of the `jail` utility — the administrative tool which can be used in FreeBSD to start, stop, and control FreeBSD jails.
- The mailing lists and their archives. The archives of the FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) and other mailing lists hosted by the FreeBSD list server (http://lists.FreeBSD.org/mailman/listinfo) already contain a wealth of material for jails. It should always be engaging to search the archives, or post a new question to the freebsd-questions (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) mailing list.

15.2 Terms Related to Jails

To facilitate better understanding of parts of the FreeBSD system related to jails, their internals and the way they interact with the rest of FreeBSD, the following terms are used further in this chapter:

chroot(8) (command)

> Utility, which uses chroot(2) FreeBSD system call to change the root directory of a process and all its descendants.

chroot(2) (environment)

> The environment of processes running in a “chroot”. This includes resources such as the part of the file system which is visible, user and group IDs which are available, network interfaces and other IPC mechanisms, etc.

jail(8) (command)

> The system administration utility which allows launching of processes within a jail environment.

host (system, process, user, etc.)

The controlling system of a jail environment. The host system has access to all the hardware resources available, and can control processes both outside of and inside a jail environment. One of the important differences of the host system from a jail is that the limitations which apply to superuser processes inside a jail are not enforced for processes of the host system.

hosted (system, process, user, etc.)

A process, user or other entity, whose access to resources is restricted by an FreeBSD jail.

15.3 Introduction

Since system administration is a difficult and perplexing task, many powerful tools were developed to make life easier for the administrator. These tools mostly provide enhancements of some sort to the way systems are installed, configured and maintained. Part of the tasks which an administrator is expected to do is to properly configure the security of a system, so that it can continue serving its real purpose, without allowing security violations.

One of the tools which can be used to enhance the security of a FreeBSD system are *jails*. Jails were introduced in FreeBSD 4.X by Poul-Henning Kamp <phk@FreeBSD.org>, but were greatly improved in FreeBSD 5.X to make them a powerful and flexible subsystem. Their development still goes on, enhancing their usefulness, performance, reliability, and security.

15.3.1 What is a Jail

BSD-like operating systems have had chroot(2) since the time of 4.2BSD. The chroot(8) utility can be used to change the root directory of a set of processes, creating a safe environment, separate from the rest of the system. Processes created in the chrooted environment can not access files or resources outside of it. For that reason, compromising a service running in a chrooted environment should not allow the attacker to compromise the entire system. The chroot(8) utility is good for easy tasks, which do not require a lot of flexibility or complex and advanced features. Since the inception of the chroot concept, however, many ways have been found to escape from a chrooted environment and, although they have been fixed in modern versions of the FreeBSD kernel, it was clear that chroot(2) was not the ideal solution for securing services. A new subsystem had to be implemented.

This is one of the main reasons why *jails* were developed.

Jails improve on the concept of the traditional chroot(2) environment, in several ways. In a traditional chroot(2) environment, processes are only limited in the part of the file system they can access. The rest of the system resources (like the set of system users, the running processes, or the networking subsystem) are shared by the chrooted processes and the processes of the host system. Jails expand this model by virtualizing not only access to the file system, but also the set of users, the networking subsystem of the FreeBSD kernel and a few other things. A more complete set of fine-grained controls available for tuning the access of a jailed environment is described in Section 15.5.

A jail is characterized by four elements:

- A directory subtree — the starting point from which a jail is entered. Once inside the jail, a process is not permitted to escape outside of this subtree. Traditional security issues which plagued the original chroot(2) design will not affect FreeBSD jails.

- A hostname — the hostname which will be used within the jail. Jails are mainly used for hosting network services, therefore having a descriptive hostname for each jail can really help the system administrator.
- An IP address — this will be assigned to the jail and cannot be changed in any way during the jail's life span. The IP address of a jail is usually an alias address for an existing network interface, but this is not strictly necessary.
- A command — the path name of an executable to run inside the jail. This is relative to the root directory of the jail environment, and may vary a lot, depending on the type of the specific jail environment.

Apart from these, jails can have their own set of users and their own `root` user. Naturally, the powers of the `root` user are limited within the jail environment and, from the point of view of the host system, the jail `root` user is not an omnipotent user. In addition, the `root` user of a jail is not allowed to perform critical operations to the system outside of the associated jail(8) environment. More information about capabilities and restrictions of the `root` user will be discussed in Section 15.5 below.

15.4 Creating and Controlling Jails

Some administrators divide jails into the following two types: "complete" jails, which resemble a real FreeBSD system, and "service" jails, dedicated to one application or service, possibly running with privileges. This is only a conceptual division and the process of building a jail is not affected by it. The jail(8) manual page is quite clear about the procedure for building a jail:

```
# setenv D /here/is/the/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 Selecting a location for a jail is the best starting point. This is where the jail will physically reside within the file system of the jail's host. A good choice can be `/usr/jail/`*`jailname`*, where *`jailname`* is the hostname identifying the jail. The `/usr/` file system usually has enough space for the jail file system, which for "complete" jails is, essentially, a replication of every file present in a default installation of the FreeBSD base system.

❷ If you have already rebuilt your userland using `make world` or `make buildworld`, you can skip this step and install your existing userland into the new jail.

❸ This command will populate the directory subtree chosen as jail's physical location on the file system with the necessary binaries, libraries, manual pages and so on.

❹ The `distribution` target for **make** installs every needed configuration file. In simple words, it installs every installable file of `/usr/src/etc/` to the `/etc` directory of the jail environment: `$D/etc/`.

❺ Mounting the devfs(8) file system inside a jail is not required. On the other hand, any, or almost any application requires access to at least one device, depending on the purpose of the given application. It is very important to control access to devices from inside a jail, as improper settings could permit an attacker to do nasty things in the jail. Control over devfs(8) is managed through rulesets which are described in the devfs(8) and devfs.conf(5) manual pages.

Once a jail is installed, it can be started by using the jail(8) utility. The jail(8) utility takes four mandatory arguments which are described in the Section 15.3.1. Other arguments may be specified too, e.g., to run the jailed process with the credentials of a specific user. The *command* argument depends on the type of the jail; for a *virtual system*, `/etc/rc` is a good choice, since it will replicate the startup sequence of a real FreeBSD system. For a *service* jail, it depends on the service or application that will run within the jail.

Jails are often started at boot time and the FreeBSD `rc` mechanism provides an easy way to do this.

1. A list of the jails which are enabled to start at boot time should be added to the rc.conf(5) file:

   ```
   jail_enable="YES"   # Set to NO to disable starting of any jails
   jail_list="www"     # Space separated list of names of jails
   ```

   Note: Jail names in `jail_list` should contain alphanumeric characters only.

2. For each jail listed in `jail_list`, a group of rc.conf(5) settings, which describe the particular jail, should be added:

   ```
   jail_www_rootdir="/usr/jail/www"     # jail's root directory
   jail_www_hostname="www.example.org"  # jail's hostname
   jail_www_ip="192.168.0.10"           # jail's IP address
   jail_www_devfs_enable="YES"          # mount devfs in the jail
   jail_www_devfs_ruleset="www_ruleset" # devfs ruleset to apply to jail
   ```

   The default startup of jails configured in rc.conf(5), will run the `/etc/rc` script of the jail, which assumes the jail is a complete virtual system. For service jails, the default startup command of the jail should be changed, by setting the `jail_`*`jailname`*`_exec_start` option appropriately.

   Note: For a full list of available options, please see the rc.conf(5) manual page.

The `/etc/rc.d/jail` script can be used to start or stop a jail by hand, if an entry for it exists in `rc.conf`:

```
# /etc/rc.d/jail start www
# /etc/rc.d/jail stop www
```

A clean way to shut down a jail(8) is not available at the moment. This is because commands normally used to accomplish a clean system shutdown cannot be used inside a jail. The best way to shut down a jail is to run the following command from within the jail itself or using the jexec(8) utility from outside the jail:

```
# sh /etc/rc.shutdown
```

More information about this can be found in the jail(8) manual page.

15.5 Fine Tuning and Administration

There are several options which can be set for any jail, and various ways of combining a host FreeBSD system with jails, to produce higher level applications. This section presents:

- Some of the options available for tuning the behavior and security restrictions implemented by a jail installation.
- Some of the high-level applications for jail management, which are available through the FreeBSD Ports Collection, and can be used to implement overall jail-based solutions.

15.5.1 System tools for jail tuning in FreeBSD

Fine tuning of a jail's configuration is mostly done by setting sysctl(8) variables. A special subtree of sysctl exists as a basis for organizing all the relevant options: the `security.jail.*` hierarchy of FreeBSD kernel options. Here is a list of the main jail-related sysctls, complete with their default value. Names should be self-explanatory, but for more information about them, please refer to the jail(8) and sysctl(8) manual pages.

- `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`

These variables can be used by the system administrator of the *host system* to add or remove some of the limitations imposed by default on the `root` user. Note that there are some limitations which cannot be removed. The `root` user is not allowed to mount or unmount file systems from within a jail(8). The `root` inside a jail may not load or unload devfs(8) rulesets, set firewall rules, or do many other administrative tasks which require modifications of in-kernel data, such as setting the `securelevel` of the kernel.

The base system of FreeBSD contains a basic set of tools for viewing information about the active jails, and attaching to a jail to run administrative commands. The jls(8) and jexec(8) commands are part of the base FreeBSD system, and can be used to perform the following simple tasks:

- Print a list of active jails and their corresponding jail identifier (JID), IP address, hostname and path.
- Attach to a running jail, from its host system, and run a command inside the jail or perform administrative tasks inside the jail itself. This is especially useful when the `root` user wants to cleanly shut down a jail. The jexec(8) utility can also be used to start a shell in a jail to do administration in it; for example:

```
# jexec 1 tcsh
```

15.5.2 High-level administrative tools in FreeBSD Ports Collection

Among the many third-party utilities for jail administration, one of the most complete and useful is `sysutils/jailutils`. It is a set of small applications that contribute to jail(8) management. Please refer to its web page for more information.

15.6 Application of Jails

15.6.1 Service Jails

This section is based upon an idea originally presented by Simon L. Nielsen <simon@FreeBSD.org> at http://simon.nitro.dk/service-jails.html, and an updated article written by Ken Tom <locals@gmail.com>. This section illustrates how to set up a FreeBSD system that adds an additional layer of security, using the jail(8) feature. It is also assumed that the given system is at least running RELENG_6_0 and the information provided earlier in this chapter has been well understood.

15.6.1.1 Design

One of the major problems with jails is the management of their upgrade process. This tends to be a problem because every jail has to be rebuilt from scratch whenever it is updated. This is usually not a problem for a single jail, since the update process is fairly simple, but can be quite time consuming and tedious if a lot of jails are created.

> **Warning:** This setup requires advanced experience with FreeBSD and usage of its features. If the presented steps below look too complicated, it is advised to take a look at a simpler system such as `sysutils/ezjail`, which provides an easier method of administering FreeBSD jails and is not as sophisticated as this setup.

This idea has been presented to resolve such issues by sharing as much as is possible between jails, in a safe way — using read-only mount_nullfs(8) mounts, so that updating will be be simpler, and putting single services into individual jails will become more attractive. Additionally, it provides a simple way to add or remove jails as well as a way to upgrade them.

> **Note:** Examples of services in this context are: an HTTP server, a DNS server, a SMTP server, and so forth.

The goals of the setup described in this section are:

- Create a simple and easy to understand jail structure. This implies *not* having to run a full installworld on each and every jail.
- Make it easy to add new jails or remove existing ones.
- Make it easy to update or upgrade existing jails.
- Make it possible to run a customized FreeBSD branch.
- Be paranoid about security, reducing as much as possible the possibility of compromise.
- Save space and inodes, as much as possible.

As it has been already mentioned, this design relies heavily on having a single master template which is read-only (known as **nullfs**) mounted into each jail and one read-write device per jail. A device can be a separate physical disc, a partition, or a vnode backed md(4) device. In this example, we will use read-write **nullfs** mounts.

The file system layout is described in the following list:

- Each jail will be mounted under the `/home/j` directory.

- `/home/j/mroot` is the template for each jail and the read-only partition for all of the jails.
- A blank directory will be created for each jail under the `/home/j` directory.
- Each jail will have a `/s` directory, that will be linked to the read-write portion of the system.
- Each jail shall have its own read-write system that is based upon `/home/j/skel`.
- Each jailspace (read-write portion of each jail) shall be created in `/home/js`.

> **Note:** This assumes that the jails are based under the `/home` partition. This can, of course, be changed to anything else, but this change will have to be reflected in each of the examples below.

15.6.1.2 Creating the Template

This section will describe the steps needed to create the master template that will be the read-only portion for the jails to use.

It is always a good idea to update the FreeBSD system to the latest -RELEASE branch. Check the corresponding Handbook Chapter (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/makeworld.html) to accomplish this task. In the case the update is not feasible, the buildworld will be required in order to be able to proceed. Additionally, the `sysutils/cpdup` package will be required. We will use the portsnap(8) utility to download the FreeBSD Ports Collection. The Handbook Portsnap Chapter (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/portsnap.html) is always good reading for newcomers.

1. First, create a directory structure for the read-only file system which will contain the FreeBSD binaries for our jails, then change directory to the FreeBSD source tree and install the read-only file system to the jail template:

```
# mkdir /home/j /home/j/mroot
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot
```

2. Next, prepare a FreeBSD Ports Collection for the jails as well as a FreeBSD source tree, which is required for **mergemaster**:

```
# 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. Create a skeleton for the read-write portion of the system:

```
# 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. Use **mergemaster** to install missing configuration files. Then get rid of the extra directories that **mergemaster** creates:

```
# 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. Now, symlink the read-write file system to the read-only file system. Please make sure that the symlinks are created in the correct `s/` locations. Real directories or the creation of directories in the wrong locations will cause the installation to fail.

```
# 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. As a last step, create a generic `/home/j/skel/etc/make.conf` with its contents as shown below:

```
WRKDIRPREFIX?=  /s/portbuild
```

Having `WRKDIRPREFIX` set up this way will make it possible to compile FreeBSD ports inside each jail. Remember that the ports directory is part of the read-only system. The custom path for `WRKDIRPREFIX` allows builds to be done in the read-write portion of every jail.

15.6.1.3 Creating Jails

Now that we have a complete FreeBSD jail template, we can setup and configure the jails in `/etc/rc.conf`. This example demonstrates the creation of 3 jails: “NS”, “MAIL” and “WWW”.

1. Put the following lines into the `/etc/fstab` file, so that the read-only template for the jails and the read-write space will be available in the respective jails:

```
/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
```

> **Note:** Partitions marked with a 0 pass number are not checked by fsck(8) during boot, and partitions marked with a 0 dump number are not backed up by dump(8). We do not want **fsck** to check **nullfs** mounts or **dump** to back up the read-only nullfs mounts of the jails. This is why they are marked with “0 0” in the last two columns of each `fstab` entry above.

2. Configure the jails in `/etc/rc.conf`:

```
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"
```

> **Warning:** The reason why the `jail_name_rootdir` variable is set to `/usr/home` instead of `/home` is that the physical path of the `/home` directory on a default FreeBSD installation is `/usr/home`. The `jail_name_rootdir` variable must *not* be set to a path which includes a symbolic link, otherwise the jails will refuse to start. Use the realpath(1) utility to determine a value which should be set to this variable. Please see the FreeBSD-SA-07:01.jail Security Advisory for more information.

3. Create the required mount points for the read-only file system of each jail:

   ```
   # mkdir /home/j/ns /home/j/mail /home/j/www
   ```

4. Install the read-write template into each jail. Note the use of `sysutils/cpdup`, which helps to ensure that a correct copy is done of each directory:

   ```
   # 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. In this phase, the jails are built and prepared to run. First, mount the required file systems for each jail, and then start them using the `/etc/rc.d/jail` script:

   ```
   # mount -a
   # /etc/rc.d/jail start
   ```

The jails should be running now. To check if they have started correctly, use the jls(8) command. Its output should be similar to the following:

```
# 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
```

At this point, it should be possible to log onto each jail, add new users or configure daemons. The `JID` column indicates the jail identification number of each running jail. Use the following command in order to perform administrative tasks in the jail whose `JID` is 3:

```
# jexec 3 tcsh
```

15.6.1.4 Upgrading

In time, there will be a need to upgrade the system to a newer version of FreeBSD, either because of a security issue, or because new features have been implemented which are useful for the existing jails. The design of this setup provides an easy way to upgrade existing jails. Additionally, it minimizes their downtime, as the jails will be brought down only in the very last minute. Also, it provides a way to roll back to the older versions should any problems occur.

1. The first step is to upgrade the host system in the usual manner. Then create a new temporary read-only template in `/home/j/mroot2`.

   ```
   # mkdir /home/j/mroot2
   # cd /usr/src
   # make installworld DESTDIR=/home/j/mroot2
   # cd /home/j/mroot2
   # cpdup /usr/src usr/src
   # mkdir s
   ```

   The `installworld` run creates a few unnecessary directories, which should be removed:

   ```
   # chflags -R 0 var
   # rm -R etc var root usr/local tmp
   ```

2. Recreate the read-write symlinks for the master file system:

   ```
   # 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. The right time to stop the jails is now:

   ```
   # /etc/rc.d/jail stop
   ```

4. Unmount the original file systems:

   ```
   # 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
   ```

   Note: The read-write systems are attached to the read-only system (`/s`) and must be unmounted first.

5. Move the old read-only file system and replace it with the new one. This will serve as a backup and archive of the old read-only file system should something go wrong. The naming convention used here corresponds to when a new read-only file system has been created. Move the original FreeBSD Ports Collection over to the new file system to save some space and inodes:

   ```
   # cd /home/j
   # mv mroot mroot.20060601
   # mv mroot2 mroot
   ```

```
# mv mroot.20060601/usr/ports mroot/usr
```

6. At this point the new read-only template is ready, so the only remaining task is to remount the file systems and start the jails:

```
# mount -a
# /etc/rc.d/jail start
```

Use jls(8) to check if the jails started correctly. Do not forget to run mergemaster in each jail. The configuration files will need to be updated as well as the rc.d scripts.

Chapter 16
Mandatory Access Control

16.1 Synopsis

FreeBSD 5.X introduced new security extensions from the TrustedBSD project based on the POSIX.1e draft. Two of the most significant new security mechanisms are file system Access Control Lists (ACLs) and Mandatory Access Control (MAC) facilities. Mandatory Access Control allows new access control modules to be loaded, implementing new security policies. Some provide protections of a narrow subset of the system, hardening a particular service. Others provide comprehensive labeled security across all subjects and objects. The mandatory part of the definition comes from the fact that the enforcement of the controls is done by administrators and the system, and is not left up to the discretion of users as is done with discretionary access control (DAC, the standard file and System V IPC permissions on FreeBSD).

This chapter will focus on the Mandatory Access Control Framework (MAC Framework), and a set of pluggable security policy modules enabling various security mechanisms.

After reading this chapter, you will know:

- What MAC security policy modules are currently included in FreeBSD and their associated mechanisms.
- What MAC security policy modules implement as well as the difference between a labeled and non-labeled policy.
- How to efficiently configure a system to use the MAC framework.
- How to configure the different security policy modules included with the MAC framework.
- How to implement a more secure environment using the MAC framework and the examples shown.
- How to test the MAC configuration to ensure the framework has been properly implemented.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Chapter 3).
- Be familiar with the basics of kernel configuration/compilation (Chapter 8).
- Have some familiarity with security and how it pertains to FreeBSD (Chapter 14).

> **Warning:** The improper use of the information contained herein may cause loss of system access, aggravation of users, or inability to access the features provided by X11. More importantly, MAC should not be relied upon to completely secure a system. The MAC framework only augments existing security policy; without sound security practices and regular security checks, the system will never be completely secure.
>
> It should also be noted that the examples contained within this chapter are just that, examples. It is not recommended that these particular settings be rolled out on a production system. Implementing the various security policy modules takes a good deal of thought and testing. One who does not fully understand exactly how everything works may find him or herself going back through the entire system and reconfiguring many files or directories.

16.1.1 What Will Not Be Covered

This chapter covers a broad range of security issues relating to the MAC framework. The development of new MAC security policy modules will not be covered. A number of security policy modules included with the MAC framework have specific characteristics which are provided for both testing and new module development. These include the mac_test(4), mac_stub(4) and mac_none(4). For more information on these security policy modules and the various mechanisms they provide, please review the manual pages.

16.2 Key Terms in this Chapter

Before reading this chapter, a few key terms must be explained. This will hopefully clear up any confusion that may occur and avoid the abrupt introduction of new terms and information.

- *compartment*: A compartment is a set of programs and data to be partitioned or separated, where users are given explicit access to specific components of a system. Also, a compartment represents a grouping, such as a work group, department, project, or topic. Using compartments, it is possible to implement a need-to-know security policy.
- *high water mark*: A high water mark policy is one which permits the raising of security levels for the purpose of accessing higher level information. In most cases, the original level is restored after the process is complete. Currently, the FreeBSD MAC framework does not have a policy for this, but the definition is included for completeness.
- *integrity*: Integrity, as a key concept, is the level of trust which can be placed on data. As the integrity of the data is elevated, so does the ability to trust that data.
- *label*: A label is a security attribute which can be applied to files, directories, or other items in the system. It could be considered a confidentiality stamp; when a label is placed on a file it describes the security properties for that specific file and will only permit access by files, users, resources, etc. with a similar security setting. The meaning and interpretation of label values depends on the policy configuration: while some policies might treat a label as representing the integrity or secrecy of an object, other policies might use labels to hold rules for access.
- *level*: The increased or decreased setting of a security attribute. As the level increases, its security is considered to elevate as well.
- *low water mark*: A low water mark policy is one which permits lowering of the security levels for the purpose of accessing information which is less secure. In most cases, the original security level of the user is restored after the process is complete. The only security policy module in FreeBSD to use this is mac_lomac(4).
- *multilabel*: The `multilabel` property is a file system option which can be set in single user mode using the tunefs(8) utility, during the boot operation using the fstab(5) file, or during the creation of a new file system. This option will permit an administrator to apply different MAC labels on different objects. This option only applies to security policy modules which support labeling.
- *object*: An object or system object is an entity through which information flows under the direction of a *subject*. This includes directories, files, fields, screens, keyboards, memory, magnetic storage, printers or any other data storage/moving device. Basically, an object is a data container or a system resource; access to an *object* effectively means access to the data.

- *policy*: A collection of rules which defines how objectives are to be achieved. A *policy* usually documents how certain items are to be handled. This chapter will consider the term *policy* in this context as a *security policy*; i.e. a collection of rules which will control the flow of data and information and define whom will have access to that data and information.
- *sensitivity*: Usually used when discussing MLS. A sensitivity level is a term used to describe how important or secret the data should be. As the sensitivity level increases, so does the importance of the secrecy, or confidentiality of the data.
- *single label*: A single label is when the entire file system uses one label to enforce access control over the flow of data. When a file system has this set, which is any time when the `multilabel` option is not set, all files will conform to the same label setting.
- *subject*: a subject is any active entity that causes information to flow between *objects*; e.g. a user, user processor, system process, etc. On FreeBSD, this is almost always a thread acting in a process on behalf of a user.

16.3 Explanation of MAC

With all of these new terms in mind, consider how the MAC framework augments the security of the system as a whole. The various security policy modules provided by the MAC framework could be used to protect the network and file systems, block users from accessing certain ports and sockets, and more. Perhaps the best use of the policy modules is to blend them together, by loading several security policy modules at a time for a multi-layered security environment. In a multi-layered security environment, multiple policy modules are in effect to keep security in check. This is different to a hardening policy, which typically hardens elements of a system that is used only for specific purposes. The only downside is administrative overhead in cases of multiple file system labels, setting network access control user by user, etc.

These downsides are minimal when compared to the lasting effect of the framework; for instance, the ability to pick and choose which policies are required for a specific configuration keeps performance overhead down. The reduction of support for unneeded policies can increase the overall performance of the system as well as offer flexibility of choice. A good implementation would consider the overall security requirements and effectively implement the various security policy modules offered by the framework.

Thus a system utilizing MAC features should at least guarantee that a user will not be permitted to change security attributes at will; all user utilities, programs and scripts must work within the constraints of the access rules provided by the selected security policy modules; and that total control of the MAC access rules are in the hands of the system administrator.

It is the sole duty of the system administrator to carefully select the correct security policy modules. Some environments may need to limit access control over the network; in these cases, the mac_portacl(4), mac_ifoff(4) and even mac_biba(4) policy modules might make good starting points. In other cases, strict confidentiality of file system objects might be required. Policy modules such as mac_bsdextended(4) and mac_mls(4) exist for this purpose.

Policy decisions could be made based on network configuration. Perhaps only certain users should be permitted access to facilities provided by ssh(1) to access the network or the Internet. The mac_portacl(4) would be the policy module of choice for these situations. But what should be done in the case of file systems? Should all access to certain directories be severed from other groups or specific users? Or should we limit user or utility access to specific files by setting certain objects as classified?

In the file system case, access to objects might be considered confidential to some users, but not to others. For an example, a large development team might be broken off into smaller groups of individuals. Developers in project A might not be permitted to access objects written by developers in project B. Yet they might need to access objects

created by developers in project C; that is quite a situation indeed. Using the different security policy modules provided by the MAC framework; users could be divided into these groups and then given access to the appropriate areas without fear of information leakage.

Thus, each security policy module has a unique way of dealing with the overall security of a system. Module selection should be based on a well thought out security policy. In many cases, the overall policy may need to be revised and reimplemented on the system. Understanding the different security policy modules offered by the MAC framework will help administrators choose the best policies for their situations.

The default FreeBSD kernel does not include the option for the MAC framework; thus the following kernel option must be added before trying any of the examples or information in this chapter:

```
options MAC
```

And the kernel will require a rebuild and a reinstall.

> **Caution:** While the various manual pages for MAC policy modules state that they may be built into the kernel, it is possible to lock the system out of the network and more. Implementing MAC is much like implementing a firewall, care must be taken to prevent being completely locked out of the system. The ability to revert back to a previous configuration should be considered while the implementation of MAC remotely should be done with extreme caution.

16.4 Understanding MAC Labels

A MAC label is a security attribute which may be applied to subjects and objects throughout the system.

When setting a label, the user must be able to comprehend what it is, exactly, that is being done. The attributes available on an object depend on the policy module loaded, and that policy modules interpret their attributes in different ways. If improperly configured due to lack of comprehension, or the inability to understand the implications, the result will be the unexpected and perhaps, undesired, behavior of the system.

The security label on an object is used as a part of a security access control decision by a policy. With some policies, the label by itself contains all information necessary to make a decision; in other models, the labels may be processed as part of a larger rule set, etc.

For instance, setting the label of `biba/low` on a file will represent a label maintained by the Biba security policy module, with a value of “low”.

A few policy modules which support the labeling feature in FreeBSD offer three specific predefined labels. These are the low, high, and equal labels. Although they enforce access control in a different manner with each policy module, you can be sure that the low label will be the lowest setting, the equal label will set the subject or object to be disabled or unaffected, and the high label will enforce the highest setting available in the Biba and MLS policy modules.

Within single label file system environments, only one label may be used on objects. This will enforce one set of access permissions across the entire system and in many environments may be all that is required. There are a few cases where multiple labels may be set on objects or subjects in the file system. For those cases, the `multilabel` option may be passed to tunefs(8).

In the case of Biba and MLS, a numeric label may be set to indicate the precise level of hierarchical control. This numeric level is used to partition or sort information into different groups of say, classification only permitting access to that group or a higher group level.

In most cases the administrator will only be setting up a single label to use throughout the file system.

Hey wait, this is similar to DAC! I thought MAC gave control strictly to the administrator. That statement still holds true, to some extent as `root` is the one in control and who configures the policies so that users are placed in the appropriate categories/access levels. Alas, many policy modules can restrict the `root` user as well. Basic control over objects will then be released to the group, but `root` may revoke or modify the settings at any time. This is the hierarchal/clearance model covered by policies such as Biba and MLS.

16.4.1 Label Configuration

Virtually all aspects of label policy module configuration will be performed using the base system utilities. These commands provide a simple interface for object or subject configuration or the manipulation and verification of the configuration.

All configuration may be done by use of the setfmac(8) and setpmac(8) utilities. The `setfmac` command is used to set MAC labels on system objects while the `setpmac` command is used to set the labels on system subjects. Observe:

```
# setfmac biba/high test
```

If no errors occurred with the command above, a prompt will be returned. The only time these commands are not quiescent is when an error occurred; similarly to the chmod(1) and chown(8) commands. In some cases this error may be a `Permission denied` and is usually obtained when the label is being set or modified on an object which is restricted.[1] The system administrator may use the following commands to overcome this:

```
# setfmac biba/high test
Permission denied
# setpmac biba/low setfmac biba/high test
# getfmac test
test: biba/high
```

As we see above, `setpmac` can be used to override the policy module's settings by assigning a different label to the invoked process. The `getpmac` utility is usually used with currently running processes, such as **sendmail**: although it takes a process ID in place of a command the logic is extremely similar. If users attempt to manipulate a file not in their access, subject to the rules of the loaded policy modules, the `Operation not permitted` error will be displayed by the `mac_set_link` function.

16.4.1.1 Common Label Types

For the mac_biba(4), mac_mls(4) and mac_lomac(4) policy modules, the ability to assign simple labels is provided. These take the form of high, equal and low, what follows is a brief description of what these labels provide:

- The `low` label is considered the lowest label setting an object or subject may have. Setting this on objects or subjects will block their access to objects or subjects marked high.
- The `equal` label should only be placed on objects considered to be exempt from the policy.
- The `high` label grants an object or subject the highest possible setting.

With respect to each policy module, each of those settings will instate a different information flow directive. Reading the proper manual pages will further explain the traits of these generic label configurations.

16.4.1.1.1 Advanced Label Configuration

Numeric grade labels are used for `comparison:compartment+compartment`; thus the following:

```
biba/10:2+3+6(5:2+3-20:2+3+4+5+6)
```

May be interpreted as:

“Biba Policy Label”/“Grade 10” :“Compartments 2, 3 and 6”: (“grade 5 ...”)

In this example, the first grade would be considered the “effective grade” with “effective compartments”, the second grade is the low grade and the last one is the high grade. In most configurations these settings will not be used; indeed, they offered for more advanced configurations.

When applied to system objects, they will only have a current grade/compartments as opposed to system subjects as they reflect the range of available rights in the system, and network interfaces, where they are used for access control.

The grade and compartments in a subject and object pair are used to construct a relationship referred to as “dominance”, in which a subject dominates an object, the object dominates the subject, neither dominates the other, or both dominate each other. The “both dominate” case occurs when the two labels are equal. Due to the information flow nature of Biba, you have rights to a set of compartments, “need to know”, that might correspond to projects, but objects also have a set of compartments. Users may have to subset their rights using `su` or `setpmac` in order to access objects in a compartment from which they are not restricted.

16.4.1.2 Users and Label Settings

Users themselves are required to have labels so that their files and processes may properly interact with the security policy defined on the system. This is configured through the `login.conf` file by use of login classes. Every policy module that uses labels will implement the user class setting.

An example entry containing every policy module setting is displayed below:

```
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]:
```

The `label` option is used to set the user class default label which will be enforced by MAC. Users will never be permitted to modify this value, thus it can be considered not optional in the user case. In a real configuration, however, the administrator will never wish to enable every policy module. It is recommended that the rest of this chapter be reviewed before any of this configuration is implemented.

> **Note:** Users may change their label after the initial login; however, this change is subject constraints of the policy. The example above tells the Biba policy that a process's minimum integrity is 5, its maximum is 15, but the default effective label is 10. The process will run at 10 until it chooses to change label, perhaps due to the user using the setpmac command, which will be constrained by Biba to the range set at login.

In all cases, after a change to `login.conf`, the login class capability database must be rebuilt using `cap_mkdb` and this will be reflected throughout every forthcoming example or discussion.

It is useful to note that many sites may have a particularly large number of users requiring several different user classes. In depth planning is required as this may get extremely difficult to manage.

Future versions of FreeBSD will include a new way to deal with mapping users to labels; however, this will not be available until some time after FreeBSD 5.3.

16.4.1.3 Network Interfaces and Label Settings

Labels may also be set on network interfaces to help control the flow of data across the network. In all cases they function in the same way the policies function with respect to objects. Users at high settings in `biba`, for example, will not be permitted to access network interfaces with a label of low.

The `maclabel` may be passed to `ifconfig` when setting the MAC label on network interfaces. For example:

```
# ifconfig bge0 maclabel biba/equal
```

will set the MAC label of `biba/equal` on the bge(4) interface. When using a setting similar to `biba/high(low-high)` the entire label should be quoted; otherwise an error will be returned.

Each policy module which supports labeling has a tunable which may be used to disable the MAC label on network interfaces. Setting the label to `equal` will have a similar effect. Review the output from `sysctl`, the policy manual pages, or even the information found later in this chapter for those tunables.

16.4.2 Singlelabel or Multilabel?

By default the system will use the `singlelabel` option. But what does this mean to the administrator? There are several differences which, in their own right, offer pros and cons to the flexibility in the systems security model.

The `singlelabel` only permits for one label, for instance `biba/high` to be used for each subject or object. It provides for lower administration overhead but decreases the flexibility of policies which support labeling. Many administrators may want to use the `multilabel` option in their security policy.

The `multilabel` option will permit each subject or object to have its own independent MAC label in place of the standard `singlelabel` option which will allow only one label throughout the partition. The `multilabel` and

`single` label options are only required for the policies which implement the labeling feature, including the Biba, Lomac, MLS and SEBSD policies.

In many cases, the `multilabel` may not need to be set at all. Consider the following situation and security model:

- FreeBSD web-server using the MAC framework and a mix of the various policies.
- This machine only requires one label, `biba/high`, for everything in the system. Here the file system would not require the `multilabel` option as a single label will always be in effect.
- But, this machine will be a web server and should have the web server run at `biba/low` to prevent write up capabilities. The Biba policy and how it works will be discussed later, so if the previous comment was difficult to interpret just continue reading and return. The server could use a separate partition set at `biba/low` for most if not all of its runtime state. Much is lacking from this example, for instance the restrictions on data, configuration and user settings; however, this is just a quick example to prove the aforementioned point.

If any of the non-labeling policies are to be used, then the `multilabel` option would never be required. These include the `seeotheruids`, `portacl` and `partition` policies.

It should also be noted that using `multilabel` with a partition and establishing a security model based on `multilabel` functionality could open the doors for higher administrative overhead as everything in the file system would have a label. This includes directories, files, and even device nodes.

The following command will set `multilabel` on the file systems to have multiple labels. This may only be done in single user mode:

```
# tunefs -l enable /
```

This is not a requirement for the swap file system.

> **Note:** Some users have experienced problems with setting the `multilabel` flag on the root partition. If this is the case, please review the Section 16.17 of this chapter.

16.5 Planning the Security Configuration

Whenever a new technology is implemented, a planning phase is always a good idea. During the planning stages, an administrator should in general look at the “big picture”, trying to keep in view at least the following:

- The implementation requirements;
- The implementation goals;

For MAC installations, these include:

- How to classify information and resources available on the target systems.
- What sorts of information or resources to restrict access to along with the type of restrictions that should be applied.
- Which MAC module or modules will be required to achieve this goal.

It is always possible to reconfigure and change the system resources and security settings, it is quite often very inconvenient to search through the system and fix existing files and user accounts. Planning helps to ensure a trouble-free and efficient trusted system implementation. A trial run of the trusted system, including the configuration, is often vital and definitely beneficial *before* a MAC implementation is used on production systems. The idea of just letting loose on a system with MAC is like setting up for failure.

Different environments may have explicit needs and requirements. Establishing an in depth and complete security profile will decrease the need of changes once the system goes live. As such, the future sections will cover the different modules available to administrators; describe their use and configuration; and in some cases provide insight on what situations they would be most suitable for. For instance, a web server might roll out the mac_biba(4) and mac_bsdextended(4) policies. In other cases, a machine with very few local users, the mac_partition(4) might be a good choice.

16.6 Module Configuration

Every module included with the MAC framework may be either compiled into the kernel as noted above or loaded as a run-time kernel module. The recommended method is to add the module name to the `/boot/loader.conf` file so that it will load during the initial boot operation.

The following sections will discuss the various MAC modules and cover their features. Implementing them into a specific environment will also be a consideration of this chapter. Some modules support the use of labeling, which is controlling access by enforcing a label such as “this is allowed and this is not”. A label configuration file may control how files may be accessed, network communication can be exchanged, and more. The previous section showed how the `multilabel` flag could be set on file systems to enable per-file or per-partition access control.

A single label configuration would enforce only one label across the system, that is why the `tunefs` option is called `multilabel`.

16.7 The MAC seeotheruids Module

Module name: `mac_seeotheruids.ko`

Kernel configuration line: `options MAC_SEEOTHERUIDS`

Boot option: `mac_seeotheruids_load="YES"`

The mac_seeotheruids(4) module mimics and extends the `security.bsd.see_other_uids` and `security.bsd.see_other_gids` `sysctl` tunables. This option does not require any labels to be set before configuration and can operate transparently with the other modules.

After loading the module, the following `sysctl` tunables may be used to control the features:

- `security.mac.seeotheruids.enabled` will enable the module’s features and use the default settings. These default settings will deny users the ability to view processes and sockets owned by other users.
- `security.mac.seeotheruids.specificgid_enabled` will allow a certain group to be exempt from this policy. To exempt specific groups from this policy, use the `security.mac.seeotheruids.specificgid=`*`XXX`* `sysctl` tunable. In the above example, the *`XXX`* should be replaced with the numeric group ID to be exempted.
- `security.mac.seeotheruids.primarygroup_enabled` is used to exempt specific primary groups from this policy. When using this tunable, the `security.mac.seeotheruids.specificgid_enabled` may not be set.

16.8 The MAC bsdextended Module

Module name: `mac_bsdextended.ko`

Kernel configuration line: `options MAC_BSDEXTENDED`

Boot option: `mac_bsdextended_load="YES"`

The mac_bsdextended(4) module enforces the file system firewall. This module's policy provides an extension to the standard file system permissions model, permitting an administrator to create a firewall-like ruleset to protect files, utilities, and directories in the file system hierarchy. When access to a file system object is attempted, the list of rules is iterated until either a matching rule is located or the end is reached. This behavior may be changed by the use of a sysctl(8) parameter, security.mac.bsdextended.firstmatch_enabled. Similar to other firewall modules in FreeBSD, a file containing access control rules can be created and read by the system at boot time using an rc.conf(5) variable.

The rule list may be entered using a utility, ugidfw(8), that has a syntax similar to that of ipfw(8). More tools can be written by using the functions in the libugidfw(3) library.

Extreme caution should be taken when working with this module; incorrect use could block access to certain parts of the file system.

16.8.1 Examples

After the mac_bsdextended(4) module has been loaded, the following command may be used to list the current rule configuration:

```
# ugidfw list
0 slots, 0 rules
```

As expected, there are no rules defined. This means that everything is still completely accessible. To create a rule which will block all access by users but leave `root` unaffected, simply run the following command:

```
# ugidfw add subject not uid root new object not uid root mode n
```

This is a very bad idea as it will block all users from issuing even the most simple commands, such as `ls`. A more patriotic list of rules might be:

```
# ugidfw set 2 subject uid user1 object uid user2 mode n
# ugidfw set 3 subject uid user1 object gid user2 mode n
```

This will block any and all access, including directory listings, to *`user2`*'s home directory from the username `user1`.

In place of `user1`, the `not uid` *`user2`* could be passed. This will enforce the same access restrictions above for all users in place of just one user.

> **Note:** The `root` user will be unaffected by these changes.

This should provide a general idea of how the mac_bsdextended(4) module may be used to help fortify a file system. For more information, see the mac_bsdextended(4) and the ugidfw(8) manual pages.

16.9 The MAC ifoff Module

Module name: `mac_ifoff.ko`

Kernel configuration line: `options MAC_IFOFF`

Boot option: `mac_ifoff_load="YES"`

The mac_ifoff(4) module exists solely to disable network interfaces on the fly and keep network interfaces from being brought up during the initial system boot. It does not require any labels to be set up on the system, nor does it have a dependency on other MAC modules.

Most of the control is done through the `sysctl` tunables listed below.

- `security.mac.ifoff.lo_enabled` will enable/disable all traffic on the loopback (lo(4)) interface.
- `security.mac.ifoff.bpfrecv_enabled` will enable/disable all traffic on the Berkeley Packet Filter interface (bpf(4))
- `security.mac.ifoff.other_enabled` will enable/disable traffic on all other interfaces.

One of the most common uses of mac_ifoff(4) is network monitoring in an environment where network traffic should not be permitted during the boot sequence. Another suggested use would be to write a script which uses `security/aide` to automatically block network traffic if it finds new or altered files in protected directories.

16.10 The MAC portacl Module

Module name: `mac_portacl.ko`

Kernel configuration line: `MAC_PORTACL`

Boot option: `mac_portacl_load="YES"`

The mac_portacl(4) module is used to limit binding to local TCP and UDP ports using a variety of `sysctl` variables. In essence mac_portacl(4) makes it possible to allow non-`root` users to bind to specified privileged ports, i.e. ports fewer than 1024.

Once loaded, this module will enable the MAC policy on all sockets. The following tunables are available:

- `security.mac.portacl.enabled` will enable/disable the policy completely.
- `security.mac.portacl.port_high` will set the highest port number that mac_portacl(4) will enable protection for.
- `security.mac.portacl.suser_exempt` will, when set to a non-zero value, exempt the `root` user from this policy.
- `security.mac.portacl.rules` will specify the actual mac_portacl policy; see below.

The actual `mac_portacl` policy, as specified in the `security.mac.portacl.rules` sysctl, is a text string of the form: `rule[,rule,...]` with as many rules as needed. Each rule is of the form: `idtype:id:protocol:port`. The *`idtype`* parameter can be `uid` or `gid` and used to interpret the *`id`* parameter as either a user id or group id, respectively. The *`protocol`* parameter is used to determine if the rule should apply to TCP or UDP by setting the parameter to `tcp` or `udp`. The final *`port`* parameter is the port number to allow the specified user or group to bind to.

Note: Since the ruleset is interpreted directly by the kernel only numeric values can be used for the user ID, group ID, and port parameters. I.e. user, group, and port service names cannot be used.

By default, on UNIX-like systems, ports fewer than 1024 can only be used by/bound to privileged processes, i.e. those run as `root`. For mac_portacl(4) to allow non-privileged processes to bind to ports below 1024 this standard UNIX restriction has to be disabled. This can be accomplished by setting the sysctl(8) variables `net.inet.ip.portrange.reservedlow` and `net.inet.ip.portrange.reservedhigh` to zero.

See the examples below or review the mac_portacl(4) manual page for further information.

16.10.1 Examples

The following examples should illuminate the above discussion a little better:

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0 net.inet.ip.portrange.reservedhigh=0
```

First we set mac_portacl(4) to cover the standard privileged ports and disable the normal UNIX bind restrictions.

```
# sysctl security.mac.portacl.suser_exempt=1
```

The `root` user should not be crippled by this policy, thus set the `security.mac.portacl.suser_exempt` to a non-zero value. The mac_portacl(4) module has now been set up to behave the same way UNIX-like systems behave by default.

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

Allow the user with UID 80 (normally the `www` user) to bind to port 80. This can be used to allow the `www` user to run a web server without ever having `root` privilege.

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

Permit the user with the UID of 1001 to bind to the TCP ports 110 (“pop3”) and 995 (“pop3s”). This will permit this user to start a server that accepts connections on ports 110 and 995.

16.11 The MAC partition Module

Module name: `mac_partition.ko`

Kernel configuration line: `options MAC_PARTITION`

Boot option: `mac_partition_load="YES"`

The mac_partition(4) policy will drop processes into specific “partitions” based on their MAC label. Think of it as a special type of jail(8), though that is hardly a worthy comparison.

This is one module that should be added to the loader.conf(5) file so that it loads and enables the policy during the boot process.

Most configuration for this policy is done using the setpmac(8) utility which will be explained below. The following `sysctl` tunable is available for this policy:

- `security.mac.partition.enabled` will enable the enforcement of MAC process partitions.

When this policy is enabled, users will only be permitted to see their processes, and any others within their partition, but will not be permitted to work with utilities outside the scope of this partition. For instance, a user in the `insecure` class above will not be permitted to access the `top` command as well as many other commands that must spawn a process.

To set or drop utilities into a partition label, use the `setpmac` utility:

```
# setpmac partition/13 top
```

This will add the `top` command to the label set on users in the `insecure` class. Note that all processes spawned by users in the `insecure` class will stay in the `partition/13` label.

16.11.1 Examples

The following command will show you the partition label and the process list:

```
# ps Zax
```

This next command will allow the viewing of another user's process partition label and that user's currently running processes:

```
# ps -ZU trhodes
```

> **Note:** Users can see processes in `root`'s label unless the mac_seeotheruids(4) policy is loaded.

A really crafty implementation could have all of the services disabled in `/etc/rc.conf` and started by a script that starts them with the proper labeling set.

> **Note:** The following policies support integer settings in place of the three default labels offered. These options, including their limitations, are further explained in the module manual pages.

16.12 The MAC Multi-Level Security Module

Module name: `mac_mls.ko`

Kernel configuration line: `options MAC_MLS`

Boot option: `mac_mls_load="YES"`

The mac_mls(4) policy controls access between subjects and objects in the system by enforcing a strict information flow policy.

In MLS environments, a "clearance" level is set in each subject or objects label, along with compartments. Since these clearance or sensibility levels can reach numbers greater than six thousand; it would be a daunting task for any system administrator to thoroughly configure each subject or object. Thankfully, three "instant" labels are already included in this policy.

These labels are `mls/low`, `mls/equal` and `mls/high`. Since these labels are described in depth in the manual page, they will only get a brief description here:

- The `mls/low` label contains a low configuration which permits it to be dominated by all other objects. Anything labeled with `mls/low` will have a low clearance level and not be permitted to access information of a higher level. In addition, this label will prevent objects of a higher clearance level from writing or passing information on to them.
- The `mls/equal` label should be placed on objects considered to be exempt from the policy.
- The `mls/high` label is the highest level of clearance possible. Objects assigned this label will hold dominance over all other objects in the system; however, they will not permit the leaking of information to objects of a lower class.

MLS provides for:

- A hierarchical security level with a set of non hierarchical categories;
- Fixed rules: no read up, no write down (a subject can have read access to objects on its own level or below, but not above. Similarly, a subject can have write access to objects on its own level or above but not beneath.);
- Secrecy (preventing inappropriate disclosure of data);
- Basis for the design of systems that concurrently handle data at multiple sensitivity levels (without leaking information between secret and confidential).

The following `sysctl` tunables are available for the configuration of special services and interfaces:

- `security.mac.mls.enabled` is used to enable/disable the MLS policy.
- `security.mac.mls.ptys_equal` will label all pty(4) devices as `mls/equal` during creation.
- `security.mac.mls.revocation_enabled` is used to revoke access to objects after their label changes to a label of a lower grade.
- `security.mac.mls.max_compartments` is used to set the maximum number of compartment levels with objects; basically the maximum compartment number allowed on a system.

To manipulate the MLS labels, the setfmac(8) command has been provided. To assign a label to an object, issue the following command:

```
# setfmac mls/5 test
```

To get the MLS label for the file `test` issue the following command:

```
# getfmac test
```

This is a summary of the MLS policy's features. Another approach is to create a master policy file in `/etc` which specifies the MLS policy information and to feed that file into the `setfmac` command. This method will be explained after all policies are covered.

16.12.1 Planning Mandatory Sensitivity

With the Multi-Level Security Policy Module, an administrator plans for controlling the flow of sensitive information. By default, with its block read up block write down nature, the system defaults everything to a low state.

Everything is accessible and an administrator slowly changes this during the configuration stage; augmenting the confidentiality of the information.

Beyond the three basic label options above, an administrator may group users and groups as required to block the information flow between them. It might be easier to look at the information in clearance levels familiarized with words, for instance classifications such as `Confidential`, `Secret`, and `Top Secret`. Some administrators might just create different groups based on project levels. Regardless of classification method, a well thought out plan must exist before implementing such a restrictive policy.

Some example situations for this security policy module could be an e-commerce web server, a file server holding critical company information, and financial institution environments. The most unlikely place would be a personal workstation with only two or three users.

16.13 The MAC Biba Module

Module name: `mac_biba.ko`

Kernel configuration line: `options MAC_BIBA`

Boot option: `mac_biba_load="YES"`

The mac_biba(4) module loads the MAC Biba policy. This policy works much like that of the MLS policy with the exception that the rules for information flow are slightly reversed. This is said to prevent the downward flow of sensitive information whereas the MLS policy prevents the upward flow of sensitive information; thus, much of this section can apply to both policies.

In Biba environments, an “integrity” label is set on each subject or object. These labels are made up of hierarchal grades, and non-hierarchal components. As an object’s or subject’s grade ascends, so does its integrity.

Supported labels are `biba/low`, `biba/equal`, and `biba/high`; as explained below:

- The `biba/low` label is considered the lowest integrity an object or subject may have. Setting this on objects or subjects will block their write access to objects or subjects marked high. They still have read access though.
- The `biba/equal` label should only be placed on objects considered to be exempt from the policy.
- The `biba/high` label will permit writing to objects set at a lower label, but not permit reading that object. It is recommended that this label be placed on objects that affect the integrity of the entire system.

Biba provides for:

- Hierarchical integrity level with a set of non hierarchical integrity categories;
- Fixed rules: no write up, no read down (opposite of MLS). A subject can have write access to objects on its own level or below, but not above. Similarly, a subject can have read access to objects on its own level or above, but not below;
- Integrity (preventing inappropriate modification of data);
- Integrity levels (instead of MLS sensitivity levels).

The following `sysctl` tunables can be used to manipulate the Biba policy.

- `security.mac.biba.enabled` may be used to enable/disable enforcement of the Biba policy on the target machine.
- `security.mac.biba.ptys_equal` may be used to disable the Biba policy on pty(4) devices.
- `security.mac.biba.revocation_enabled` will force the revocation of access to objects if the label is changed to dominate the subject.

To access the Biba policy setting on system objects, use the `setfmac` and `getfmac` commands:

```
# setfmac biba/low test
# getfmac test
test: biba/low
```

16.13.1 Planning Mandatory Integrity

Integrity, different from sensitivity, guarantees that the information will never be manipulated by untrusted parties. This includes information passed between subjects, objects, and both. It ensures that users will only be able to modify and in some cases even access information they explicitly need to.

The mac_biba(4) security policy module permits an administrator to address which files and programs a user or users may see and invoke while assuring that the programs and files are free from threats and trusted by the system for that user, or group of users.

During the initial planning phase, an administrator must be prepared to partition users into grades, levels, and areas. Users will be blocked access not only to data but programs and utilities both before and after they start. The system will default to a high label once this policy module is enabled, and it is up to the administrator to configure the different grades and levels for users. Instead of using clearance levels as described above, a good planning method could include topics. For instance, only allow developers modification access to the source code repository, source code compiler, and other development utilities. While other users would be grouped into other categories such as testers, designers, or just ordinary users and would only be permitted read access.

With its natural security control, a lower integrity subject is unable to write to a higher integrity subject; a higher integrity subject cannot observe or read a lower integrity object. Setting a label at the lowest possible grade could make it inaccessible to subjects. Some prospective environments for this security policy module would include a constrained web server, development and test machine, and source code repository. A less useful implementation would be a personal workstation, a machine used as a router, or a network firewall.

16.14 The MAC LOMAC Module

Module name: `mac_lomac.ko`

Kernel configuration line: `options MAC_LOMAC`

Boot option: `mac_lomac_load="YES"`

Unlike the MAC Biba policy, the mac_lomac(4) policy permits access to lower integrity objects only after decreasing the integrity level to not disrupt any integrity rules.

The MAC version of the Low-watermark integrity policy, not to be confused with the older lomac(4) implementation, works almost identically to Biba, but with the exception of using floating labels to support subject demotion via an auxiliary grade compartment. This secondary compartment takes the form of `[auxgrade]`. When

assigning a lomac policy with an auxiliary grade, it should look a little bit like: `lomac/10[2]` where the number two (2) is the auxiliary grade.

The MAC LOMAC policy relies on the ubiquitous labeling of all system objects with integrity labels, permitting subjects to read from low integrity objects and then downgrading the label on the subject to prevent future writes to high integrity objects. This is the `[auxgrade]` option discussed above, thus the policy may provide for greater compatibility and require less initial configuration than Biba.

16.14.1 Examples

Like the Biba and MLS policies; the `setfmac` and `setpmac` utilities may be used to place labels on system objects:

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes lomac/high[low]
```

Notice the auxiliary grade here is `low`, this is a feature provided only by the MAC LOMAC policy.

16.15 Nagios in a MAC Jail

The following demonstration will implement a secure environment using various MAC modules with properly configured policies. This is only a test and should not be considered the complete answer to everyone's security woes. Just implementing a policy and ignoring it never works and could be disastrous in a production environment.

Before beginning this process, the `multilabel` option must be set on each file system as stated at the beginning of this chapter. Not doing so will result in errors. While at it, ensure that the `net-mngt/nagios-plugins`, `net-mngt/nagios`, and `www/apache13` ports are all installed, configured, and working correctly.

16.15.1 Create an insecure User Class

Begin the procedure by adding the following user class to the `/etc/login.conf` file:

```
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):
```

And adding the following line to the default user class:

```
:label=biba/high:
```

Once this is completed, the following command must be issued to rebuild the database:

```
# cap_mkdb /etc/login.conf
```

16.15.2 Boot Configuration

Do not reboot yet, just add the following lines to `/boot/loader.conf` so the required modules will load during system initialization:

```
mac_biba_load="YES"
mac_seeotheruids_load="YES"
```

16.15.3 Configure Users

Set the `root` user to the default class using:

```
# pw usermod root -L default
```

All user accounts that are not `root` or system users will now require a login class. The login class is required otherwise users will be refused access to common commands such as vi(1). The following `sh` script should do the trick:

```
# for x in ‘awk -F: ’($3 >= 1001) && ($3 != 65534) { print $1 }’ \
 /etc/passwd‘; do pw usermod $x -L default; done;
```

Drop the `nagios` and `www` users into the insecure class:

```
# pw usermod nagios -L insecure

# pw usermod www -L insecure
```

16.15.4 Create the Contexts File

A contexts file should now be created; the following example file should be placed in `/etc/policy.contexts`.

```
# This is the default BIBA policy for this system.

# System:
/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

# For Nagios:
/usr/local/etc/nagios
/usr/local/etc/nagios/*         biba/10

/var/spool/nagios               biba/10
/var/spool/nagios/*             biba/10

# For apache
/usr/local/etc/apache           biba/10
/usr/local/etc/apache/*         biba/10
```

This policy will enforce security by setting restrictions on the flow of information. In this specific configuration, users, `root` and others, should never be allowed to access **Nagios**. Configuration files and processes that are a part of **Nagios** will be completely self contained or jailed.

This file may now be read into our system by issuing the following command:

```
# setfsmac -ef /etc/policy.contexts /
# setfsmac -ef /etc/policy.contexts /
```

> **Note:** The above file system layout may be different depending on environment; however, it must be run on every single file system.

The `/etc/mac.conf` file requires the following modifications in the main section:

```
default_labels file ?biba
default_labels ifnet ?biba
default_labels process ?biba
default_labels socket ?biba
```

16.15.5 Enable Networking

Add the following line to `/boot/loader.conf`:

```
security.mac.biba.trust_all_interfaces=1
```

And the following to the network card configuration stored in `rc.conf`. If the primary Internet configuration is done via DHCP, this may need to be configured manually after every system boot:

```
maclabel biba/equal
```

16.15.6 Testing the Configuration

Ensure that the web server and **Nagios** will not be started on system initialization, and reboot. Ensure the `root` user cannot access any of the files in the **Nagios** configuration directory. If `root` can issue an ls(1) command on `/var/spool/nagios`, then something is wrong. Otherwise a "permission denied" error should be returned.

If all seems well, **Nagios**, **Apache**, and **Sendmail** can now be started in a way fitting of the security policy. The following commands will make this happen:

```
# 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
```

Double check to ensure that everything is working properly. If not, check the log files or error messages. Use the sysctl(8) utility to disable the mac_biba(4) security policy module enforcement and try starting everything again, like normal.

> **Note:** The `root` user can change the security enforcement and edit the configuration files without fear. The following command will permit the degradation of the security policy to a lower grade for a newly spawned shell:
>
> ```
> # setpmac biba/10 csh
> ```
>
> To block this from happening, force the user into a range via login.conf(5). If setpmac(8) attempts to run a command outside of the compartment's range, an error will be returned and the command will not be executed. In this case, setting root to `biba/high(high-high)`.

16.16 User Lock Down

This example considers a relatively small, fewer than fifty users, storage system. Users would have login capabilities, and be permitted to not only store data but access resources as well.

For this scenario, the mac_bsdextended(4) mixed with mac_seeotheruids(4) could co-exist and block access not only to system objects but to hide user processes as well.

Begin by adding the following line to `/boot/loader.conf`:

```
mac_seeotheruids_load="YES"
```

The mac_bsdextended(4) security policy module may be activated through the use of the following rc.conf variable:

```
ugidfw_enable="YES"
```

Default rules stored in `/etc/rc.bsdextended` will be loaded at system initialization; however, the default entries may need modification. Since this machine is expected only to service users, everything may be left commented out except the last two. These will force the loading of user owned system objects by default.

Add the required users to this machine and reboot. For testing purposes, try logging in as a different user across two consoles. Run the `ps aux` command to see if processes of other users are visible. Try to run ls(1) on another users home directory, it should fail.

Do not try to test with the `root` user unless the specific `sysctl`s have been modified to block super user access.

> **Note:** When a new user is added, their mac_bsdextended(4) rule will not be in the ruleset list. To update the ruleset quickly, simply unload the security policy module and reload it again using the kldunload(8) and kldload(8) utilities.

16.17 Troubleshooting the MAC Framework

During the development stage, a few users reported problems with normal configuration. Some of these problems are listed below:

16.17.1 The `multilabel` option cannot be enabled on `/`

The `multilabel` flag does not stay enabled on my root (`/`) partition!

It seems that one out of every fifty users has this problem, indeed, we had this problem during our initial configuration. Further observation of this so called “bug” has lead me to believe that it is a result of either incorrect documentation or misinterpretation of the documentation. Regardless of why it happened, the following steps may be taken to resolve it:

1. Edit `/etc/fstab` and set the root partition at `ro` for read-only.
2. Reboot into single user mode.
3. Run `tunefs -l enable` on `/`.
4. Reboot the system into normal mode.
5. Run `mount -urw /` and change the `ro` back to `rw` in `/etc/fstab` and reboot the system again.
6. Double-check the output from the `mount` to ensure that `multilabel` has been properly set on the root file system.

16.17.2 Cannot start a X11 server after MAC

After establishing a secure environment with MAC, I am no longer able to start X!

This could be caused by the MAC `partition` policy or by a mislabeling in one of the MAC labeling policies. To debug, try the following:

1. Check the error message; if the user is in the `insecure` class, the `partition` policy may be the culprit. Try setting the user's class back to the `default` class and rebuild the database with the `cap_mkdb` command. If this does not alleviate the problem, go to step two.
2. Double-check the label policies. Ensure that the policies are set correctly for the user in question, the X11 application, and the `/dev` entries.
3. If neither of these resolve the problem, send the error message and a description of your environment to the TrustedBSD discussion lists located at the TrustedBSD (http://www.TrustedBSD.org) website or to the FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) mailing list.

16.17.3 Error: _secure_path(3) cannot stat `.login_conf`

When I attempt to switch from the `root` user to another user in the system, the error message `_secure_path: unable to state .login_conf` appears.

This message is usually shown when the user has a higher label setting then that of the user whom they are attempting to become. For instance a user on the system, `joe`, has a default label of `biba/low`. The `root` user, who has a label of `biba/high`, cannot view `joe`'s home directory. This will happen regardless if `root` has used the `su` command to become `joe`, or not. In this scenario, the Biba integrity model will not permit `root` to view objects set at a lower integrity level.

16.17.4 The `root` username is broken!

In normal or even single user mode, the `root` is not recognized. The `whoami` command returns 0 (zero) and `su` returns `who are you?`. What could be going on?

This can happen if a labeling policy has been disabled, either by a sysctl(8) or the policy module was unloaded. If the policy is being disabled or has been temporarily disabled, then the login capabilities database needs to be reconfigured with the `label` option being removed. Double check the `login.conf` file to ensure that all `label` options have been removed and rebuild the database with the `cap_mkdb` command.

This may also happen if a policy restricts access to the `master.passwd` file or database. Usually caused by an administrator altering the file under a label which conflicts with the general policy being used by the system. In these cases, the user information would be read by the system and access would be blocked as the file has inherited the new label. Disable the policy via a sysctl(8) and everything should return to normal.

Notes

1. Other conditions may produce different failures. For instance, the file may not be owned by the user attempting to relabel the object, the object may not exist or may be read only. A mandatory policy will not allow the process to relabel the file, maybe because of a property of the file, a property of the process, or a property of the proposed new label value. For example: a user running at low integrity tries to change the label of a high integrity file. Or perhaps a user running at low integrity tries to change the label of a low integrity file to a high integrity label.

Chapter 17
Security Event Auditing

17.1 Synopsis

The FreeBSD operating system includes support for fine-grained security event auditing. Event auditing allows the reliable, fine-grained, and configurable logging of a variety of security-relevant system events, including logins, configuration changes, and file and network access. These log records can be invaluable for live system monitoring, intrusion detection, and postmortem analysis. FreeBSD implements Sun's published BSM API and file format, and is interoperable with both Sun's Solaris and Apple®'s Mac OS X audit implementations.

This chapter focuses on the installation and configuration of Event Auditing. It explains audit policies, and provides an example audit configuration.

After reading this chapter, you will know:

- What Event Auditing is and how it works.
- How to configure Event Auditing on FreeBSD for users and processes.
- How to review the audit trail using the audit reduction and review tools.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Chapter 3).
- Be familiar with the basics of kernel configuration/compilation (Chapter 8).
- Have some familiarity with security and how it pertains to FreeBSD (Chapter 14).

> **Warning:** The audit facility has some known limitations which include that not all security-relevant system events are currently auditable, and that some login mechanisms, such as X11-based display managers and third party daemons, do not properly configure auditing for user login sessions.
>
> The security event auditing facility is able to generate very detailed logs of system activity: on a busy system, trail file data can be very large when configured for high detail, exceeding gigabytes a week in some configurations. Administrators should take into account disk space requirements associated with high volume audit configurations. For example, it may be desirable to dedicate a file system to the `/var/audit` tree so that other file systems are not affected if the audit file system becomes full.

17.2 Key Terms in this Chapter

Before reading this chapter, a few key audit-related terms must be explained:

- *event*: An auditable event is any event that can be logged using the audit subsystem. Examples of security-relevant events include the creation of a file, the building of a network connection, or a user logging in. Events are either "attributable", meaning that they can be traced to an authenticated user, or "non-attributable" if they cannot be. Examples of non-attributable events are any events that occur before authentication in the login process, such as bad password attempts.
- *class*: Event classes are named sets of related events, and are used in selection expressions. Commonly used classes of events include "file creation" (fc), "exec" (ex) and "login_logout" (lo).
- *record*: A record is an audit log entry describing a security event. Records contain a record event type, information on the subject (user) performing the action, date and time information, information on any objects or arguments, and a success or failure condition.
- *trail*: An audit trail, or log file, consists of a series of audit records describing security events. Typically, trails are in roughly chronological order with respect to the time events completed. Only authorized processes are allowed to commit records to the audit trail.
- *selection expression*: A selection expression is a string containing a list of prefixes and audit event class names used to match events.
- *preselection*: The process by which the system identifies which events are of interest to the administrator in order to avoid generating audit records describing events that are not of interest. The preselection configuration uses a series of selection expressions to identify which classes of events to audit for which users, as well as global settings that apply to both authenticated and unauthenticated processes.
- *reduction*: The process by which records from existing audit trails are selected for preservation, printing, or analysis. Likewise, the process by which undesired audit records are removed from the audit trail. Using reduction, administrators can implement policies for the preservation of audit data. For example, detailed audit trails might be kept for one month, but after that, trails might be reduced in order to preserve only login information for archival purposes.

17.3 Installing Audit Support

User space support for Event Auditing is installed as part of the base FreeBSD operating system. Kernel support for Event Auditing is compiled in by default, but support for this feature must be explicitly compiled into the custom kernel by adding the following line to the kernel configuration file:

```
options AUDIT
```

Rebuild and reinstall the kernel via the normal process explained in Chapter 8.

Once an audit-enabled kernel is built, installed, and the system has been rebooted, enable the audit daemon by adding the following line to rc.conf(5):

```
auditd_enable="YES"
```

Audit support must then be started by a reboot, or by manually starting the audit daemon:

```
/etc/rc.d/auditd start
```

17.4 Audit Configuration

All configuration files for security audit are found in `/etc/security`. The following files must be present before the audit daemon is started:

- `audit_class` - Contains the definitions of the audit classes.
- `audit_control` - Controls aspects of the audit subsystem, such as default audit classes, minimum disk space to leave on the audit log volume, maximum audit trail size, etc.
- `audit_event` - Textual names and descriptions of system audit events, as well as a list of which classes each event is in.
- `audit_user` - User-specific audit requirements, which are combined with the global defaults at login.
- `audit_warn` - A customizable shell script used by **auditd** to generate warning messages in exceptional situations, such as when space for audit records is running low or when the audit trail file has been rotated.

> **Warning:** Audit configuration files should be edited and maintained carefully, as errors in configuration may result in improper logging of events.

17.4.1 Event Selection Expressions

Selection expressions are used in a number of places in the audit configuration to determine which events should be audited. Expressions contain a list of event classes to match, each with a prefix indicating whether matching records should be accepted or ignored, and optionally to indicate if the entry is intended to match successful or failed operations. Selection expressions are evaluated from left to right, and two expressions are combined by appending one onto the other.

The following list contains the default audit event classes present in `audit_class`:

- `all` - *all* - Match all event classes.
- `ad` - *administrative* - Administrative actions performed on the system as a whole.
- `ap` - *application* - Application defined action.
- `cl` - *file close* - Audit calls to the `close` system call.
- `ex` - *exec* - Audit program execution. Auditing of command line arguments and environmental variables is controlled via audit_control(5) using the `argv` and `envv` parameters to the `policy` setting.
- `fa` - *file attribute access* - Audit the access of object attributes such as stat(1), pathconf(2) and similar events.
- `fc` - *file create* - Audit events where a file is created as a result.
- `fd` - *file delete* - Audit events where file deletion occurs.
- `fm` - *file attribute modify* - Audit events where file attribute modification occurs, such as chown(8), chflags(1), flock(2), etc.
- `fr` - *file read* - Audit events in which data is read, files are opened for reading, etc.
- `fw` - *file write* - Audit events in which data is written, files are written or modified, etc.
- `io` - *ioctl* - Audit use of the ioctl(2) system call.

- `ip` - *ipc* - Audit various forms of Inter-Process Communication, including POSIX pipes and System V IPC operations.
- `lo` - *login_logout* - Audit login(1) and logout(1) events occurring on the system.
- `na` - *non attributable* - Audit non-attributable events.
- `no` - *invalid class* - Match no audit events.
- `nt` - *network* - Audit events related to network actions, such as connect(2) and accept(2).
- `ot` - *other* - Audit miscellaneous events.
- `pc` - *process* - Audit process operations, such as exec(3) and exit(3).

These audit event classes may be customized by modifying the `audit_class` and `audit_event` configuration files.

Each audit class in the list is combined with a prefix indicating whether successful/failed operations are matched, and whether the entry is adding or removing matching for the class and type.

- (none) Audit both successful and failed instances of the event.
- `+` Audit successful events in this class.
- `-` Audit failed events in this class.
- `^` Audit neither successful nor failed events in this class.
- `^+` Do not audit successful events in this class.
- `^-` Do not audit failed events in this class.

The following example selection string selects both successful and failed login/logout events, but only successful execution events:

```
lo,+ex
```

17.4.2 Configuration Files

In most cases, administrators will need to modify only two files when configuring the audit system: `audit_control` and `audit_user`. The first controls system-wide audit properties and policies; the second may be used to fine-tune auditing by user.

17.4.2.1 The `audit_control` File

The `audit_control` file specifies a number of defaults for the audit subsystem. Viewing the contents of this file, we see the following:

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo
policy:cnt
filesz:0
```

The `dir` option is used to set one or more directories where audit logs will be stored. If more than one directory entry appears, they will be used in order as they fill. It is common to configure audit so that audit logs are stored on a

dedicated file system, in order to prevent interference between the audit subsystem and other subsystems if the file system fills.

The `flags` field sets the system-wide default preselection mask for attributable events. In the example above, successful and failed login and logout events are audited for all users.

The `minfree` option defines the minimum percentage of free space for the file system where the audit trail is stored. When this threshold is exceeded, a warning will be generated. The above example sets the minimum free space to twenty percent.

The `naflags` option specifies audit classes to be audited for non-attributed events, such as the login process and system daemons.

The `policy` option specifies a comma-separated list of policy flags controlling various aspects of audit behavior. The default `cnt` flag indicates that the system should continue running despite an auditing failure (this flag is highly recommended). Another commonly used flag is `argv`, which causes command line arguments to the execve(2) system call to be audited as part of command execution.

The `filesz` option specifies the maximum size in bytes to allow an audit trail file to grow to before automatically terminating and rotating the trail file. The default, 0, disables automatic log rotation. If the requested file size is non-zero and below the minimum 512k, it will be ignored and a log message will be generated.

17.4.2.2 The `audit_user` File

The `audit_user` file permits the administrator to specify further audit requirements for specific users. Each line configures auditing for a user via two fields: the first is the `alwaysaudit` field, which specifies a set of events that should always be audited for the user, and the second is the `neveraudit` field, which specifies a set of events that should never be audited for the user.

The following example `audit_user` file audits login/logout events and successful command execution for the `root` user, and audits file creation and successful command execution for the `www` user. If used with the example `audit_control` file above, the `lo` entry for `root` is redundant, and login/logout events will also be audited for the `www` user.

```
root:lo,+ex:no
www:fc,+ex:no
```

17.5 Administering the Audit Subsystem

17.5.1 Viewing Audit Trails

Audit trails are stored in the BSM binary format, so tools must be used to modify or convert to text. The praudit(1) command converts trail files to a simple text format; the auditreduce(1) command may be used to reduce the audit trail file for analysis, archiving, or printing purposes. `auditreduce` supports a variety of selection parameters, including event type, event class, user, date or time of the event, and the file path or object acted on.

For example, the `praudit` utility will dump the entire contents of a specified audit log in plain text:

```
# praudit /var/audit/AUDITFILE
```

Where *AUDITFILE* is the audit log to dump.

Audit trails consist of a series of audit records made up of tokens, which `praudit` prints sequentially one per line. Each token is of a specific type, such as `header` holding an audit record header, or `path` holding a file path from a name lookup. The following is an example of an `execve` event:

```
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
```

This audit represents a successful `execve` call, in which the command `finger doug` has been run. The arguments token contains both the processed command line presented by the shell to the kernel. The `path` token holds the path to the executable as looked up by the kernel. The `attribute` token describes the binary, and in particular, includes the file mode which can be used to determine if the application was setuid. The `subject` token describes the subject process, and stores in sequence the audit user ID, effective user ID and group ID, real user ID and group ID, process ID, session ID, port ID, and login address. Notice that the audit user ID and real user ID differ: the user `robert` has switched to the `root` account before running this command, but it is audited using the original authenticated user. Finally, the `return` token indicates the successful execution, and the `trailer` concludes the record.

`praudit` also supports an XML output format, which can be selected using the `-x` argument.

17.5.2 Reducing Audit Trails

Since audit logs may be very large, an administrator will likely want to select a subset of records for using, such as records associated with a specific user:

```
# auditreduce -u trhodes /var/audit/AUDITFILE | praudit
```

This will select all audit records produced for the user `trhodes` stored in the *AUDITFILE* file.

17.5.3 Delegating Audit Review Rights

Members of the `audit` group are given permission to read audit trails in `/var/audit`; by default, this group is empty, so only the `root` user may read audit trails. Users may be added to the `audit` group in order to delegate audit review rights to the user. As the ability to track audit log contents provides significant insight into the behavior of users and processes, it is recommended that the delegation of audit review rights be performed with caution.

17.5.4 Live Monitoring Using Audit Pipes

Audit pipes are cloning pseudo-devices in the device file system which allow applications to tap the live audit record stream. This is primarily of interest to authors of intrusion detection and system monitoring applications. However, for the administrator the audit pipe device is a convenient way to allow live monitoring without running into problems with audit trail file ownership or log rotation interrupting the event stream. To track the live audit event stream, use the following command line:

```
# praudit /dev/auditpipe
```

By default, audit pipe device nodes are accessible only to the `root` user. To make them accessible to the members of the `audit` group, add a `devfs` rule to `devfs.rules`:

```
add path 'auditpipe*' mode 0440 group audit
```

See devfs.rules(5) for more information on configuring the devfs file system.

> **Warning:** It is easy to produce audit event feedback cycles, in which the viewing of each audit event results in the generation of more audit events. For example, if all network I/O is audited, and praudit(1) is run from an SSH session, then a continuous stream of audit events will be generated at a high rate, as each event being printed will generate another event. It is advisable to run `praudit` on an audit pipe device from sessions without fine-grained I/O auditing in order to avoid this happening.

17.5.5 Rotating Audit Trail Files

Audit trails are written to only by the kernel, and managed only by the audit daemon, **auditd**. Administrators should not attempt to use newsyslog.conf(5) or other tools to directly rotate audit logs. Instead, the `audit` management tool may be used to shut down auditing, reconfigure the audit system, and perform log rotation. The following command causes the audit daemon to create a new audit log and signal the kernel to switch to using the new log. The old log will be terminated and renamed, at which point it may then be manipulated by the administrator.

```
# audit -n
```

> **Warning:** If the **auditd** daemon is not currently running, this command will fail and an error message will be produced.

Adding the following line to `/etc/crontab` will force the rotation every twelve hours from cron(8):

```
0     */12       *       *       *       root    /usr/sbin/audit -n
```

The change will take effect once you have saved the new `/etc/crontab`.

Automatic rotation of the audit trail file based on file size is possible via the `filesz` option in audit_control(5), and is described in the configuration files section of this chapter.

17.5.6 Compressing Audit Trails

As audit trail files can become very large, it is often desirable to compress or otherwise archive trails once they have been closed by the audit daemon. The `audit_warn` script can be used to perform customized operations for a variety of audit-related events, including the clean termination of audit trails when they are rotated. For example, the following may be added to the `audit_warn` script to compress audit trails on close:

```
#
# Compress audit trail files on close.
```

```
#
if [ "$1" = closefile ]; then
        gzip -9 $2
fi
```

Other archiving activities might include copying trail files to a centralized server, deleting old trail files, or reducing the audit trail to remove unneeded records. The script will be run only when audit trail files are cleanly terminated, so will not be run on trails left unterminated following an improper shutdown.

Chapter 18
Storage

18.1 Synopsis

This chapter covers the use of disks in FreeBSD. This includes memory-backed disks, network-attached disks, standard SCSI/IDE storage devices, and devices using the USB interface.

After reading this chapter, you will know:

- The terminology FreeBSD uses to describe the organization of data on a physical disk (partitions and slices).
- How to add additional hard disks to your system.
- How to configure FreeBSD to use USB storage devices.
- How to set up virtual file systems, such as memory disks.
- How to use quotas to limit disk space usage.
- How to encrypt disks to secure them against attackers.
- How to create and burn CDs and DVDs on FreeBSD.
- The various storage media options for backups.
- How to use backup programs available under FreeBSD.
- How to backup to floppy disks.
- What file system snapshots are and how to use them efficiently.

Before reading this chapter, you should:

- Know how to configure and install a new FreeBSD kernel (Chapter 8).

18.2 Device Names

The following is a list of physical storage devices supported in FreeBSD, and the device names associated with them.

Table 18-1. Physical Disk Naming Conventions

Drive type	Drive device name
IDE hard drives	`ad`
IDE CDROM drives	`acd`
SCSI hard drives and USB Mass storage devices	`da`
SCSI CDROM drives	`cd`

Drive type	Drive device name
Assorted non-standard CDROM drives	`mcd` for Mitsumi CD-ROM and `scd` for Sony CD-ROM devices
Floppy drives	`fd`
SCSI tape drives	`sa`
IDE tape drives	`ast`
Flash drives	`fla` for DiskOnChip® Flash device
RAID drives	`aacd` for Adaptec® AdvancedRAID, `mlxd` and `mlyd` for Mylex®, `amrd` for AMI MegaRAID®, `idad` for Compaq Smart RAID, `twed` for 3ware® RAID.

18.3 Adding Disks

The following section will describe how to add a new SCSI disk to a machine that currently only has a single drive. First turn off the computer and install the drive in the computer following the instructions of the computer, controller, and drive manufacturer. Due to the wide variations of procedures to do this, the details are beyond the scope of this document.

Login as user `root`. After you have installed the drive, inspect `/var/run/dmesg.boot` to ensure the new disk was found. Continuing with our example, the newly added drive will be `da1` and we want to mount it on `/1` (if you are adding an IDE drive, the device name will be `ad1`).

FreeBSD runs on IBM-PC compatible computers, therefore it must take into account the PC BIOS partitions. These are different from the traditional BSD partitions. A PC disk has up to four BIOS partition entries. If the disk is going to be truly dedicated to FreeBSD, you can use the *dedicated* mode. Otherwise, FreeBSD will have to live within one of the PC BIOS partitions. FreeBSD calls the PC BIOS partitions *slices* so as not to confuse them with traditional BSD partitions. You may also use slices on a disk that is dedicated to FreeBSD, but used in a computer that also has another operating system installed. This is a good way to avoid confusing the `fdisk` utility of other, non-FreeBSD operating systems.

In the slice case the drive will be added as `/dev/da1s1e`. This is read as: SCSI disk, unit number 1 (second SCSI disk), slice 1 (PC BIOS partition 1), and `e` BSD partition. In the dedicated case, the drive will be added simply as `/dev/da1e`.

Due to the use of 32-bit integers to store the number of sectors, bsdlabel(8) is limited to 2^32-1 sectors per disk or 2TB in most cases. The fdisk(8) format allows a starting sector of no more than 2^32-1 and a length of no more than 2^32-1, limiting partitions to 2TB and disks to 4TB in most cases. The sunlabel(8) format is limited to 2^32-1 sectors per partition and 8 partitions for a total of 16TB. For larger disks, gpt(8) partitions may be used.

18.3.1 Using sysinstall(8)

1. Navigating **Sysinstall**

   You may use `sysinstall` to partition and label a new disk using its easy to use menus. Either login as user `root` or use the `su` command. Run `sysinstall` and enter the `Configure` menu. Within the `FreeBSD Configuration Menu`, scroll down and select the `Fdisk` option.

2. **fdisk** Partition Editor

Once inside **fdisk**, pressing **A** will use the entire disk for FreeBSD. When asked if you want to "remain cooperative with any future possible operating systems", answer `YES`. Write the changes to the disk using **W**. Now exit the FDISK editor by pressing **Q**. Next you will be asked about the "Master Boot Record". Since you are adding a disk to an already running system, choose `None`.

3. Disk Label Editor

   Next, you need to exit **sysinstall** and start it again. Follow the directions above, although this time choose the `Label` option. This will enter the `Disk Label Editor`. This is where you will create the traditional BSD partitions. A disk can have up to eight partitions, labeled `a-h`. A few of the partition labels have special uses. The `a` partition is used for the root partition (`/`). Thus only your system disk (e.g, the disk you boot from) should have an `a` partition. The `b` partition is used for swap partitions, and you may have many disks with swap partitions. The `c` partition addresses the entire disk in dedicated mode, or the entire FreeBSD slice in slice mode. The other partitions are for general use.

   sysinstall's Label editor favors the `e` partition for non-root, non-swap partitions. Within the Label editor, create a single file system by pressing **C**. When prompted if this will be a FS (file system) or swap, choose `FS` and type in a mount point (e.g, `/mnt`). When adding a disk in post-install mode, **sysinstall** will not create entries in `/etc/fstab` for you, so the mount point you specify is not important.

   You are now ready to write the new label to the disk and create a file system on it. Do this by pressing **W**. Ignore any errors from **sysinstall** that it could not mount the new partition. Exit the Label Editor and **sysinstall** completely.

4. Finish

   The last step is to edit `/etc/fstab` to add an entry for your new disk.

18.3.2 Using Command Line Utilities

18.3.2.1 Using Slices

This setup will allow your disk to work correctly with other operating systems that might be installed on your computer and will not confuse other operating systems' `fdisk` utilities. It is recommended to use this method for new disk installs. Only use `dedicated` mode if you have a good reason to do so!

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# fdisk -BI da1 #Initialize your new disk
# bsdlabel -B -w da1s1 auto #Label it.
# bsdlabel -e da1s1 # Edit the bsdlabel just created and add any partitions.
# mkdir -p /1
# newfs /dev/da1s1e # Repeat this for every partition you created.
# mount /dev/da1s1e /1 # Mount the partition(s)
# vi /etc/fstab # Add the appropriate entry/entries to your /etc/fstab.
```

If you have an IDE disk, substitute `ad` for `da`.

18.3.2.2 Dedicated

If you will not be sharing the new drive with another operating system, you may use the `dedicated` mode. Remember this mode can confuse Microsoft operating systems; however, no damage will be done by them. IBM's OS/2 however, will "appropriate" any partition it finds which it does not understand.

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# bsdlabel -Bw da1 auto
# bsdlabel -e da1    # create the 'e' partition
# newfs /dev/da1e
# mkdir -p /1
# vi /etc/fstab    # add an entry for /dev/da1e
# mount /1
```

An alternate method is:

```
# dd if=/dev/zero of=/dev/da1 count=2
# bsdlabel /dev/da1 | bsdlabel -BR da1 /dev/stdin
# newfs /dev/da1e
# mkdir -p /1
# vi /etc/fstab     # add an entry for /dev/da1e
# mount /1
```

18.4 RAID

18.4.1 Software RAID

18.4.1.1 Concatenated Disk Driver (CCD) Configuration

When choosing a mass storage solution the most important factors to consider are speed, reliability, and cost. It is rare to have all three in balance; normally a fast, reliable mass storage device is expensive, and to cut back on cost either speed or reliability must be sacrificed.

In designing the system described below, cost was chosen as the most important factor, followed by speed, then reliability. Data transfer speed for this system is ultimately constrained by the network. And while reliability is very important, the CCD drive described below serves online data that is already fully backed up on CD-R's and can easily be replaced.

Defining your own requirements is the first step in choosing a mass storage solution. If your requirements prefer speed or reliability over cost, your solution will differ from the system described in this section.

18.4.1.1.1 Installing the Hardware

In addition to the IDE system disk, three Western Digital 30GB, 5400 RPM IDE disks form the core of the CCD disk described below providing approximately 90GB of online storage. Ideally, each IDE disk would have its own IDE controller and cable, but to minimize cost, additional IDE controllers were not used. Instead the disks were configured with jumpers so that each IDE controller has one master, and one slave.

Upon reboot, the system BIOS was configured to automatically detect the disks attached. More importantly, FreeBSD detected them on reboot:

```
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
```

> **Note:** If FreeBSD does not detect all the disks, ensure that you have jumpered them correctly. Most IDE drives also have a "Cable Select" jumper. This is *not* the jumper for the master/slave relationship. Consult the drive documentation for help in identifying the correct jumper.

Next, consider how to attach them as part of the file system. You should research both vinum(4) (Chapter 21) and ccd(4). In this particular configuration, ccd(4) was chosen.

18.4.1.1.2 Setting Up the CCD

The ccd(4) driver allows you to take several identical disks and concatenate them into one logical file system. In order to use ccd(4), you need a kernel with ccd(4) support built in. Add this line to your kernel configuration file, rebuild, and reinstall the kernel:

```
device   ccd
```

The ccd(4) support can also be loaded as a kernel loadable module.

To set up ccd(4), you must first use bsdlabel(8) to label the disks:

```
bsdlabel -w ad1 auto
bsdlabel -w ad2 auto
bsdlabel -w ad3 auto
```

This creates a bsdlabel for `ad1c`, `ad2c` and `ad3c` that spans the entire disk.

The next step is to change the disk label type. You can use bsdlabel(8) to edit the disks:

```
bsdlabel -e ad1
bsdlabel -e ad2
bsdlabel -e ad3
```

This opens up the current disk label on each disk with the editor specified by the `EDITOR` environment variable, typically vi(1).

An unmodified disk label will look something like this:

```
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  c: 60074784        0    unused        0     0     0   # (Cyl.    0 - 59597)
```

Add a new `e` partition for ccd(4) to use. This can usually be copied from the `c` partition, but the `fstype` *must* be **`4.2BSD`**. The disk label should now look something like this:

```
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 Building the File System

Now that you have all the disks labeled, you must build the ccd(4). To do that, use ccdconfig(8), with options similar to the following:

```
ccdconfig ccd0❶ 32❷ 0❸ /dev/ad1e❹ /dev/ad2e /dev/ad3e
```

The use and meaning of each option is shown below:

❶ The first argument is the device to configure, in this case, `/dev/ccd0c`. The `/dev/` portion is optional.

❷ The interleave for the file system. The interleave defines the size of a stripe in disk blocks, each normally 512 bytes. So, an interleave of 32 would be 16,384 bytes.

❸ Flags for ccdconfig(8). If you want to enable drive mirroring, you can specify a flag here. This configuration does not provide mirroring for ccd(4), so it is set at 0 (zero).

❹ The final arguments to ccdconfig(8) are the devices to place into the array. Use the complete pathname for each device.

After running ccdconfig(8) the ccd(4) is configured. A file system can be installed. Refer to newfs(8) for options, or simply run:

```
newfs /dev/ccd0c
```

18.4.1.1.4 Making it All Automatic

Generally, you will want to mount the ccd(4) upon each reboot. To do this, you must configure it first. Write out your current configuration to `/etc/ccd.conf` using the following command:

```
ccdconfig -g > /etc/ccd.conf
```

During reboot, the script `/etc/rc` runs `ccdconfig -C` if `/etc/ccd.conf` exists. This automatically configures the ccd(4) so it can be mounted.

> **Note:** If you are booting into single user mode, before you can mount(8) the ccd(4), you need to issue the following command to configure the array:
>
> ```
> ccdconfig -C
> ```

To automatically mount the ccd(4), place an entry for the ccd(4) in `/etc/fstab` so it will be mounted at boot time:

```
/dev/ccd0c              /media       ufs     rw      2       2
```

18.4.1.2 The Vinum Volume Manager

The Vinum Volume Manager is a block device driver which implements virtual disk drives. It isolates disk hardware from the block device interface and maps data in ways which result in an increase in flexibility, performance and reliability compared to the traditional slice view of disk storage. vinum(4) implements the RAID-0, RAID-1 and RAID-5 models, both individually and in combination.

See Chapter 21 for more information about vinum(4).

18.4.2 Hardware RAID

FreeBSD also supports a variety of hardware RAID controllers. These devices control a RAID subsystem without the need for FreeBSD specific software to manage the array.

Using an on-card BIOS, the card controls most of the disk operations itself. The following is a brief setup description using a Promise IDE RAID controller. When this card is installed and the system is started up, it displays a prompt requesting information. Follow the instructions to enter the card's setup screen. From here, you have the ability to combine all the attached drives. After doing so, the disk(s) will look like a single drive to FreeBSD. Other RAID levels can be set up accordingly.

18.4.3 Rebuilding ATA RAID1 Arrays

FreeBSD allows you to hot-replace a failed disk in an array. This requires that you catch it before you reboot.

You will probably see something like the following in `/var/log/messages` or in the dmesg(8) output:

```
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
```

Using atacontrol(8), check for further information:

```
# 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. You will first need to detach the ata channel with the failed disk so you can safely remove it:

   ```
   # atacontrol detach ata3
   ```

2. Replace the disk.

3. Reattach the ata channel:

   ```
   # atacontrol attach ata3
   Master:  ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
   Slave:   no device present
   ```

4. Add the new disk to the array as a spare:

   ```
   # atacontrol addspare ar0 ad6
   ```

5. Rebuild the array:

   ```
   # atacontrol rebuild ar0
   ```

6. It is possible to check on the progress by issuing the following command:

   ```
   # dmesg | tail -10
   [output removed]
   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. Wait until this operation completes.

18.5 USB Storage Devices

A lot of external storage solutions, nowadays, use the Universal Serial Bus (USB): hard drives, USB thumbdrives, CD-R burners, etc. FreeBSD provides support for these devices.

18.5.1 Configuration

The USB mass storage devices driver, umass(4), provides the support for USB storage devices. If you use the `GENERIC` kernel, you do not have to change anything in your configuration. If you use a custom kernel, be sure that the following lines are present in your kernel configuration file:

```
device scbus
device da
device pass
device uhci
device ohci
```

```
device ehci
device usb
device umass
```

The umass(4) driver uses the SCSI subsystem to access to the USB storage devices, your USB device will be seen as a SCSI device by the system. Depending on the USB chipset on your motherboard, you only need either `device uhci` or `device ohci` for USB 1.X support, however having both in the kernel configuration file is harmless. Support for USB 2.0 controllers is provided by the ehci(4) driver (the `device ehci` line). Do not forget to compile and install the new kernel if you added any lines.

> **Note:** If your USB device is a CD-R or DVD burner, the SCSI CD-ROM driver, cd(4), must be added to the kernel via the line:
>
> ```
> device cd
> ```
>
> Since the burner is seen as a SCSI drive, the driver atapicam(4) should not be used in the kernel configuration.

18.5.2 Testing the Configuration

The configuration is ready to be tested: plug in your USB device, and in the system message buffer (dmesg(8)), the drive should appear as something like:

```
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)
```

Of course, the brand, the device node (`da0`) and other details can differ according to your configuration.

Since the USB device is seen as a SCSI one, the `camcontrol` command can be used to list the USB storage devices attached to the system:

```
# camcontrol devlist
<Generic Traveling Disk 1.11>      at scbus0 target 0 lun 0 (da0,pass0)
```

If the drive comes with a file system, you should be able to mount it. The Section 18.3 will help you to format and create partitions on the USB drive if needed.

> **Warning:** Allowing untrusted users to mount arbitrary media, e.g. by enabling `vfs.usermount` as described below, should not be considered safe from a security point of view. Most file systems in FreeBSD were not built to safeguard against malicious devices.

To make this device mountable as a normal user, certain steps have to be taken. First, the devices that are created when a USB storage device is connected need to be accessible by the user. A solution is to make all users of these devices a member of the `operator` group. This is done with pw(8). Second, when the devices are created, the

`operator` group should be able to read and write them. This is accomplished by adding these lines to `/etc/devfs.rules`:

```
[localrules=5]
add path 'da*' mode 0660 group operator
```

> **Note:** If there already are SCSI disks in the system, it must be done a bit different. E.g., if the system already contains disks `da0` through `da2` attached to the system, change the second line as follows:
>
> ```
> add path 'da[3-9]*' mode 0660 group operator
> ```
>
> This will exclude the already existing disks from belonging to the `operator` group.

You also have to enable your devfs.rules(5) ruleset in your `/etc/rc.conf` file:

```
devfs_system_ruleset="localrules"
```

Next, the kernel has to be configured to allow regular users to mount file systems. The easiest way is to add the following line to `/etc/sysctl.conf`:

```
vfs.usermount=1
```

Note that this only takes effect after the next reboot. Alternatively, one can also use sysctl(8) to set this variable.

The final step is to create a directory where the file system is to be mounted. This directory needs to be owned by the user that is to mount the file system. One way to do that is for `root` to create a subdirectory owned by that user as `/mnt/`*`username`* (replace *`username`* by the login name of the actual user and *`usergroup`* by the user's primary group):

```
# mkdir /mnt/username
# chown username:usergroup /mnt/username
```

Suppose a USB thumbdrive is plugged in, and a device `/dev/da0s1` appears. Since these devices usually come preformatted with a FAT file system, one can mount them like this:

```
% mount -t msdosfs -o -m=644,-M=755 /dev/da0s1 /mnt/username
```

If you unplug the device (the disk must be unmounted before), you should see, in the system message buffer, something like the following:

```
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 Further Reading

Beside the Adding Disks and Mounting and Unmounting File Systems sections, reading various manual pages may be also useful: umass(4), camcontrol(8), and usbconfig(8) under FreeBSD 8.X or usbdevs(8) under earlier versions of FreeBSD.

18.6 Creating and Using Optical Media (CDs)

18.6.1 Introduction

CDs have a number of features that differentiate them from conventional disks. Initially, they were not writable by the user. They are designed so that they can be read continuously without delays to move the head between tracks. They are also much easier to transport between systems than similarly sized media were at the time.

CDs do have tracks, but this refers to a section of data to be read continuously and not a physical property of the disk. To produce a CD on FreeBSD, you prepare the data files that are going to make up the tracks on the CD, then write the tracks to the CD.

The ISO 9660 file system was designed to deal with these differences. It unfortunately codifies file system limits that were common then. Fortunately, it provides an extension mechanism that allows properly written CDs to exceed those limits while still working with systems that do not support those extensions.

The `sysutils/cdrtools` port includes mkisofs(8), a program that you can use to produce a data file containing an ISO 9660 file system. It has options that support various extensions, and is described below.

Which tool to use to burn the CD depends on whether your CD burner is ATAPI or something else. ATAPI CD burners use the `burncd` program that is part of the base system. SCSI and USB CD burners should use `cdrecord` from the `sysutils/cdrtools` port. It is also possible to use `cdrecord` and other tools for SCSI drives on ATAPI hardware with the ATAPI/CAM module.

If you want CD burning software with a graphical user interface, you may wish to take a look at either **X-CD-Roast** or **K3b**. These tools are available as packages or from the `sysutils/xcdroast` and `sysutils/k3b` ports. **X-CD-Roast** and **K3b** require the ATAPI/CAM module with ATAPI hardware.

18.6.2 mkisofs

The mkisofs(8) program, which is part of the `sysutils/cdrtools` port, produces an ISO 9660 file system that is an image of a directory tree in the UNIX file system name space. The simplest usage is:

```
# mkisofs -o imagefile.iso /path/to/tree
```

This command will create an *`imagefile.iso`* containing an ISO 9660 file system that is a copy of the tree at */path/to/tree*. In the process, it will map the file names to names that fit the limitations of the standard ISO 9660 file system, and will exclude files that have names uncharacteristic of ISO file systems.

A number of options are available to overcome those restrictions. In particular, `-R` enables the Rock Ridge extensions common to UNIX systems, `-J` enables Joliet extensions used by Microsoft systems, and `-hfs` can be used to create HFS file systems used by Mac OS.

For CDs that are going to be used only on FreeBSD systems, `-U` can be used to disable all filename restrictions. When used with `-R`, it produces a file system image that is identical to the FreeBSD tree you started from, though it may violate the ISO 9660 standard in a number of ways.

The last option of general use is `-b`. This is used to specify the location of the boot image for use in producing an "El Torito" bootable CD. This option takes an argument which is the path to a boot image from the top of the tree being written to the CD. By default, mkisofs(8) creates an ISO image in the so-called "floppy disk emulation" mode, and thus expects the boot image to be exactly 1200, 1440 or 2880 KB in size. Some boot loaders, like the one used by the FreeBSD distribution disks, do not use emulation mode; in this case, the `-no-emul-boot` option should be used. So, if `/tmp/myboot` holds a bootable FreeBSD system with the boot image in `/tmp/myboot/boot/cdboot`, you could produce the image of an ISO 9660 file system in `/tmp/bootable.iso` like so:

```
# mkisofs -R -no-emul-boot -b boot/cdboot -o /tmp/bootable.iso /tmp/myboot
```

Having done that, if you have `md` configured in your kernel, you can mount the file system with:

```
# mdconfig -a -t vnode -f /tmp/bootable.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

At which point you can verify that `/mnt` and `/tmp/myboot` are identical.

There are many other options you can use with mkisofs(8) to fine-tune its behavior. In particular: modifications to an ISO 9660 layout and the creation of Joliet and HFS discs. See the mkisofs(8) manual page for details.

18.6.3 burncd

If you have an ATAPI CD burner, you can use the `burncd` command to burn an ISO image onto a CD. `burncd` is part of the base system, installed as `/usr/sbin/burncd`. Usage is very simple, as it has few options:

```
# burncd -f cddevice data imagefile.iso fixate
```

Will burn a copy of *`imagefile.iso`* on *`cddevice`*. The default device is `/dev/acd0`. See burncd(8) for options to set the write speed, eject the CD after burning, and write audio data.

18.6.4 cdrecord

If you do not have an ATAPI CD burner, you will have to use `cdrecord` to burn your CDs. `cdrecord` is not part of the base system; you must install it from either the port at `sysutils/cdrtools` or the appropriate package. Changes to the base system can cause binary versions of this program to fail, possibly resulting in a "coaster". You should therefore either upgrade the port when you upgrade your system, or if you are tracking -STABLE, upgrade the port when a new version becomes available.

While `cdrecord` has many options, basic usage is even simpler than `burncd`. Burning an ISO 9660 image is done with:

```
# cdrecord dev=device imagefile.iso
```

The tricky part of using `cdrecord` is finding the `dev` to use. To find the proper setting, use the `-scanbus` flag of `cdrecord`, which might produce results like this:

```
# 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) *
```

This lists the appropriate `dev` value for the devices on the list. Locate your CD burner, and use the three numbers separated by commas as the value for `dev`. In this case, the CRW device is 1,5,0, so the appropriate input would be `dev=1,5,0`. There are easier ways to specify this value; see cdrecord(1) for details. That is also the place to look for information on writing audio tracks, controlling the speed, and other things.

18.6.5 Duplicating Audio CDs

You can duplicate an audio CD by extracting the audio data from the CD to a series of files, and then writing these files to a blank CD. The process is slightly different for ATAPI and SCSI drives.

SCSI Drives

1. Use `cdda2wav` to extract the audio.

   ```
   % cdda2wav -vall -D2,0 -B -Owav
   ```

2. Use `cdrecord` to write the `.wav` files.

   ```
   % cdrecord -v dev=2,0 -dao -useinfo  *.wav
   ```

   Make sure that *2,0* is set appropriately, as described in Section 18.6.4.

ATAPI Drives

> **Note:** With the help of the ATAPI/CAM module, `cdda2wav` can also be used on ATAPI drives. This tool is usually a better choice for most of users (jitter correction, endianness issues, etc.) than the method proposed below.

1. The ATAPI CD driver makes each track available as `/dev/acd`*`d`*`t`*`nn`*, where *d* is the drive number, and *nn* is the track number written with two decimal digits, prefixed with zero as needed. So the first track on the first disk is `/dev/acd0t01`, the second is `/dev/acd0t02`, the third is `/dev/acd0t03`, and so on.

   Make sure the appropriate files exist in `/dev`. If the entries are missing, force the system to retaste the media:

   ```
   # dd if=/dev/acd0 of=/dev/null count=1
   ```

2. Extract each track using dd(1). You must also use a specific block size when extracting the files.

   ```
   # dd if=/dev/acd0t01 of=track1.cdr bs=2352
   # dd if=/dev/acd0t02 of=track2.cdr bs=2352
   ...
   ```

3. Burn the extracted files to disk using `burncd`. You must specify that these are audio files, and that `burncd` should fixate the disk when finished.

   ```
   # burncd -f /dev/acd0 audio track1.cdr track2.cdr ... fixate
   ```

18.6.6 Duplicating Data CDs

You can copy a data CD to a image file that is functionally equivalent to the image file created with mkisofs(8), and you can use it to duplicate any data CD. The example given here assumes that your CDROM device is `acd0`. Substitute your correct CDROM device.

```
# dd if=/dev/acd0 of=file.iso bs=2048
```

Now that you have an image, you can burn it to CD as described above.

18.6.7 Using Data CDs

Now that you have created a standard data CDROM, you probably want to mount it and read the data on it. By default, mount(8) assumes that a file system is of type `ufs`. If you try something like:

```
# mount /dev/cd0 /mnt
```

you will get a complaint about `Incorrect super block`, and no mount. The CDROM is not a `UFS` file system, so attempts to mount it as such will fail. You just need to tell mount(8) that the file system is of type `ISO9660`, and everything will work. You do this by specifying the `-t cd9660` option mount(8). For example, if you want to mount the CDROM device, `/dev/cd0`, under `/mnt`, you would execute:

```
# mount -t cd9660 /dev/cd0 /mnt
```

Note that your device name (`/dev/cd0` in this example) could be different, depending on the interface your CDROM uses. Also, the `-t cd9660` option just executes mount_cd9660(8). The above example could be shortened to:

```
# mount_cd9660 /dev/cd0 /mnt
```

You can generally use data CDROMs from any vendor in this way. Disks with certain ISO 9660 extensions might behave oddly, however. For example, Joliet disks store all filenames in two-byte Unicode characters. The FreeBSD kernel does not speak Unicode, but the FreeBSD CD9660 driver is able to convert Unicode characters on the fly. If some non-English characters show up as question marks you will need to specify the local charset you use with the `-C` option. For more information, consult the mount_cd9660(8) manual page.

Note: To be able to do this character conversion with the help of the `-C` option, the kernel will require the `cd9660_iconv.ko` module to be loaded. This can be done either by adding this line to `loader.conf`:

```
cd9660_iconv_load="YES"
```

and then rebooting the machine, or by directly loading the module with kldload(8).

Occasionally, you might get `Device not configured` when trying to mount a CDROM. This usually means that the CDROM drive thinks that there is no disk in the tray, or that the drive is not visible on the bus. It can take a couple of seconds for a CDROM drive to realize that it has been fed, so be patient.

Sometimes, a SCSI CDROM may be missed because it did not have enough time to answer the bus reset. If you have a SCSI CDROM please add the following option to your kernel configuration and rebuild your kernel.

```
options SCSI_DELAY=15000
```

This tells your SCSI bus to pause 15 seconds during boot, to give your CDROM drive every possible chance to answer the bus reset.

18.6.8 Burning Raw Data CDs

You can choose to burn a file directly to CD, without creating an ISO 9660 file system. Some people do this for backup purposes. This runs more quickly than burning a standard CD:

```
# burncd -f /dev/acd1 -s 12 data archive.tar.gz fixate
```

In order to retrieve the data burned to such a CD, you must read data from the raw device node:

```
# tar xzvf /dev/acd1
```

You cannot mount this disk as you would a normal CDROM. Such a CDROM cannot be read under any operating system except FreeBSD. If you want to be able to mount the CD, or share data with another operating system, you must use mkisofs(8) as described above.

18.6.9 Using the ATAPI/CAM Driver

This driver allows ATAPI devices (CD-ROM, CD-RW, DVD drives etc...) to be accessed through the SCSI subsystem, and so allows the use of applications like `sysutils/cdrdao` or cdrecord(1).

To use this driver, you will need to add the following line to the `/boot/loader.conf` file:

```
atapicam_load="YES"
```

then, reboot your machine.

Note: If you prefer to statically compile the atapicam(4) support in your kernel, you will have to add this line to your kernel configuration file:

```
device atapicam
```

You also need the following lines in your kernel configuration file:

```
device ata
device scbus
device cd
device pass
```

which should already be present. Then rebuild, install your new kernel, and reboot your machine.

During the boot process, your burner should show up, like so:

```
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
```

The drive could now be accessed via the `/dev/cd0` device name, for example to mount a CD-ROM on `/mnt`, just type the following:

```
# mount -t cd9660 /dev/cd0 /mnt
```

As `root`, you can run the following command to get the SCSI address of the burner:

```
# camcontrol devlist
<MATSHITA CDRW/DVD UJDA740 1.00>   at scbus1 target 0 lun 0 (pass0,cd0)
```

So `1,0,0` will be the SCSI address to use with cdrecord(1) and other SCSI application.

For more information about ATAPI/CAM and SCSI system, refer to the atapicam(4) and cam(4) manual pages.

18.7 Creating and Using Optical Media (DVDs)

18.7.1 Introduction

Compared to the CD, the DVD is the next generation of optical media storage technology. The DVD can hold more data than any CD and is nowadays the standard for video publishing.

Five physical recordable formats can be defined for what we will call a recordable DVD:

- DVD-R: This was the first DVD recordable format available. The DVD-R standard is defined by the DVD Forum (http://www.dvdforum.com/forum.shtml). This format is write once.
- DVD-RW: This is the rewritable version of the DVD-R standard. A DVD-RW can be rewritten about 1000 times.
- DVD-RAM: This is also a rewritable format supported by the DVD Forum. A DVD-RAM can be seen as a removable hard drive. However, this media is not compatible with most DVD-ROM drives and DVD-Video players; only a few DVD writers support the DVD-RAM format. Read the Section 18.7.9 for more information on DVD-RAM use.
- DVD+RW: This is a rewritable format defined by the DVD+RW Alliance (http://www.dvdrw.com/). A DVD+RW can be rewritten about 1000 times.

- DVD+R: This format is the write once variation of the DVD+RW format.

A single layer recordable DVD can hold up to 4,700,000,000 bytes which is actually 4.38 GB or 4485 MB (1 kilobyte is 1024 bytes).

> **Note:** A distinction must be made between the physical media and the application. For example, a DVD-Video is a specific file layout that can be written on any recordable DVD physical media: DVD-R, DVD+R, DVD-RW etc. Before choosing the type of media, you must be sure that both the burner and the DVD-Video player (a standalone player or a DVD-ROM drive on a computer) are compatible with the media under consideration.

18.7.2 Configuration

The program growisofs(1) will be used to perform DVD recording. This command is part of the **dvd+rw-tools** utilities (`sysutils/dvd+rw-tools`). The **dvd+rw-tools** support all DVD media types.

These tools use the SCSI subsystem to access to the devices, therefore the ATAPI/CAM support must be added to your kernel. If your burner uses the USB interface this addition is useless, and you should read the Section 18.5 for more details on USB devices configuration.

You also have to enable DMA access for ATAPI devices, this can be done in adding the following line to the `/boot/loader.conf` file:

```
hw.ata.atapi_dma="1"
```

Before attempting to use the **dvd+rw-tools** you should consult the dvd+rw-tools' hardware compatibility notes (http://fy.chalmers.se/~appro/linux/DVD+RW/hcn.html) for any information related to your DVD burner.

> **Note:** If you want a graphical user interface, you should have a look to **K3b** (`sysutils/k3b`) which provides a user friendly interface to growisofs(1) and many other burning tools.

18.7.3 Burning Data DVDs

The growisofs(1) command is a frontend to mkisofs, it will invoke mkisofs(8) to create the file system layout and will perform the write on the DVD. This means you do not need to create an image of the data before the burning process.

To burn onto a DVD+R or a DVD-R the data from the `/path/to/data` directory, use the following command:

```
# growisofs -dvd-compat -Z /dev/cd0 -J -R /path/to/data
```

The options `-J -R` are passed to mkisofs(8) for the file system creation (in this case: an ISO 9660 file system with Joliet and Rock Ridge extensions), consult the mkisofs(8) manual page for more details.

The option `-Z` is used for the initial session recording in any case: multiple sessions or not. The DVD device, */dev/cd0*, must be changed according to your configuration. The `-dvd-compat` parameter will close the disk, the recording will be unappendable. In return this should provide better media compatibility with DVD-ROM drives.

It is also possible to burn a pre-mastered image, for example to burn the image *imagefile.iso*, we will run:

```
# growisofs -dvd-compat -Z /dev/cd0=imagefile.iso
```

The write speed should be detected and automatically set according to the media and the drive being used. If you want to force the write speed, use the `-speed=` parameter. For more information, read the growisofs(1) manual page.

> **Note:** In order to have working files larger than 4.38GB in your compilation, an UDF/ISO-9660 hybrid filesystem must be created by passing additional `-udf -iso-level 3` parameter to mkisofs(8) and all related programs (i.e., growisofs(1)). This is required only when creating an ISO image file, or writing files directly to a disk. Disk created this way must be mounted as an UDF filesystem with mount_udf(8) utility, so it will be usable only on an UDF aware Operating System,otherwise it will look as if it contains corrupted files.
>
> To create a such ISO file:
>
> ```
> % mkisofs -R -J -udf -iso-level 3 -o imagefile.iso /path/to/data
> ```
>
> To burn files directly to a disk:
>
> ```
> # growisofs -dvd-compat -udf -iso-level 3 -Z /dev/cd0 -J -R /path/to/data
> ```
>
> When you have an ISO image containing large files already inside, no additional options are required for growisofs(1) to burn that image on a disk.
>
> Also, be sure that you have an up-to-date version of `sysutils/cdrtools` (which contain mkisofs(8)), as the older ones does not contain large files support. If you experience troubles please move to the development package, i.e., `sysutils/cdrtools-devel` and read mkisofs(8) manual page.

18.7.4 Burning a DVD-Video

A DVD-Video is a specific file layout based on ISO 9660 and the micro-UDF (M-UDF) specifications. The DVD-Video also presents a specific data structure hierarchy, it is the reason why you need a particular program such as `multimedia/dvdauthor` to author the DVD.

If you already have an image of the DVD-Video file system, just burn it in the same way as for any image, see the previous section for an example. If you have made the DVD authoring and the result is in, for example, the directory `/path/to/video`, the following command should be used to burn the DVD-Video:

```
# growisofs -Z /dev/cd0 -dvd-video /path/to/video
```

The `-dvd-video` option will be passed down to mkisofs(8) and will instruct it to create a DVD-Video file system layout. Beside this, the `-dvd-video` option implies `-dvd-compat` growisofs(1) option.

18.7.5 Using a DVD+RW

Unlike CD-RW, a virgin DVD+RW needs to be formatted before first use. The growisofs(1) program will take care of it automatically whenever appropriate, which is the *recommended* way. However you can use the `dvd+rw-format` command to format the DVD+RW:

```
# dvd+rw-format /dev/cd0
```

You need to perform this operation just once, keep in mind that only virgin DVD+RW medias need to be formatted. Then you can burn the DVD+RW in the way seen in previous sections.

If you want to burn new data (burn a totally new file system not append some data) onto a DVD+RW, you do not need to blank it, you just have to write over the previous recording (in performing a new initial session), like this:

```
# growisofs -Z /dev/cd0 -J -R /path/to/newdata
```

DVD+RW format offers the possibility to easily append data to a previous recording. The operation consists in merging a new session to the existing one, it is not multisession writing, growisofs(1) will *grow* the ISO 9660 file system present on the media.

For example, if we want to append data to our previous DVD+RW, we have to use the following:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

The same mkisofs(8) options we used to burn the initial session should be used during next writes.

> **Note:** You may want to use the `-dvd-compat` option if you want better media compatibility with DVD-ROM drives. In the DVD+RW case, this will not prevent you from adding data.

If for any reason you really want to blank the media, do the following:

```
# growisofs -Z /dev/cd0=/dev/zero
```

18.7.6 Using a DVD-RW

A DVD-RW accepts two disc formats: the incremental sequential one and the restricted overwrite. By default DVD-RW discs are in sequential format.

A virgin DVD-RW can be directly written without the need of a formatting operation, however a non-virgin DVD-RW in sequential format needs to be blanked before to be able to write a new initial session.

To blank a DVD-RW in sequential mode, run:

```
# dvd+rw-format -blank=full /dev/cd0
```

> **Note:** A full blanking (`-blank=full`) will take about one hour on a 1x media. A fast blanking can be performed using the `-blank` option if the DVD-RW will be recorded in Disk-At-Once (DAO) mode. To burn the DVD-RW in DAO mode, use the command:
>
> ```
> # growisofs -use-the-force-luke=dao -Z /dev/cd0=imagefile.iso
> ```
>
> The `-use-the-force-luke=dao` option should not be required since growisofs(1) attempts to detect minimally (fast blanked) media and engage DAO write.
>
> In fact one should use restricted overwrite mode with any DVD-RW, this format is more flexible than the default incremental sequential one.

To write data on a sequential DVD-RW, use the same instructions as for the other DVD formats:

```
# growisofs -Z /dev/cd0 -J -R /path/to/data
```

If you want to append some data to your previous recording, you will have to use the growisofs(1) `-M` option. However, if you perform data addition on a DVD-RW in incremental sequential mode, a new session will be created on the disc and the result will be a multi-session disc.

A DVD-RW in restricted overwrite format does not need to be blanked before a new initial session, you just have to overwrite the disc with the `-Z` option, this is similar to the DVD+RW case. It is also possible to grow an existing ISO 9660 file system written on the disc in a same way as for a DVD+RW with the `-M` option. The result will be a one-session DVD.

To put a DVD-RW in the restricted overwrite format, the following command must be used:

```
# dvd+rw-format /dev/cd0
```

To change back to the sequential format use:

```
# dvd+rw-format -blank=full /dev/cd0
```

18.7.7 Multisession

Very few DVD-ROM drives support multisession DVDs, they will most of time, hopefully, only read the first session. DVD+R, DVD-R and DVD-RW in sequential format can accept multiple sessions, the notion of multiple sessions does not exist for the DVD+RW and the DVD-RW restricted overwrite formats.

Using the following command after an initial (non-closed) session on a DVD+R, DVD-R, or DVD-RW in sequential format, will add a new session to the disc:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

Using this command line with a DVD+RW or a DVD-RW in restricted overwrite mode, will append data in merging the new session to the existing one. The result will be a single-session disc. This is the way used to add data after an initial write on these medias.

> **Note:** Some space on the media is used between each session for end and start of sessions. Therefore, one should add sessions with large amount of data to optimize media space. The number of sessions is limited to 154 for a DVD+R, about 2000 for a DVD-R, and 127 for a DVD+R Double Layer.

18.7.8 For More Information

To obtain more information about a DVD, the `dvd+rw-mediainfo /dev/cd0` command can be ran with the disc in the drive.

More information about the **dvd+rw-tools** can be found in the growisofs(1) manual page, on the dvd+rw-tools web site (http://fy.chalmers.se/~appro/linux/DVD+RW/) and in the cdwrite mailing list (http://lists.debian.org/cdwrite/) archives.

> **Note:** The `dvd+rw-mediainfo` output of the resulting recording or the media with issues is mandatory for any problem report. Without this output, it will be quite impossible to help you.

18.7.9 Using a DVD-RAM

18.7.9.1 Configuration

DVD-RAM writers come with either SCSI or ATAPI interface. DMA access for ATAPI devices has to be enabled, this can be done by adding the following line to the `/boot/loader.conf` file:

```
hw.ata.atapi_dma="1"
```

18.7.9.2 Preparing the Medium

As previously mentioned in the chapter introduction, a DVD-RAM can be seen as a removable hard drive. As any other hard drive the DVD-RAM must be "prepared" before the first use. In the example, the whole disk space will be used with a standard UFS2 file system:

```
# dd if=/dev/zero of=/dev/acd0 bs=2k count=1
# bsdlabel -Bw acd0
# newfs /dev/acd0
```

The DVD device, `acd0`, must be changed according to the configuration.

18.7.9.3 Using the Medium

Once the previous operations have been performed on the DVD-RAM, it can be mounted as a normal hard drive:

```
# mount /dev/acd0 /mnt
```

After this the DVD-RAM will be both readable and writeable.

18.8 Creating and Using Floppy Disks

Storing data on floppy disks is sometimes useful, for example when one does not have any other removable storage media or when one needs to transfer small amounts of data to another computer.

This section will explain how to use floppy disks in FreeBSD. It will primarily cover formatting and usage of 3.5inch DOS floppies, but the concepts are similar for other floppy disk formats.

18.8.1 Formatting Floppies

18.8.1.1 The Device

Floppy disks are accessed through entries in `/dev`, just like other devices. To access the raw floppy disk, simply use `/dev/fd`*`N`*.

18.8.1.2 Formatting

A floppy disk needs to be low-level formated before it can be used. This is usually done by the vendor, but formatting is a good way to check media integrity. Although it is possible to force larger (or smaller) disk sizes, 1440kB is what most floppy disks are designed for.

To low-level format the floppy disk you need to use fdformat(1). This utility expects the device name as an argument.

Make note of any error messages, as these can help determine if the disk is good or bad.

18.8.1.2.1 Formatting Floppy Disks

Use the `/dev/fdN` devices to format the floppy. Insert a new 3.5inch floppy disk in your drive and issue:

```
# /usr/sbin/fdformat -f 1440 /dev/fd0
```

18.8.2 The Disk Label

After low-level formatting the disk, you will need to place a disk label on it. This disk label will be destroyed later, but it is needed by the system to determine the size of the disk and its geometry later.

The new disk label will take over the whole disk, and will contain all the proper information about the geometry of the floppy. The geometry values for the disk label are listed in `/etc/disktab`.

You can run now bsdlabel(8) like so:

```
# /sbin/bsdlabel -B -w /dev/fd0 fd1440
```

18.8.3 The File System

Now the floppy is ready to be high-level formated. This will place a new file system on it, which will let FreeBSD read and write to the disk. After creating the new file system, the disk label is destroyed, so if you want to reformat the disk, you will have to recreate the disk label.

The floppy's file system can be either UFS or FAT. FAT is generally a better choice for floppies.

To put a new file system on the floppy, issue:

```
# /sbin/newfs_msdos /dev/fd0
```

The disk is now ready for use.

18.8.4 Using the Floppy

To use the floppy, mount it with mount_msdosfs(8). One can also use `emulators/mtools` from the ports collection.

18.9 Creating and Using Data Tapes

The major tape media are the 4mm, 8mm, QIC, mini-cartridge and DLT.

18.9.1 4mm (DDS: Digital Data Storage)

4mm tapes are replacing QIC as the workstation backup media of choice. This trend accelerated greatly when Conner purchased Archive, a leading manufacturer of QIC drives, and then stopped production of QIC drives. 4mm drives are small and quiet but do not have the reputation for reliability that is enjoyed by 8mm drives. The cartridges are less expensive and smaller (3 x 2 x 0.5 inches, 76 x 51 x 12 mm) than 8mm cartridges. 4mm, like 8mm, has comparatively short head life for the same reason, both use helical scan.

Data throughput on these drives starts ~150 kB/s, peaking at ~500 kB/s. Data capacity starts at 1.3 GB and ends at 2.0 GB. Hardware compression, available with most of these drives, approximately doubles the capacity. Multi-drive tape library units can have 6 drives in a single cabinet with automatic tape changing. Library capacities reach 240 GB.

The DDS-3 standard now supports tape capacities up to 12 GB (or 24 GB compressed).

4mm drives, like 8mm drives, use helical-scan. All the benefits and drawbacks of helical-scan apply to both 4mm and 8mm drives.

Tapes should be retired from use after 2,000 passes or 100 full backups.

18.9.2 8mm (Exabyte)

8mm tapes are the most common SCSI tape drives; they are the best choice of exchanging tapes. Nearly every site has an Exabyte 2 GB 8mm tape drive. 8mm drives are reliable, convenient and quiet. Cartridges are inexpensive and small (4.8 x 3.3 x 0.6 inches; 122 x 84 x 15 mm). One downside of 8mm tape is relatively short head and tape life due to the high rate of relative motion of the tape across the heads.

Data throughput ranges from ~250 kB/s to ~500 kB/s. Data sizes start at 300 MB and go up to 7 GB. Hardware compression, available with most of these drives, approximately doubles the capacity. These drives are available as single units or multi-drive tape libraries with 6 drives and 120 tapes in a single cabinet. Tapes are changed automatically by the unit. Library capacities reach 840+ GB.

The Exabyte “Mammoth” model supports 12 GB on one tape (24 GB with compression) and costs approximately twice as much as conventional tape drives.

Data is recorded onto the tape using helical-scan, the heads are positioned at an angle to the media (approximately 6 degrees). The tape wraps around 270 degrees of the spool that holds the heads. The spool spins while the tape slides over the spool. The result is a high density of data and closely packed tracks that angle across the tape from one edge to the other.

18.9.3 QIC

QIC-150 tapes and drives are, perhaps, the most common tape drive and media around. QIC tape drives are the least expensive “serious” backup drives. The downside is the cost of media. QIC tapes are expensive compared to 8mm or 4mm tapes, up to 5 times the price per GB data storage. But, if your needs can be satisfied with a half-dozen tapes, QIC may be the correct choice. QIC is the *most* common tape drive. Every site has a QIC drive of some density or another. Therein lies the rub, QIC has a large number of densities on physically similar (sometimes identical) tapes.

QIC drives are not quiet. These drives audibly seek before they begin to record data and are clearly audible whenever reading, writing or seeking. QIC tapes measure 6 x 4 x 0.7 inches (152 x 102 x 17 mm).

Data throughput ranges from ~150 kB/s to ~500 kB/s. Data capacity ranges from 40 MB to 15 GB. Hardware compression is available on many of the newer QIC drives. QIC drives are less frequently installed; they are being supplanted by DAT drives.

Data is recorded onto the tape in tracks. The tracks run along the long axis of the tape media from one end to the other. The number of tracks, and therefore the width of a track, varies with the tape's capacity. Most if not all newer drives provide backward-compatibility at least for reading (but often also for writing). QIC has a good reputation regarding the safety of the data (the mechanics are simpler and more robust than for helical scan drives).

Tapes should be retired from use after 5,000 backups.

18.9.4 DLT

DLT has the fastest data transfer rate of all the drive types listed here. The 1/2" (12.5mm) tape is contained in a single spool cartridge (4 x 4 x 1 inches; 100 x 100 x 25 mm). The cartridge has a swinging gate along one entire side of the cartridge. The drive mechanism opens this gate to extract the tape leader. The tape leader has an oval hole in it which the drive uses to "hook" the tape. The take-up spool is located inside the tape drive. All the other tape cartridges listed here (9 track tapes are the only exception) have both the supply and take-up spools located inside the tape cartridge itself.

Data throughput is approximately 1.5 MB/s, three times the throughput of 4mm, 8mm, or QIC tape drives. Data capacities range from 10 GB to 20 GB for a single drive. Drives are available in both multi-tape changers and multi-tape, multi-drive tape libraries containing from 5 to 900 tapes over 1 to 20 drives, providing from 50 GB to 9 TB of storage.

With compression, DLT Type IV format supports up to 70 GB capacity.

Data is recorded onto the tape in tracks parallel to the direction of travel (just like QIC tapes). Two tracks are written at once. Read/write head lifetimes are relatively long; once the tape stops moving, there is no relative motion between the heads and the tape.

18.9.5 AIT

AIT is a new format from Sony, and can hold up to 50 GB (with compression) per tape. The tapes contain memory chips which retain an index of the tape's contents. This index can be rapidly read by the tape drive to determine the position of files on the tape, instead of the several minutes that would be required for other tapes. Software such as **SAMS:Alexandria** can operate forty or more AIT tape libraries, communicating directly with the tape's memory chip to display the contents on screen, determine what files were backed up to which tape, locate the correct tape, load it, and restore the data from the tape.

Libraries like this cost in the region of $20,000, pricing them a little out of the hobbyist market.

18.9.6 Using a New Tape for the First Time

The first time that you try to read or write a new, completely blank tape, the operation will fail. The console messages should be similar to:

```
sa0(ncr1:4:0): NOT READY asc:4,1
```

```
sa0(ncr1:4:0):  Logical unit is in process of becoming ready
```

The tape does not contain an Identifier Block (block number 0). All QIC tape drives since the adoption of QIC-525 standard write an Identifier Block to the tape. There are two solutions:

- `mt fsf 1` causes the tape drive to write an Identifier Block to the tape.
- Use the front panel button to eject the tape.

  Re-insert the tape and `dump` data to the tape.

  `dump` will report `DUMP: End of tape detected` and the console will show: `HARDWARE FAILURE info:280 asc:80,96`.

  rewind the tape using: `mt rewind`.

  Subsequent tape operations are successful.

18.10 Backups to Floppies

18.10.1 Can I Use Floppies for Backing Up My Data?

Floppy disks are not really a suitable media for making backups as:

- The media is unreliable, especially over long periods of time.
- Backing up and restoring is very slow.
- They have a very limited capacity (the days of backing up an entire hard disk onto a dozen or so floppies has long since passed).

However, if you have no other method of backing up your data then floppy disks are better than no backup at all.

If you do have to use floppy disks then ensure that you use good quality ones. Floppies that have been lying around the office for a couple of years are a bad choice. Ideally use new ones from a reputable manufacturer.

18.10.2 So How Do I Backup My Data to Floppies?

The best way to backup to floppy disk is to use tar(1) with the `-M` (multi volume) option, which allows backups to span multiple floppies.

To backup all the files in the current directory and sub-directory use this (as `root`):

```
# tar Mcvf /dev/fd0 *
```

When the first floppy is full tar(1) will prompt you to insert the next volume (because tar(1) is media independent it refers to volumes; in this context it means floppy disk).

```
Prepare volume #2 for /dev/fd0 and hit return:
```

This is repeated (with the volume number incrementing) until all the specified files have been archived.

18.10.3 Can I Compress My Backups?

Unfortunately, tar(1) will not allow the `-z` option to be used for multi-volume archives. You could, of course, gzip(1) all the files, tar(1) them to the floppies, then gunzip(1) the files again!

18.10.4 How Do I Restore My Backups?

To restore the entire archive use:

```
# tar Mxvf /dev/fd0
```

There are two ways that you can use to restore only specific files. First, you can start with the first floppy and use:

```
# tar Mxvf /dev/fd0 filename
```

The utility tar(1) will prompt you to insert subsequent floppies until it finds the required file.

Alternatively, if you know which floppy the file is on then you can simply insert that floppy and use the same command as above. Note that if the first file on the floppy is a continuation from the previous one then tar(1) will warn you that it cannot restore it, even if you have not asked it to!

18.11 Backup Strategies

The first requirement in devising a backup plan is to make sure that all of the following problems are covered:

- Disk failure
- Accidental file deletion
- Random file corruption
- Complete machine destruction (e.g. fire), including destruction of any on-site backups.

It is perfectly possible that some systems will be best served by having each of these problems covered by a completely different technique. Except for strictly personal systems with very low-value data, it is unlikely that one technique would cover all of them.

Some of the techniques in the toolbox are:

- Archives of the whole system, backed up onto permanent media offsite. This actually provides protection against all of the possible problems listed above, but is slow and inconvenient to restore from. You can keep copies of the backups onsite and/or online, but there will still be inconveniences in restoring files, especially for non-privileged users.
- Filesystem snapshots. This is really only helpful in the accidental file deletion scenario, but it can be *very* helpful in that case, and is quick and easy to deal with.
- Copies of whole filesystems and/or disks (e.g. periodic rsync(1) of the whole machine). This is generally most useful in networks with unique requirements. For general protection against disk failure, it is usually inferior to RAID. For restoring accidentally deleted files, it can be comparable to UFS snapshots, but that depends on your preferences.

- RAID. Minimizes or avoids downtime when a disk fails. At the expense of having to deal with disk failures more often (because you have more disks), albeit at a much lower urgency.
- Checking fingerprints of files. The mtree(8) utility is very useful for this. Although it is not a backup technique, it helps guarantee that you will notice when you need to resort to your backups. This is particularly important for offline backups, and should be checked periodically.

It is quite easy to come up with even more techniques, many of them variations on the ones listed above. Specialized requirements will usually lead to specialized techniques (for example, backing up a live database usually requires a method particular to the database software as an intermediate step). The important thing is to know what dangers you want to protect against, and how you will handle each.

18.12 Backup Basics

The three major backup programs are dump(8), tar(1), and cpio(1).

18.12.1 Dump and Restore

The traditional UNIX backup programs are `dump` and `restore`. They operate on the drive as a collection of disk blocks, below the abstractions of files, links and directories that are created by the file systems. Unlike other backup software, `dump` backs up an entire file system on a device. It is unable to backup only part of a file system or a directory tree that spans more than one file system. The `dump` command does not write files and directories to tape, but rather writes the raw data blocks that comprise files and directories. When being used to extract data, `restore` stores temporary files in `/tmp/` by default — if you are operating from a recovery disk with a small `/tmp` directory, you may need to set the `TMPDIR` environment variable to a directory with more free space for the restore to be successful.

> **Note:** If you use `dump` on your root directory, you would not back up `/home`, `/usr` or many other directories since these are typically mount points for other file systems or symbolic links into those file systems.

`dump` has quirks that remain from its early days in Version 6 of AT&T UNIX (circa 1975). The default parameters are suitable for 9-track tapes (6250 bpi), not the high-density media available today (up to 62,182 ftpi). These defaults must be overridden on the command line to utilize the capacity of current tape drives.

It is also possible to backup data across the network to a tape drive attached to another computer with `rdump` and `rrestore`. Both programs rely upon rcmd(3) and ruserok(3) to access the remote tape drive. Therefore, the user performing the backup must be listed in the `.rhosts` file on the remote computer. The arguments to `rdump` and `rrestore` must be suitable to use on the remote computer. When `rdump`ing from a FreeBSD computer to an Exabyte tape drive connected to a Sun called `komodo`, use:

```
# /sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nsa8 /dev/da0a 2>&1
```

Beware: there are security implications to allowing `.rhosts` authentication. Evaluate your situation carefully.

It is also possible to use `dump` and `restore` in a more secure fashion over `ssh`.

Example 18-1. Using dump over ssh

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh -c blowfish \
```

```
targetuser@targetmachine.example.com dd of=/mybigfiles/dump-usr-l0.gz
```

Or using `dump`'s built-in method, setting the environment variable `RSH`:

Example 18-2. Using dump over ssh with `RSH` set

```
# RSH=/usr/bin/ssh /sbin/dump -0uan -f targetuser@targetmachine.example.com:/dev/sa0 /usr
```

18.12.2 `tar`

tar(1) also dates back to Version 6 of AT&T UNIX (circa 1975). `tar` operates in cooperation with the file system; it writes files and directories to tape. `tar` does not support the full range of options that are available from cpio(1), but it does not require the unusual command pipeline that `cpio` uses.

To `tar` to an Exabyte tape drive connected to a Sun called `komodo`, use:

```
# tar cf - . | rsh komodo dd of=tape-device obs=20b
```

If you are worried about the security of backing up over a network you should use the `ssh` command instead of `rsh`.

18.12.3 `cpio`

cpio(1) is the original UNIX file interchange tape program for magnetic media. `cpio` has options (among many others) to perform byte-swapping, write a number of different archive formats, and pipe the data to other programs. This last feature makes `cpio` an excellent choice for installation media. `cpio` does not know how to walk the directory tree and a list of files must be provided through `stdin`.

`cpio` does not support backups across the network. You can use a pipeline and `rsh` to send the data to a remote tape drive.

```
# for f in directory_list; do
find $f >> backup.list
done
# cpio -v -o --format=newc < backup.list | ssh user@host "cat > backup_device"
```

Where *`directory_list`* is the list of directories you want to back up, *`user@host`* is the user/hostname combination that will be performing the backups, and *`backup_device`* is where the backups should be written to (e.g., `/dev/nsa0`).

18.12.4 `pax`

pax(1) is IEEE/POSIX's answer to `tar` and `cpio`. Over the years the various versions of `tar` and `cpio` have gotten slightly incompatible. So rather than fight it out to fully standardize them, POSIX created a new archive utility. `pax` attempts to read and write many of the various `cpio` and `tar` formats, plus new formats of its own. Its command set more resembles `cpio` than `tar`.

18.12.5 Amanda

Amanda (Advanced Maryland Network Disk Archiver) is a client/server backup system, rather than a single program. An **Amanda** server will backup to a single tape drive any number of computers that have **Amanda** clients and a network connection to the **Amanda** server. A common problem at sites with a number of large disks is that the length of time required to backup to data directly to tape exceeds the amount of time available for the task. **Amanda** solves this problem. **Amanda** can use a "holding disk" to backup several file systems at the same time. **Amanda** creates "archive sets": a group of tapes used over a period of time to create full backups of all the file systems listed in **Amanda**'s configuration file. The "archive set" also contains nightly incremental (or differential) backups of all the file systems. Restoring a damaged file system requires the most recent full backup and the incremental backups.

The configuration file provides fine control of backups and the network traffic that **Amanda** generates. **Amanda** will use any of the above backup programs to write the data to tape. **Amanda** is available as either a port or a package, it is not installed by default.

18.12.6 Do Nothing

"Do nothing" is not a computer program, but it is the most widely used backup strategy. There are no initial costs. There is no backup schedule to follow. Just say no. If something happens to your data, grin and bear it!

If your time and your data is worth little to nothing, then "Do nothing" is the most suitable backup program for your computer. But beware, UNIX is a useful tool, you may find that within six months you have a collection of files that are valuable to you.

"Do nothing" is the correct backup method for `/usr/obj` and other directory trees that can be exactly recreated by your computer. An example is the files that comprise the HTML or PostScript version of this Handbook. These document formats have been created from SGML input files. Creating backups of the HTML or PostScript files is not necessary. The SGML files are backed up regularly.

18.12.7 Which Backup Program Is Best?

dump(8) *Period.* Elizabeth D. Zwicky torture tested all the backup programs discussed here. The clear choice for preserving all your data and all the peculiarities of UNIX file systems is `dump`. Elizabeth created file systems containing a large variety of unusual conditions (and some not so unusual ones) and tested each program by doing a backup and restore of those file systems. The peculiarities included: files with holes, files with holes and a block of nulls, files with funny characters in their names, unreadable and unwritable files, devices, files that change size during the backup, files that are created/deleted during the backup and more. She presented the results at LISA V in Oct. 1991. See torture-testing Backup and Archive Programs (http://www.coredumps.de/doc/dump/zwicky/testdump.doc.html).

18.12.8 Emergency Restore Procedure

18.12.8.1 Before the Disaster

There are only four steps that you need to perform in preparation for any disaster that may occur.

First, print the bsdlabel from each of your disks (e.g. `bsdlabel da0 | lpr`), your file system table (`/etc/fstab`) and all boot messages, two copies of each.

Second, burn a “livefs” CDROM. This CDROM contains support for booting into a FreeBSD “livefs” rescue mode allowing the user to perform many tasks like running dump(8), restore(8), fdisk(8), bsdlabel(8), newfs(8), mount(8), and more. Livefs CD image for FreeBSD/i386 8.2-RELEASE is available from ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.2/FreeBSD-8.2-RELEASE-i386-livefs.iso.

Third, create backup tapes regularly. Any changes that you make after your last backup may be irretrievably lost. Write-protect the backup tapes.

Fourth, test the “livefs” CDROM you made in step two and backup tapes. Make notes of the procedure. Store these notes with the CDROM, the printouts and the backup tapes. You will be so distraught when restoring that the notes may prevent you from destroying your backup tapes (How? In place of `tar xvf /dev/sa0`, you might accidentally type `tar cvf /dev/sa0` and over-write your backup tape).

For an added measure of security, make “livefs” CDROM and two backup tapes each time. Store one of each at a remote location. A remote location is NOT the basement of the same office building. A number of firms in the World Trade Center learned this lesson the hard way. A remote location should be physically separated from your computers and disk drives by a significant distance.

18.12.8.2 After the Disaster

The key question is: did your hardware survive? You have been doing regular backups so there is no need to worry about the software.

If the hardware has been damaged, the parts should be replaced before attempting to use the computer.

If your hardware is okay, insert the “livefs” CDROM in the CDROM drive and boot the computer. The original install menu will be displayed on the screen. Select the correct country, then choose Fixit -- Repair mode with CDROM/DVD/floppy or start a shell. option and select the CDROM/DVD -- Use the live filesystem CDROM/DVD item. The tool `restore` and the other programs that you need are located in `/mnt2/rescue`.

Recover each file system separately.

Try to `mount` (e.g. `mount /dev/da0a /mnt`) the root partition of your first disk. If the bsdlabel was damaged, use `bsdlabel` to re-partition and label the disk to match the label that you printed and saved. Use `newfs` to re-create the file systems. Re-mount the root partition of the disk read-write (`mount -u -o rw /mnt`). Use your backup program and backup tapes to recover the data for this file system (e.g. `restore vrf /dev/sa0`). Unmount the file system (e.g. `umount /mnt`). Repeat for each file system that was damaged.

Once your system is running, backup your data onto new tapes. Whatever caused the crash or data loss may strike again. Another hour spent now may save you from further distress later.

18.13 Network, Memory, and File-Backed File Systems

Aside from the disks you physically insert into your computer: floppies, CDs, hard drives, and so forth; other forms of disks are understood by FreeBSD - the *virtual disks*.

These include network file systems such as the Network File System and Coda, memory-based file systems and file-backed file systems.

According to the FreeBSD version you run, you will have to use different tools for creation and use of file-backed and memory-based file systems.

Note: Use devfs(5) to allocate device nodes transparently for the user.

18.13.1 File-Backed File System

The utility mdconfig(8) is used to configure and enable memory disks, md(4), under FreeBSD. To use mdconfig(8), you have to load md(4) module or to add the support in your kernel configuration file:

```
device md
```

The mdconfig(8) command supports three kinds of memory backed virtual disks: memory disks allocated with malloc(9), memory disks using a file or swap space as backing. One possible use is the mounting of floppy or CD images kept in files.

To mount an existing file system image:

Example 18-3. Using `mdconfig` to Mount an Existing File System Image

```
# mdconfig -a -t vnode -f diskimage -u 0
# mount /dev/md0 /mnt
```

To create a new file system image with mdconfig(8):

Example 18-4. Creating a New File-Backed Disk with `mdconfig`

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdconfig -a -t vnode -f newimage -u 0
# bsdlabel -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
```

If you do not specify the unit number with the `-u` option, mdconfig(8) will use the md(4) automatic allocation to select an unused device. The name of the allocated unit will be output on stdout like `md4`. For more details about mdconfig(8), please refer to the manual page.

The utility mdconfig(8) is very useful, however it asks many command lines to create a file-backed file system. FreeBSD also comes with a tool called mdmfs(8), this program configures a md(4) disk using mdconfig(8), puts a UFS file system on it using newfs(8), and mounts it using mount(8). For example, if you want to create and mount the same file system image as above, simply type the following:

Example 18-5. Configure and Mount a File-Backed Disk with `mdmfs`

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdmfs -F newimage -s 5m md0 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md0        4718    4  4338     0%    /mnt
```

If you use the option `md` without unit number, mdmfs(8) will use md(4) auto-unit feature to automatically select an unused device. For more details about mdmfs(8), please refer to the manual page.

18.13.2 Memory-Based File System

For a memory-based file system the "swap backing" should normally be used. Using swap backing does not mean that the memory disk will be swapped out to disk by default, but merely that the memory disk will be allocated from a memory pool which can be swapped out to disk if needed. It is also possible to create memory-based disk which are malloc(9) backed, but using malloc backed memory disks, especially large ones, can result in a system panic if the kernel runs out of memory.

Example 18-6. Creating a New Memory-Based Disk with `mdconfig`

```
# 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
```

Example 18-7. Creating a New Memory-Based Disk with `mdmfs`

```
# 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 Detaching a Memory Disk from the System

When a memory-based or file-based file system is not used, you should release all resources to the system. The first thing to do is to unmount the file system, then use mdconfig(8) to detach the disk from the system and release the resources.

For example to detach and free all resources used by `/dev/md4`:

```
# mdconfig -d -u 4
```

It is possible to list information about configured md(4) devices in using the command `mdconfig -l`.

18.14 File System Snapshots

FreeBSD offers a feature in conjunction with Soft Updates: File system snapshots.

Snapshots allow a user to create images of specified file systems, and treat them as a file. Snapshot files must be created in the file system that the action is performed on, and a user may create no more than 20 snapshots per file system. Active snapshots are recorded in the superblock so they are persistent across unmount and remount operations along with system reboots. When a snapshot is no longer required, it can be removed with the standard rm(1) command. Snapshots may be removed in any order, however all the used space may not be acquired because another snapshot will possibly claim some of the released blocks.

The un-alterable `snapshot` file flag is set by mksnap_ffs(8) after initial creation of a snapshot file. The unlink(1) command makes an exception for snapshot files since it allows them to be removed.

Snapshots are created with the mount(8) command. To place a snapshot of `/var` in the file `/var/snapshot/snap` use the following command:

```
# mount -u -o snapshot /var/snapshot/snap /var
```

Alternatively, you can use mksnap_ffs(8) to create a snapshot:

```
# mksnap_ffs /var /var/snapshot/snap
```

One can find snapshot files on a file system (e.g. `/var`) by using the find(1) command:

```
# find /var -flags snapshot
```

Once a snapshot has been created, it has several uses:

- Some administrators will use a snapshot file for backup purposes, because the snapshot can be transfered to CDs or tape.
- The file system integrity checker, fsck(8), may be run on the snapshot. Assuming that the file system was clean when it was mounted, you should always get a clean (and unchanging) result. This is essentially what the background fsck(8) process does.
- Run the dump(8) utility on the snapshot. A dump will be returned that is consistent with the file system and the timestamp of the snapshot. dump(8) can also take a snapshot, create a dump image and then remove the snapshot in one command using the `-L` flag.
- mount(8) the snapshot as a frozen image of the file system. To mount(8) the snapshot `/var/snapshot/snap` run:

  ```
  # mdconfig -a -t vnode -f /var/snapshot/snap -u 4
  # mount -r /dev/md4 /mnt
  ```

You can now walk the hierarchy of your frozen `/var` file system mounted at `/mnt`. Everything will initially be in the same state it was during the snapshot creation time. The only exception is that any earlier snapshots will appear as zero length files. When the use of a snapshot has delimited, it can be unmounted with:

```
# umount /mnt
# mdconfig -d -u 4
```

For more information about `softupdates` and file system snapshots, including technical papers, you can visit Marshall Kirk McKusick's website at http://www.mckusick.com/.

18.15 File System Quotas

Quotas are an optional feature of the operating system that allow you to limit the amount of disk space and/or the number of files a user or members of a group may allocate on a per-file system basis. This is used most often on timesharing systems where it is desirable to limit the amount of resources any one user or group of users may allocate. This will prevent one user or group of users from consuming all of the available disk space.

18.15.1 Configuring Your System to Enable Disk Quotas

Before attempting to use disk quotas, it is necessary to make sure that quotas are configured in your kernel. This is done by adding the following line to your kernel configuration file:

```
options QUOTA
```

The stock `GENERIC` kernel does not have this enabled by default, so you will have to configure, build and install a custom kernel in order to use disk quotas. Please refer to Chapter 8 for more information on kernel configuration.

Next you will need to enable disk quotas in `/etc/rc.conf`. This is done by adding the line:

```
enable_quotas="YES"
```

For finer control over your quota startup, there is an additional configuration variable available. Normally on bootup, the quota integrity of each file system is checked by the quotacheck(8) program. The quotacheck(8) facility insures that the data in the quota database properly reflects the data on the file system. This is a very time consuming process that will significantly affect the time your system takes to boot. If you would like to skip this step, a variable in `/etc/rc.conf` is made available for the purpose:

```
check_quotas="NO"
```

Finally you will need to edit `/etc/fstab` to enable disk quotas on a per-file system basis. This is where you can either enable user or group quotas or both for all of your file systems.

To enable per-user quotas on a file system, add the `userquota` option to the options field in the `/etc/fstab` entry for the file system you want to enable quotas on. For example:

```
/dev/da1s2g   /home    ufs rw,userquota 1 2
```

Similarly, to enable group quotas, use the `groupquota` option instead of `userquota`. To enable both user and group quotas, change the entry as follows:

```
/dev/da1s2g    /home    ufs rw,userquota,groupquota 1 2
```

By default, the quota files are stored in the root directory of the file system with the names `quota.user` and `quota.group` for user and group quotas respectively. See fstab(5) for more information. Even though the fstab(5)

manual page says that you can specify an alternate location for the quota files, this is not recommended because the various quota utilities do not seem to handle this properly.

At this point you should reboot your system with your new kernel. `/etc/rc` will automatically run the appropriate commands to create the initial quota files for all of the quotas you enabled in `/etc/fstab`, so there is no need to manually create any zero length quota files.

In the normal course of operations you should not be required to run the quotacheck(8), quotaon(8), or quotaoff(8) commands manually. However, you may want to read their manual pages just to be familiar with their operation.

18.15.2 Setting Quota Limits

Once you have configured your system to enable quotas, verify that they really are enabled. An easy way to do this is to run:

```
# quota -v
```

You should see a one line summary of disk usage and current quota limits for each file system that quotas are enabled on.

You are now ready to start assigning quota limits with the edquota(8) command.

You have several options on how to enforce limits on the amount of disk space a user or group may allocate, and how many files they may create. You may limit allocations based on disk space (block quotas) or number of files (inode quotas) or a combination of both. Each of these limits are further broken down into two categories: hard and soft limits.

A hard limit may not be exceeded. Once a user reaches his hard limit he may not make any further allocations on the file system in question. For example, if the user has a hard limit of 500 kbytes on a file system and is currently using 490 kbytes, the user can only allocate an additional 10 kbytes. Attempting to allocate an additional 11 kbytes will fail.

Soft limits, on the other hand, can be exceeded for a limited amount of time. This period of time is known as the grace period, which is one week by default. If a user stays over his or her soft limit longer than the grace period, the soft limit will turn into a hard limit and no further allocations will be allowed. When the user drops back below the soft limit, the grace period will be reset.

The following is an example of what you might see when you run the edquota(8) command. When the edquota(8) command is invoked, you are placed into the editor specified by the `EDITOR` environment variable, or in the **vi** editor if the `EDITOR` variable is not set, to allow you to edit the quota limits.

```
# edquota -u test
```

```
Quotas for user test:
/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)
```

You will normally see two lines for each file system that has quotas enabled. One line for the block limits, and one line for inode limits. Simply change the value you want updated to modify the quota limit. For example, to raise this user's block limit from a soft limit of 50 and a hard limit of 75 to a soft limit of 500 and a hard limit of 600, change:

```
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
```

to:

```
/usr: kbytes in use: 65, limits (soft = 500, hard = 600)
```

The new quota limits will be in place when you exit the editor.

Sometimes it is desirable to set quota limits on a range of UIDs. This can be done by use of the `-p` option on the edquota(8) command. First, assign the desired quota limit to a user, and then run `edquota -p protouser startuid-enduid`. For example, if user `test` has the desired quota limits, the following command can be used to duplicate those quota limits for UIDs 10,000 through 19,999:

```
# edquota -p test 10000-19999
```

For more information see edquota(8) manual page.

18.15.3 Checking Quota Limits and Disk Usage

You can use either the quota(1) or the repquota(8) commands to check quota limits and disk usage. The quota(1) command can be used to check individual user or group quotas and disk usage. A user may only examine his own quota, and the quota of a group he is a member of. Only the super-user may view all user and group quotas. The repquota(8) command can be used to get a summary of all quotas and disk usage for file systems with quotas enabled.

The following is some sample output from the `quota -v` command for a user that has quota limits on two file systems.

```
Disk quotas for user test (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
```

On the `/usr` file system in the above example, this user is currently 15 kbytes over the soft limit of 50 kbytes and has 5 days of the grace period left. Note the asterisk `*` which indicates that the user is currently over his quota limit.

Normally file systems that the user is not using any disk space on will not show up in the output from the quota(1) command, even if he has a quota limit assigned for that file system. The `-v` option will display those file systems, such as the `/usr/var` file system in the above example.

18.15.4 Quotas over NFS

Quotas are enforced by the quota subsystem on the NFS server. The rpc.rquotad(8) daemon makes quota information available to the quota(1) command on NFS clients, allowing users on those machines to see their quota statistics.

Enable `rpc.rquotad` in `/etc/inetd.conf` like so:

```
rquotad/1      dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

Now restart `inetd`:

```
# /etc/rc.d/inetd restart
```

18.16 Encrypting Disk Partitions

FreeBSD offers excellent online protections against unauthorized data access. File permissions and Mandatory Access Control (MAC) (see Chapter 16) help prevent unauthorized third-parties from accessing data while the operating system is active and the computer is powered up. However, the permissions enforced by the operating system are irrelevant if an attacker has physical access to a computer and can simply move the computer's hard drive to another system to copy and analyze the sensitive data.

Regardless of how an attacker may have come into possession of a hard drive or powered-down computer, both **GEOM Based Disk Encryption (gbde)** and `geli` cryptographic subsystems in FreeBSD are able to protect the data on the computer's file systems against even highly-motivated attackers with significant resources. Unlike cumbersome encryption methods that encrypt only individual files, `gbde` and `geli` transparently encrypt entire file systems. No cleartext ever touches the hard drive's platter.

18.16.1 Disk Encryption with gbde

1. Become `root`

   Configuring **gbde** requires super-user privileges.

   ```
   % su -
   Password:
   ```

2. Add gbde(4) Support to the Kernel Configuration File

   Add the following line to the kernel configuration file:

   ```
   options GEOM_BDE
   ```

   Rebuild the kernel as described in Chapter 8.

   Reboot into the new kernel.

3. An alternative to recompiling the kernel is to use `kldload` to load gbde(4):

   ```
   # kldload geom_bde
   ```

18.16.1.1 Preparing the Encrypted Hard Drive

The following example assumes that you are adding a new hard drive to your system that will hold a single encrypted partition. This partition will be mounted as `/private`. **gbde** can also be used to encrypt `/home` and `/var/mail`, but this requires more complex instructions which exceed the scope of this introduction.

1. Add the New Hard Drive

   Install the new drive to the system as explained in Section 18.3. For the purposes of this example, a new hard drive partition has been added as `/dev/ad4s1c`. The `/dev/ad0s1*` devices represent existing standard FreeBSD partitions on the example system.

   ```
   # 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. Create a Directory to Hold gbde Lock Files

   ```
   # mkdir /etc/gbde
   ```

The **gbde** lock file contains information that **gbde** requires to access encrypted partitions. Without access to the lock file, **gbde** will not be able to decrypt the data contained in the encrypted partition without significant manual intervention which is not supported by the software. Each encrypted partition uses a separate lock file.

3. Initialize the gbde Partition

   A **gbde** partition must be initialized before it can be used. This initialization needs to be performed only once:

   ```
   # gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c.lock
   ```

   gbde(8) will open your editor, permitting you to set various configuration options in a template. For use with UFS1 or UFS2, set the sector_size to 2048:

   ```
   $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
   [...]
   ```

   gbde(8) will ask you twice to type the passphrase that should be used to secure the data. The passphrase must be the same both times. **gbde**'s ability to protect your data depends entirely on the quality of the passphrase that you choose. [1]

   The `gbde init` command creates a lock file for your **gbde** partition that in this example is stored as `/etc/gbde/ad4s1c.lock`. **gbde** lock files must end in ".lock" in order to be correctly detected by the `/etc/rc.d/gbde` start up script.

   > **Caution: gbde** lock files *must* be backed up together with the contents of any encrypted partitions. While deleting a lock file alone cannot prevent a determined attacker from decrypting a **gbde** partition, without the lock file, the legitimate owner will be unable to access the data on the encrypted partition without a significant amount of work that is totally unsupported by gbde(8) and its designer.

4. Attach the Encrypted Partition to the Kernel

   ```
   # gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
   ```

   You will be asked to provide the passphrase that you selected during the initialization of the encrypted partition. The new encrypted device will show up in `/dev` as `/dev/device_name.bde`:

   ```
   # 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. Create a File System on the Encrypted Device

   Once the encrypted device has been attached to the kernel, you can create a file system on the device. To create a file system on the encrypted device, use newfs(8). Since it is much faster to initialize a new UFS2 file system than it is to initialize the old UFS1 file system, using newfs(8) with the `-O2` option is recommended.

   ```
   # newfs -U -O2 /dev/ad4s1c.bde
   ```

Note: The newfs(8) command must be performed on an attached **gbde** partition which is identified by a `*.bde` extension to the device name.

6. Mount the Encrypted Partition

   Create a mount point for the encrypted file system.

   ```
   # mkdir /private
   ```

   Mount the encrypted file system.

   ```
   # mount /dev/ad4s1c.bde /private
   ```

7. Verify That the Encrypted File System is Available

   The encrypted file system should now be visible to df(1) and be available for use.

   ```
   % 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 Mounting Existing Encrypted File Systems

After each boot, any encrypted file systems must be re-attached to the kernel, checked for errors, and mounted, before the file systems can be used. The required commands must be executed as user `root`.

1. Attach the gbde Partition to the Kernel

   ```
   # gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
   ```

   You will be asked to provide the passphrase that you selected during initialization of the encrypted **gbde** partition.

2. Check the File System for Errors

   Since encrypted file systems cannot yet be listed in `/etc/fstab` for automatic mounting, the file systems must be checked for errors by running fsck(8) manually before mounting.

   ```
   # fsck -p -t ffs /dev/ad4s1c.bde
   ```

3. Mount the Encrypted File System

   ```
   # mount /dev/ad4s1c.bde /private
   ```

   The encrypted file system is now available for use.

18.16.1.2.1 Automatically Mounting Encrypted Partitions

It is possible to create a script to automatically attach, check, and mount an encrypted partition, but for security reasons the script should not contain the gbde(8) password. Instead, it is recommended that such scripts be run manually while providing the password via the console or ssh(1).

As an alternative, an `rc.d` script is provided. Arguments for this script can be passed via rc.conf(5), for example:

```
gbde_autoattach_all="YES"
gbde_devices="ad4s1c"
gbde_lockdir="/etc/gbde"
```

This will require that the **gbde** passphrase be entered at boot time. After typing the correct passphrase, the **gbde** encrypted partition will be mounted automatically. This can be very useful when using **gbde** on notebooks.

18.16.1.3 Cryptographic Protections Employed by gbde

gbde(8) encrypts the sector payload using 128-bit AES in CBC mode. Each sector on the disk is encrypted with a different AES key. For more information on **gbde**'s cryptographic design, including how the sector keys are derived from the user-supplied passphrase, see gbde(4).

18.16.1.4 Compatibility Issues

sysinstall(8) is incompatible with **gbde**-encrypted devices. All `*.bde` devices must be detached from the kernel before starting sysinstall(8) or it will crash during its initial probing for devices. To detach the encrypted device used in our example, use the following command:

```
# gbde detach /dev/ad4s1c
```

Also note that, as vinum(4) does not use the geom(4) subsystem, you cannot use **gbde** with **vinum** volumes.

18.16.2 Disk Encryption with `geli`

An alternative cryptographic GEOM class is available - `geli`. It is currently being developed by Pawel Jakub Dawidek <`pjd@FreeBSD.org`>. The `geli` utility is different to `gbde`; it offers different features and uses a different scheme for doing cryptographic work.

The most important features of geli(8) are:

- Utilizes the crypto(9) framework — when cryptographic hardware is available, `geli` will use it automatically.
- Supports multiple cryptographic algorithms (currently AES, Blowfish, and 3DES).
- Allows the root partition to be encrypted. The passphrase used to access the encrypted root partition will be requested during the system boot.
- Allows the use of two independent keys (e.g. a "key" and a "company key").
- `geli` is fast - performs simple sector-to-sector encryption.
- Allows backup and restore of Master Keys. When a user has to destroy his keys, it will be possible to get access to the data again by restoring keys from the backup.
- Allows to attach a disk with a random, one-time key — useful for swap partitions and temporary file systems.

More `geli` features can be found in the geli(8) manual page.

The next steps will describe how to enable support for `geli` in the FreeBSD kernel and will explain how to create and use a `geli` encryption provider.

Super-user privileges will be required since modifications to the kernel are necessary.

1. Adding `geli` Support to the Kernel

   Add the following lines to the kernel configuration file:

   ```
   options GEOM_ELI
   device crypto
   ```

   Rebuild the kernel as described in Chapter 8.

   Alternatively, the `geli` module can be loaded at boot time. Add the following line to the `/boot/loader.conf`:

   ```
   geom_eli_load="YES"
   ```

   geli(8) should now be supported by the kernel.

2. Generating the Master Key

   The following example will describe how to generate a key file, which will be used as part of the Master Key for the encrypted provider mounted under `/private`. The key file will provide some random data used to encrypt the Master Key. The Master Key will be protected by a passphrase as well. Provider's sector size will be 4kB big. Furthermore, the discussion will describe how to attach the `geli` provider, create a file system on it, how to mount it, how to work with it, and finally how to detach it.

   It is recommended to use a bigger sector size (like 4kB) for better performance.

   The Master Key will be protected with a passphrase and the data source for key file will be `/dev/random`. The sector size of `/dev/da2.eli`, which we call provider, will be 4kB.

   ```
   # 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:
   ```

   It is not mandatory that both a passphrase and a key file are used; either method of securing the Master Key can be used in isolation.

   If key file is given as "-", standard input will be used. This example shows how more than one key file can be used.

   ```
   # cat keyfile1 keyfile2 keyfile3 | geli init -K - /dev/da2
   ```

3. Attaching the Provider with the generated Key

   ```
   # geli attach -k /root/da2.key /dev/da2
   Enter passphrase:
   ```

   The new plaintext device will be named `/dev/da2.eli`.

   ```
   # ls /dev/da2*
   /dev/da2  /dev/da2.eli
   ```

4. Creating the new File System

   ```
   # dd if=/dev/random of=/dev/da2.eli bs=1m
   # newfs /dev/da2.eli
   # mount /dev/da2.eli /private
   ```

   The encrypted file system should be visible to df(1) and be available for use now:

```
# 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. Unmounting and Detaching the Provider

   Once the work on the encrypted partition is done, and the `/private` partition is no longer needed, it is prudent to consider unmounting and detaching the `geli` encrypted partition from the kernel.

   ```
   # umount /private
   # geli detach da2.eli
   ```

More information about the use of geli(8) can be found in the manual page.

18.16.2.1 Using the `geli rc.d` Script

`geli` comes with a `rc.d` script which can be used to simplify the usage of `geli`. An example of configuring `geli` through rc.conf(5) follows:

```
geli_devices="da2"
geli_da2_flags="-p -k /root/da2.key"
```

This will configure `/dev/da2` as a `geli` provider of which the Master Key file is located in `/root/da2.key`, and `geli` will not use a passphrase when attaching the provider (note that this can only be used if `-P` was given during the `geli init` phase). The system will detach the `geli` provider from the kernel before the system shuts down.

More information about configuring `rc.d` is provided in the rc.d section of the Handbook.

18.17 Encrypting Swap Space

Swap encryption in FreeBSD is easy to configure . Depending on which version of FreeBSD is being used, different options are available and configuration can vary slightly. The gbde(8) or geli(8) encryption systems can be used for swap encryption. Both systems use the `encswap` rc.d script.

The previous section, Encrypting Disk Partitions, includes a short discussion on the different encryption systems.

18.17.1 Why should Swap be Encrypted?

Like the encryption of disk partitions, encryption of swap space is done to protect sensitive information. Imagine an application that e.g. deals with passwords. As long as these passwords stay in physical memory, all is well. However, if the operating system starts swapping out memory pages to free space for other applications, the passwords may be written to the disk platters unencrypted and easy to retrieve for an adversary. Encrypting swap space can be a solution for this scenario.

18.17.2 Preparation

> **Note:** For the remainder of this section, `ad0s1b` will be the swap partition.

Up to this point the swap has been unencrypted. It is possible that there are already passwords or other sensitive data on the disk platters in cleartext. To rectify this, the data on the swap partition should be overwritten with random garbage:

```
# dd if=/dev/random of=/dev/ad0s1b bs=1m
```

18.17.3 Swap Encryption with gbde(8)

The `.bde` suffix should be added to the device in the respective `/etc/fstab` swap line:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/ad0s1b.bde         none            swap    sw              0       0
```

18.17.4 Swap Encryption with geli(8)

Alternatively, the procedure for using geli(8) for swap encryption is similar to that of using gbde(8). The `.eli` suffix should be added to the device in the respective `/etc/fstab` swap line:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/ad0s1b.eli         none            swap    sw              0       0
```

geli(8) uses the AES algorithm with a key length of 256 bit by default.

Optionally, these defaults can be altered using the `geli_swap_flags` option in `/etc/rc.conf`. The following line tells the `encswap` rc.d script to create geli(8) swap partitions using the Blowfish algorithm with a key length of 128 bit, a sectorsize of 4 kilobytes and the “detach on last close” option set:

```
geli_swap_flags="-e blowfish -l 128 -s 4096 -d"
```

Please refer to the description of the `onetime` command in the geli(8) manual page for a list of possible options.

18.17.5 Verifying that it Works

Once the system has been rebooted, proper operation of the encrypted swap can be verified using the `swapinfo` command.

If gbde(8) is being used:

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ad0s1b.bde    542720        0   542720     0%
```

If geli(8) is being used:

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ad0s1b.eli    542720        0   542720     0%
```

Notes

1. For tips on how to select a secure passphrase that is easy to remember, see the Diceware Passphrase (http://world.std.com/~reinhold/diceware.html) website.

Chapter 19
GEOM: Modular Disk Transformation Framework

19.1 Synopsis

This chapter covers the use of disks under the GEOM framework in FreeBSD. This includes the major RAID control utilities which use the framework for configuration. This chapter will not go into in depth discussion on how GEOM handles or controls I/O, the underlying subsystem, or code. This information is provided through the geom(4) manual page and its various SEE ALSO references. This chapter is also not a definitive guide to RAID configurations. Only GEOM-supported RAID classifications will be discussed.

After reading this chapter, you will know:

- What type of RAID support is available through GEOM.
- How to use the base utilities to configure, maintain, and manipulate the various RAID levels.
- How to mirror, stripe, encrypt, and remotely connect disk devices through GEOM.
- How to troubleshoot disks attached to the GEOM framework.

Before reading this chapter, you should:

- Understand how FreeBSD treats disk devices (Chapter 18).
- Know how to configure and install a new FreeBSD kernel (Chapter 8).

19.2 GEOM Introduction

GEOM permits access and control to classes — Master Boot Records, BSD labels, etc — through the use of providers, or the special files in `/dev`. Supporting various software RAID configurations, GEOM will transparently provide access to the operating system and operating system utilities.

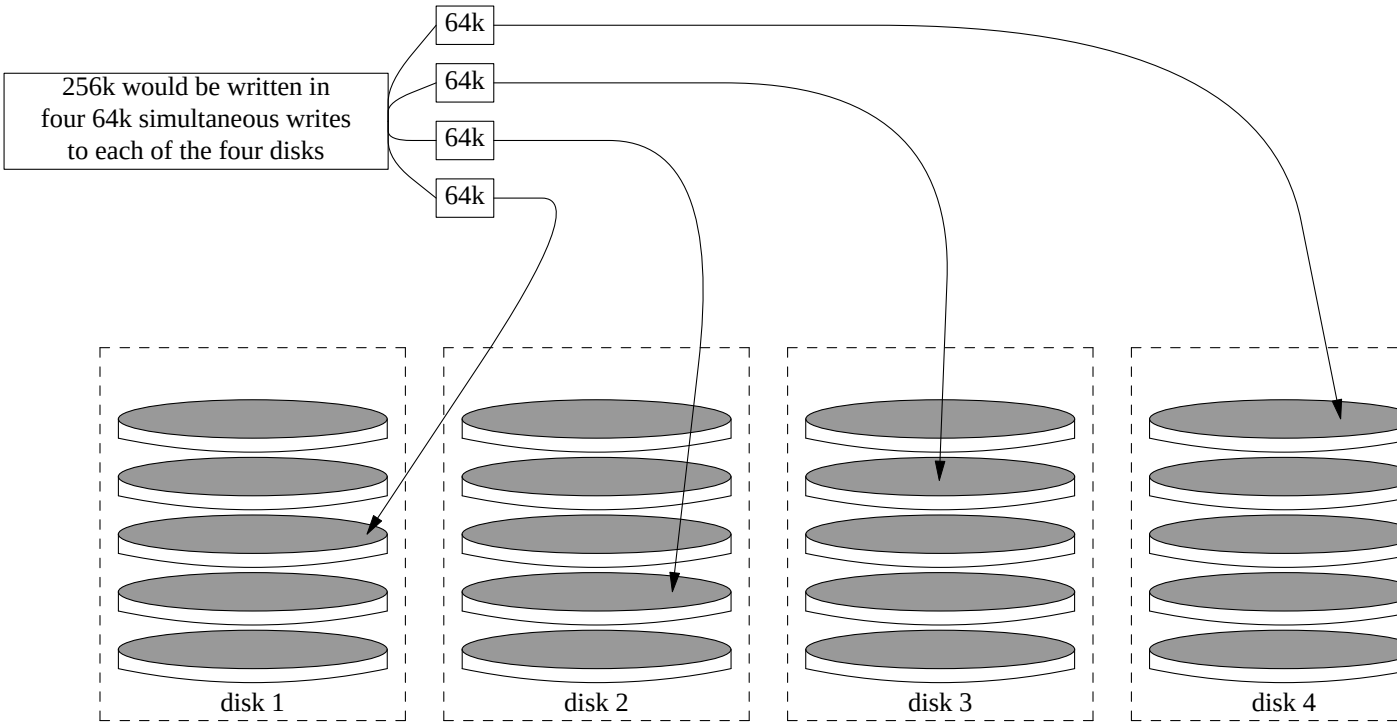
19.3 RAID0 - Striping

Striping is a method used to combine several disk drives into a single volume. In many cases, this is done through the use of hardware controllers. The GEOM disk subsystem provides software support for RAID0, also known as disk striping.

In a RAID0 system, data are split up in blocks that get written across all the drives in the array. Instead of having to wait on the system to write 256k to one disk, a RAID0 system can simultaneously write 64k to each of four different

disks, offering superior I/O performance. This performance can be enhanced further by using multiple disk controllers.

Each disk in a RAID0 stripe must be of the same size, since I/O requests are interleaved to read or write to multiple disks in parallel.



Creating a stripe of unformatted ATA disks

1. Load the `geom_stripe.ko` module:

   ```
   # kldload geom_stripe
   ```

2. Ensure that a suitable mount point exists. If this volume will become a root partition, then temporarily use another mount point such as `/mnt`:

   ```
   # mkdir /mnt
   ```

3. Determine the device names for the disks which will be striped, and create the new stripe device. For example, to stripe two unused and unpartitioned ATA disks, for example `/dev/ad2` and `/dev/ad3`:

   ```
   # gstripe label -v st0 /dev/ad2 /dev/ad3
   Metadata value stored on /dev/ad2.
   Metadata value stored on /dev/ad3.
   Done.
   ```

4. Write a standard label, also known as a partition table, on the new volume and install the default bootstrap code:

   ```
   # bsdlabel -wB /dev/stripe/st0
   ```

5. This process should have created two other devices in the `/dev/stripe` directory in addition to the `st0` device. Those include `st0a` and `st0c`. At this point a file system may be created on the `st0a` device with the `newfs` utility:

```
# newfs -U /dev/stripe/st0a
```

Many numbers will glide across the screen, and after a few seconds, the process will be complete. The volume has been created and is ready to be mounted.

To manually mount the created disk stripe:

```
# mount /dev/stripe/st0a /mnt
```

To mount this striped file system automatically during the boot process, place the volume information in `/etc/fstab` file. For this purpose, a permanent mount point, named `stripe`, is created:

```
# mkdir /stripe
# echo "/dev/stripe/st0a /stripe ufs rw 2 2" \
    >> /etc/fstab
```

The `geom_stripe.ko` module must also be automatically loaded during system initialization, by adding a line to `/boot/loader.conf`:

```
# echo 'geom_stripe_load="YES"' >> /boot/loader.conf
```

19.4 RAID1 - Mirroring

Mirroring is a technology used by many corporations and home users to back up data without interruption. When a mirror exists, it simply means that diskB replicates diskA. Or, perhaps diskC+D replicates diskA+B. Regardless of the disk configuration, the important aspect is that information on one disk or partition is being replicated. Later, that information could be more easily restored, backed up without causing service or access interruption, and even be physically stored in a data safe.

To begin, ensure the system has two disk drives of equal size, these exercises assume they are direct access (da(4)) SCSI disks.

19.4.1 Mirroring Primary Disks

Assuming FreeBSD has been installed on the first, `da0` disk device, gmirror(8) should be told to store its primary data there.

Before building the mirror, enable additional debugging information and opening access to the device by setting the `kern.geom.debugflags` sysctl(8) option to the following value:

```
# sysctl kern.geom.debugflags=17
```

Now create the mirror. Begin the process by storing meta-data information on the primary disk device, effectively creating the `/dev/mirror/gm` device using the following command:

> **Warning:** Creating a mirror out of the boot drive may result in data loss if any data has been stored on the last sector of the disk. This risk is reduced if creating the mirror is done promptly after a fresh install of FreeBSD.

```
# gmirror label -vb round-robin gm0 /dev/da0
```

The system should respond with:

```
Metadata value stored on /dev/da0.
Done.
```

Initialize GEOM, this will load the `/boot/kernel/geom_mirror.ko` kernel module:

```
# gmirror load
```

> **Note:** When this command completes successfully, it creates the `gm0` device node under the `/dev/mirror` directory.

Enable loading of the `geom_mirror.ko` kernel module during system initialization:

```
# echo 'geom_mirror_load="YES"' >> /boot/loader.conf
```

Edit the `/etc/fstab` file, replacing references to the old `da0` with the new device nodes of the `gm0` mirror device.

> **Note:** If vi(1) is your preferred editor, the following is an easy way to accomplish this task:
>
> ```
> # vi /etc/fstab
> ```
>
> In vi(1) back up the current contents of `fstab` by typing **`:w /etc/fstab.bak`**. Then replace all old `da0` references with `gm0` by typing **`:%s/da/mirror\/gm/g`**.

The resulting `fstab` file should look similar to the following. It does not matter if the disk drives are SCSI or ATA, the RAID device will be `gm` regardless.

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/mirror/gm0s1b              none            swap    sw              0       0
/dev/mirror/gm0s1a              /               ufs     rw              1       1
/dev/mirror/gm0s1d    /usr  ufs rw  0 0
/dev/mirror/gm0s1f    /home  ufs rw  2 2
#/dev/mirror/gm0s2d             /store          ufs     rw              2       2
/dev/mirror/gm0s1e              /var            ufs     rw              2       2
/dev/acd0               /cdrom          cd9660  ro,noauto       0       0
```

Reboot the system:

```
# shutdown -r now
```

During system initialization, the `gm0` should be used in place of the `da0` device. Once fully initialized, this may be checked by visually inspecting the output from the `mount` command:

```
# mount
Filesystem         1K-blocks    Used    Avail Capacity  Mounted on
/dev/mirror/gm0s1a   1012974  224604   707334    24%    /
devfs                      1       1        0   100%    /dev
/dev/mirror/gm0s1f  45970182   28596 42263972     0%    /home
/dev/mirror/gm0s1d   6090094 1348356  4254532    24%    /usr
```

```
/dev/mirror/gm0s1e   3045006 2241420   559986    80%    /var
devfs                      1       1        0   100%    /var/named/dev
```

The output looks good, as expected. Finally, to begin synchronization, insert the `da1` disk into the mirror using the following command:

```
# gmirror insert gm0 /dev/da1
```

As the mirror is built the status may be checked using the following command:

```
# gmirror status
```

Once the mirror has been built and all current data has been synchronized, the output from the above command should look like:

```
      Name    Status  Components
mirror/gm0  COMPLETE  da0
                      da1
```

If there are any issues, or the mirror is still completing the build process, the example will show `DEGRADED` in place of `COMPLETE`.

19.4.2 Troubleshooting

19.4.2.1 System refuses to boot

If the system boots up to a prompt similar to:

```
ffs_mountroot: can't find rootvp
Root mount failed: 6
mountroot>
```

Reboot the machine using the power or reset button. At the boot menu, select option six (6). This will drop the system to a loader(8) prompt. Load the kernel module manually:

```
OK? load geom_mirror
OK? boot
```

If this works then for whatever reason the module was not being loaded properly. Check whether the relevant entry in `/boot/loader.conf` is correct. If the problem persists, place:

```
options GEOM_MIRROR
```

in the kernel configuration file, rebuild and reinstall. That should remedy this issue.

19.4.3 Recovering From Disk Failure

The wonderful part about disk mirroring is that when a disk fails, it may be replaced, presumably, without losing any data.

Considering the previous RAID1 configuration, assume that `da1` has failed and now needs to be replaced. To replace it, determine which disk has failed and power down the system. At this point, the disk may be swapped with a new one and the system brought back up. After the system has restarted, the following commands may be used to replace the disk:

```
# gmirror forget gm0

# gmirror insert gm0 /dev/da1
```

Use the `gmirror status` command to monitor the progress of the rebuild. It is that simple.

19.5 GEOM Gate Network Devices

GEOM supports the remote use of devices, such as disks, CD-ROMs, files, etc. through the use of the gate utilities. This is similar to NFS.

To begin, an exports file must be created. This file specifies who is permitted to access the exported resources and what level of access they are offered. For example, to export the fourth slice on the first SCSI disk, the following `/etc/gg.exports` is more than adequate:

```
192.168.1.0/24 RW /dev/da0s4d
```

It will allow all hosts inside the private network access the file system on the `da0s4d` partition.

To export this device, ensure it is not currently mounted, and start the ggated(8) server daemon:

```
# ggated
```

Now to `mount` the device on the client machine, issue the following commands:

```
# ggatec create -o rw 192.168.1.1 /dev/da0s4d
ggate0
# mount /dev/ggate0 /mnt
```

From here on, the device may be accessed through the `/mnt` mount point.

> **Note:** It should be pointed out that this will fail if the device is currently mounted on either the server machine or any other machine on the network.

When the device is no longer needed, it may be safely unmounted with the umount(8) command, similar to any other disk device.

19.6 Labeling Disk Devices

During system initialization, the FreeBSD kernel will create device nodes as devices are found. This method of probing for devices raises some issues, for instance what if a new disk device is added via USB? It is very likely that a flash device may be handed the device name of `da0` and the original `da0` shifted to `da1`. This will cause issues mounting file systems if they are listed in `/etc/fstab`, effectively, this may also prevent the system from booting.

One solution to this issue is to chain the SCSI devices in order so a new device added to the SCSI card will be issued unused device numbers. But what about USB devices which may replace the primary SCSI disk? This happens because USB devices are usually probed before the SCSI card. One solution is to only insert these devices after the system has been booted. Another method could be to use only a single ATA drive and never list the SCSI devices in `/etc/fstab`.

A better solution is available. By using the `glabel` utility, an administrator or user may label their disk devices and use these labels in `/etc/fstab`. Because `glabel` stores the label in the last sector of a given provider, the label will remain persistent across reboots. By using this label as a device, the file system may always be mounted regardless of what device node it is accessed through.

> **Note:** This goes without saying that a label be permanent. The `glabel` utility may be used to create both a transient and permanent label. Only the permanent label will remain consistent across reboots. See the glabel(8) manual page for more information on the differences between labels.

19.6.1 Label Types and Examples

There are two types of labels, a generic label and a file system label. Labels can be permanent or temporary. Permanent labels can be created with the tunefs(8) or newfs(8) commands. They will then be created in a sub-directory of `/dev`, which will be named according to their file system type. For example, UFS2 file system labels will be created in the `/dev/ufs` directory. Permanent labels can also be created with the `glabel label` command. These are not file system specific, and will be created in the `/dev/label` directory.

A temporary label will go away with the next reboot. These labels will be created in the `/dev/label` directory and are perfect for experimentation. A temporary label can be created using the `glabel create` command. For more information, please read the manual page of glabel(8).

To create a permanent label for a UFS2 file system without destroying any data, issue the following command:

```
# tunefs -L home /dev/da3
```

> **Warning:** If the file system is full, this may cause data corruption; however, if the file system is full then the main goal should be removing stale files and not adding labels.

A label should now exist in `/dev/ufs` which may be added to `/etc/fstab`:

```
/dev/ufs/home  /home            ufs     rw              2      2
```

> **Note:** The file system must not be mounted while attempting to run `tunefs`.

Now the file system may be mounted like normal:

```
# mount /home
```

From this point on, so long as the `geom_label.ko` kernel module is loaded at boot with `/boot/loader.conf` or the `GEOM_LABEL` kernel option is present, the device node may change without any ill effect on the system.

File systems may also be created with a default label by using the `-L` flag with `newfs`. See the newfs(8) manual page for more information.

The following command can be used to destroy the label:

```
# glabel destroy home
```

The following example shows how to label the partitions of a boot disk.

Example 19-1. Labeling Partitions on the Boot Disk

By permanently labeling the partitions on the boot disk, the system should be able to continue to boot normally, even if the disk is moved to another controller or transferred to a different system. For this example, it is assumed that a single ATA disk is used, which is currently recognized by the system as `ad0`. It is also assumed that the standard FreeBSD partition scheme is used, with `/`, `/var`, `/usr` and `/tmp` file systems, as well as a swap partition.

Reboot the system, and at the loader(8) prompt, press **4** to boot into single user mode. Then enter the following commands:

```
# 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
```

The system will continue with multi-user boot. After the boot completes, edit `/etc/fstab` and replace the conventional device names, with their respective labels. The final `/etc/fstab` file will look like the following:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/label/swap         none            swap    sw              0       0
/dev/label/rootfs       /               ufs     rw              1       1
/dev/label/tmp          /tmp            ufs     rw              2       2
/dev/label/usr          /usr            ufs     rw              2       2
/dev/label/var          /var            ufs     rw              2       2
```

The system can now be rebooted. If everything went well, it will come up normally and `mount` will show:

```
# 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)
```

Starting with FreeBSD 7.2, the glabel(8) class supports a new label type for UFS file systems, based on the unique file system id, `ufsid`. These labels may be found in the `/dev/ufsid` directory and are created automatically during system startup. It is possible to use `ufsid` labels to mount partitions using the `/etc/fstab` facility. Use the `glabel status` command to receive a list of file systems and their corresponding `ufsid` labels:

```
% glabel status
```

```
                  Name  Status  Components
ufsid/486b6fc38d330916     N/A  ad4s1d
ufsid/486b6fc16926168e     N/A  ad4s1f
```

In the above example `ad4s1d` represents the `/var` file system, while `ad4s1f` represents the `/usr` file system. Using the `ufsid` values shown, these partitions may now be mounted with the following entries in `/etc/fstab`:

```
/dev/ufsid/486b6fc38d330916        /var        ufs        rw        2      2
/dev/ufsid/486b6fc16926168e        /usr        ufs        rw        2      2
```

Any partitions with `ufsid` labels can be mounted in this way, eliminating the need to create permanent labels for them manually, while still enjoying the benefits of device-name independent mounting.

19.7 UFS Journaling Through GEOM

With the release of FreeBSD 7.0, the long awaited feature of journals has been implemented. The implementation itself is provided through the GEOM subsystem and is easily configured via the gjournal(8) utility.

What is journaling? Journaling capability stores a log of file system transactions, i.e.: changes that make up a complete disk write operation, before meta-data and file writes are committed to the disk proper. This transaction log can later be replayed to redo file system transactions, preventing file system inconsistencies.

This method is yet another mechanism to protect against data loss and inconsistencies of the file system. Unlike Soft Updates which tracks and enforces meta-data updates and Snapshots which is an image of the file system, an actual log is stored in disk space specifically reserved for this task, and in some cases may be stored on another disk entirely.

Unlike other file system journaling implementations, the `gjournal` method is block based and not implemented as part of the file system - only as a GEOM extension.

To enable support for `gjournal`, the FreeBSD kernel must have the following option - which is the default on FreeBSD 7.0 and later systems:

```
options UFS_GJOURNAL
```

If journaled volumes need to be mounted during startup, the `geom_journal.ko` kernel module will also have to be loaded, by adding the following line in `/boot/loader.conf`:

```
geom_journal_load="YES"
```

Alternatively, this function can also be built into a custom kernel, by adding the following line in the kernel configuration file:

```
options GEOM_JOURNAL
```

Creating a journal on a free file system may now be done using the following steps, considering that the `da4` is a new SCSI disk:

```
# gjournal load
# gjournal label /dev/da4
```

At this point, there should be a `/dev/da4` device node and a `/dev/da4.journal` device node. A file system may now be created on this device:

```
# newfs -O 2 -J /dev/da4.journal
```

The previously issued command will create a UFS2 file system on the journaled device.

Effectively `mount` the device at the desired point with:

```
# mount /dev/da4.journal /mnt
```

> **Note:** In the case of several slices, a journal will be created for each individual slice. For instance, if `ad4s1` and `ad4s2` are both slices, then `gjournal` will create `ad4s1.journal` and `ad4s2.journal`.

For better performance, keeping the journal on another disk may be desired. For these cases, the journal provider or storage device should be listed after the device to enable journaling on. Journaling may also be enabled on current file systems by using `tunefs`; however, always make a backup before attempting to alter a file system. In most cases, the `gjournal` will fail if it is unable to create the actual journal but this does not protect against data loss incurred as a result of misusing `tunefs`.

It is also possible to journal the boot disk of a FreeBSD system. Please refer to the article Implementing UFS Journaling on a Desktop PC (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/gjournal-desktop) for detailed instructions on this task.

Chapter 20
File Systems Support

20.1 Synopsis

File systems are an integral part of any operating system. They allow for users to upload and store files, provide access to data, and of course, make hard drives useful. Different operating systems usually have one major aspect in common, that is their native file system. On FreeBSD this file system is known as the Fast File System or FFS which is built on the original Unix™ File System, also known as UFS. This is the native file system on FreeBSD which is placed on hard disks for access to data.

FreeBSD also supports a multitude of different file systems to provide support for accessing data from other operating systems locally, i.e. data stored on locally attached USB storage devices, flash drives, and hard disks. There is also support for some non-native file systems. These are file systems developed on other operating systems, like the Linux Extended File System (EXT), and the Sun Z File System (ZFS).

There are different levels of support for the various file systems in FreeBSD. Some will require a kernel module to be loaded, others may require a toolset to be installed. This chapter is designed to help users of FreeBSD access other file systems on their systems, starting with the Sun Z file system.

After reading this chapter, you will know:

- The difference between native and supported file systems.
- What file systems are supported by FreeBSD.
- How to enable, configure, access and make use of non-native file systems.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Chapter 3).
- Be familiar with the basics of kernel configuration/compilation (Chapter 8).
- Feel comfortable installing third party software in FreeBSD (Chapter 4).
- Have some familiarity with disks, storage and device names in FreeBSD (Chapter 18).

20.2 The Z File System (ZFS)

The Z file system, developed by Sun, is a new technology designed to use a pooled storage method. This means that space is only used as it is needed for data storage. It has also been designed for maximum data integrity, supporting data snapshots, multiple copies, and data checksums. A new data replication model, known as RAID-Z has been added. The RAID-Z model is similar to RAID5 but is designed to prevent data write corruption.

20.2.1 ZFS Tuning

The ZFS subsystem utilizes much of the system resources, so some tuning may be required to provide maximum efficiency during every-day use. As an experimental feature in FreeBSD this may change in the near future; however, at this time, the following steps are recommended.

20.2.1.1 Memory

The total system memory should be at least one gigabyte, with two gigabytes or more recommended. In all of the examples here, the system has one gigabyte of memory with several other tuning mechanisms in place.

Some people have had luck using fewer than one gigabyte of memory, but with such a limited amount of physical memory, when the system is under heavy load, it is very plausible that FreeBSD will panic due to memory exhaustion.

20.2.1.2 Kernel Configuration

It is recommended that unused drivers and options be removed from the kernel configuration file. Since most devices are available as modules, they may simply be loaded using the `/boot/loader.conf` file.

Users of the i386 architecture should add the following option to their kernel configuration file, rebuild their kernel, and reboot:

```
options  KVA_PAGES=512
```

This option will expand the kernel address space, thus allowing the `vm.kvm_size` tunable to be pushed beyond the currently imposed limit of 1 GB (2 GB for PAE). To find the most suitable value for this option, divide the desired address space in megabytes by four (4). In this case, it is `512` for 2 GB.

20.2.1.3 Loader Tunables

The `kmem` address space should be increased on all FreeBSD architectures. On the test system with one gigabyte of physical memory, success was achieved with the following options which should be placed in the `/boot/loader.conf` file and the system restarted:

```
vm.kmem_size="330M"
vm.kmem_size_max="330M"
vfs.zfs.arc_max="40M"
vfs.zfs.vdev.cache.size="5M"
```

For a more detailed list of recommendations for ZFS-related tuning, see http://wiki.freebsd.org/ZFSTuningGuide.

20.2.2 Using ZFS

There is a start up mechanism that allows FreeBSD to mount ZFS pools during system initialization. To set it, issue the following commands:

```
# echo 'zfs_enable="YES"' >> /etc/rc.conf
# /etc/rc.d/zfs start
```

The remainder of this document assumes three SCSI disks are available, and their device names are *da0*, *da1* and *da2*. Users of IDE hardware may use the *ad* devices in place of SCSI hardware.

20.2.2.1 Single Disk Pool

To create a simple, non-redundant ZFS pool using a single disk device, use the `zpool` command:

```
# zpool create example /dev/da0
```

To view the new pool, review the output of the `df`:

```
# 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
example      17547136       0 17547136     0%    /example
```

This output clearly shows the `example` pool has not only been created but *mounted* as well. It is also accessible just like a normal file system, files may be created on it and users are able to browse it as in the following example:

```
# cd /example
# ls
# touch testfile
# 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 testfile
```

Unfortunately this pool is not taking advantage of any ZFS features. Create a file system on this pool, and enable compression on it:

```
# zfs create example/compressed
# zfs set compression=gzip example/compressed
```

The `example/compressed` is now a ZFS compressed file system. Try copying some large files to it by copying them to `/example/compressed`.

The compression may now be disabled with:

```
# zfs set compression=off example/compressed
```

To unmount the file system, issue the following command and then verify by using the `df` utility:

```
# zfs umount example/compressed
# 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
example      17547008       0 17547008     0%    /example
```

Re-mount the file system to make it accessible again, and verify with `df`:

```
# zfs mount example/compressed
# 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
example             17547008       0 17547008     0%    /example
example/compressed  17547008       0 17547008     0%    /example/compressed
```

The pool and file system may also be observed by viewing the output from `mount`:

```
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
example on /example (zfs, local)
example/data on /example/data (zfs, local)
example/compressed on /example/compressed (zfs, local)
```

As observed, ZFS file systems, after creation, may be used like ordinary file systems; however, many other features are also available. In the following example, a new file system, `data` is created. Important files will be stored here, so the file system is set to keep two copies of each data block:

```
# zfs create example/data
# zfs set copies=2 example/data
```

It is now possible to see the data and space utilization by issuing the `df` again:

```
# 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
example             17547008       0 17547008     0%    /example
example/compressed  17547008       0 17547008     0%    /example/compressed
example/data        17547008       0 17547008     0%    /example/data
```

Notice that each file system on the pool has the same amount of available space. This is the reason for using the `df` through these examples, to show that the file systems are using only the amount of space they need and will all draw from the same pool. The ZFS file system does away with concepts such as volumes and partitions, and allows for several file systems to occupy the same pool. Destroy the file systems, and then destroy the pool as they are no longer needed:

```
# zfs destroy example/compressed
# zfs destroy example/data
# zpool destroy example
```

Disks go bad and fail, an unavoidable trait. When this disk goes bad, the data will be lost. One method of avoiding data loss due to a failed hard disk is to implement a RAID. ZFS supports this feature in its pool design which is covered in the next section.

20.2.2.2 ZFS RAID-Z

As previously noted, this section will assume that three SCSI disks exist as devices `da0`, `da1` and `da2` (or `ad0` and beyond in case IDE disks are being used). To create a RAID-Z pool, issue the following command:

```
# zpool create storage raidz da0 da1 da2
```

> **Note:** Sun recommends that the amount of devices used in a RAID-Z configuration is between three and nine. If your needs call for a single pool to consist of 10 disks or more, consider breaking it up into smaller RAID-Z groups. If you only have two disks and still require redundancy, consider using a ZFS mirror instead. See the zpool(8) manual page for more details.

The `storage` zpool should have been created. This may be verified by using the mount(8) and df(1) commands as before. More disk devices may have been allocated by adding them to the end of the list above. Make a new file system in the pool, called `home` where user files will eventually be placed:

```
# zfs create storage/home
```

It is now possible to enable compression and keep extra copies of the user's home directories and files. This may be accomplished just as before using the following commands:

```
# zfs set copies=2 storage/home
# zfs set compression=gzip storage/home
```

To make this the new home directory for users, copy the user data to this directory, and create the appropriate symbolic links:

```
# cp -rp /home/* /storage/home
# rm -rf /home /usr/home
# ln -s /storage/home /home
# ln -s /storage/home /usr/home
```

Users should now have their data stored on the freshly created `/storage/home` file system. Test by adding a new user and logging in as that user.

Try creating a snapshot which may be rolled back later:

```
# zfs snapshot storage/home@08-30-08
```

Note that the snapshot option will only capture a real file system, not a home directory or a file. The `@` character is a delimiter used between the file system name or the volume name. When a user's home directory gets trashed, restore it with:

```
# zfs rollback storage/home@08-30-08
```

To get a list of all available snapshots, run the `ls` in the file system's `.zfs/snapshot` directory. For example, to see the previously taken snapshot, perform the following command:

```
# ls /storage/home/.zfs/snapshot
```

It is possible to write a script to perform monthly snapshots on user data; however, over time, snapshots may consume a great deal of disk space. The previous snapshot may be removed using the following command:

```
# zfs destroy storage/home@08-30-08
```

There is no reason, after all of this testing, we should keep `/storage/home` around in its present state. Make it the real `/home` file system:

```
# zfs set mountpoint=/home storage/home
```

Issuing the `df` and `mount` commands will show that the system now treats our file system as the real `/home`:

```
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
storage on /storage (zfs, local)
storage/home 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
storage       26320512       0 26320512     0%    /storage
storage/home  26320512       0 26320512     0%    /home
```

This completes the RAID-Z configuration. To get status updates about the file systems created during the nightly periodic(8) runs, issue the following command:

```
# echo 'daily_status_zfs_enable="YES"' >> /etc/periodic.conf
```

20.2.2.3 Recovering RAID-Z

Every software RAID has a method of monitoring their `state`. ZFS is no exception. The status of RAID-Z devices may be viewed with the following command:

```
# zpool status -x
```

If all pools are healthy and everything is normal, the following message will be returned:

```
all pools are healthy
```

If there is an issue, perhaps a disk has gone offline, the pool state will be returned and look similar to:

```
  pool: storage
 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
```

```
 storage     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
```

This states that the device was taken offline by the administrator. This is true for this particular example. To take the disk offline, the following command was used:

```
# zpool offline storage da1
```

It is now possible to replace the `da1` after the system has been powered down. When the system is back online, the following command may issued to replace the disk:

```
# zpool replace storage da1
```

From here, the status may be checked again, this time without the `-x` flag to get state information:

```
# zpool status storage
 pool: storage
 state: ONLINE
 scrub: resilver completed with 0 errors on Sat Aug 30 19:44:11 2008
config:

 NAME        STATE     READ WRITE CKSUM
 storage     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
```

As shown from this example, everything appears to be normal.

20.2.2.4 Data Verification

As previously mentioned, ZFS uses `checksums` to verify the integrity of stored data. They are enabled automatically upon creation of file systems and may be disabled using the following command:

```
# zfs set checksum=off storage/home
```

This is not a wise idea; however, as checksums take very little storage space and are more useful enabled. There also appear to be no noticeable costs having them enabled. While enabled, it is possible to have ZFS check data integrity using checksum verification. This process is known as “scrubbing.” To verify the data integrity of the `storage` pool, issue the following command:

```
# zpool scrub storage
```

This process may take considerable time depending on the amount of data stored. It is also very I/O intensive, so much that only one of these operations may be run at any given time. After the scrub has completed, the status is updated and may be viewed by issuing a status request:

```
# zpool status storage
 pool: storage
 state: ONLINE
 scrub: scrub completed with 0 errors on Sat Aug 30 19:57:37 2008
config:

 NAME        STATE     READ WRITE CKSUM
 storage     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
```

The completion time is in plain view in this example. This feature helps to ensure data integrity over a long period of time.

There are many more options for the Z file system, see the zfs(8) and zpool(8) manual pages.

Chapter 21
The Vinum Volume Manager

21.1 Synopsis

No matter what disks you have, there are always potential problems:

- They can be too small.
- They can be too slow.
- They can be too unreliable.

Various solutions to these problems have been proposed and implemented. One way some users safeguard themselves against such issues is through the use of multiple, and sometimes redundant, disks. In addition to supporting various cards and controllers for hardware RAID systems, the base FreeBSD system includes the Vinum Volume Manager, a block device driver that implements virtual disk drives. *Vinum* is a so-called *Volume Manager*, a virtual disk driver that addresses these three problems. Vinum provides more flexibility, performance, and reliability than traditional disk storage, and implements RAID-0, RAID-1, and RAID-5 models both individually and in combination.

This chapter provides an overview of potential problems with traditional disk storage, and an introduction to the Vinum Volume Manager.

> **Note:** Starting with FreeBSD 5, Vinum has been rewritten in order to fit into the GEOM architecture (Chapter 19), retaining the original ideas, terminology, and on-disk metadata. This rewrite is called *gvinum* (for *GEOM vinum*). The following text usually refers to *Vinum* as an abstract name, regardless of the implementation variant. Any command invocations should now be done using the `gvinum` command, and the name of the kernel module has been changed from `vinum.ko` to `geom_vinum.ko`, and all device nodes reside under `/dev/gvinum` instead of `/dev/vinum`. As of FreeBSD 6, the old Vinum implementation is no longer available in the code base.

21.2 Disks Are Too Small

Disks are getting bigger, but so are data storage requirements. Often you will find you want a file system that is bigger than the disks you have available. Admittedly, this problem is not as acute as it was ten years ago, but it still exists. Some systems have solved this by creating an abstract device which stores its data on a number of disks.

21.3 Access Bottlenecks

Modern systems frequently need to access data in a highly concurrent manner. For example, large FTP or HTTP servers can maintain thousands of concurrent sessions and have multiple 100 Mbit/s connections to the outside world, well beyond the sustained transfer rate of most disks.

Current disk drives can transfer data sequentially at up to 70 MB/s, but this value is of little importance in an environment where many independent processes access a drive, where they may achieve only a fraction of these values. In such cases it is more interesting to view the problem from the viewpoint of the disk subsystem: the important parameter is the load that a transfer places on the subsystem, in other words the time for which a transfer occupies the drives involved in the transfer.

In any disk transfer, the drive must first position the heads, wait for the first sector to pass under the read head, and then perform the transfer. These actions can be considered to be atomic: it does not make any sense to interrupt them.

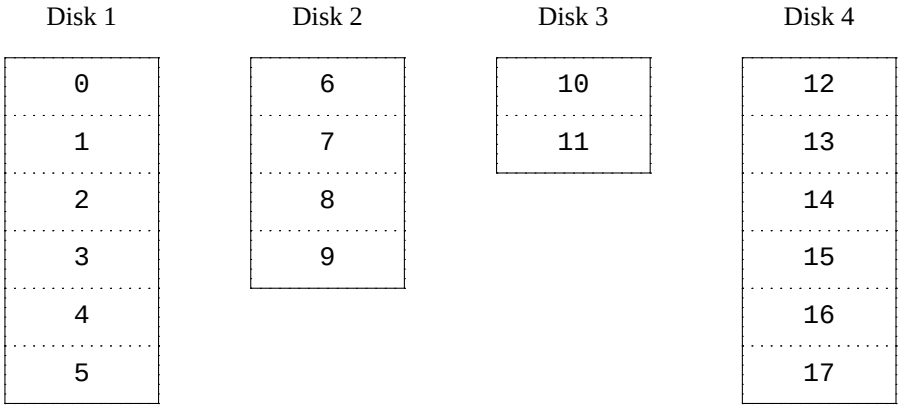
Consider a typical transfer of about 10 kB: the current generation of high-performance disks can position the heads in an average of 3.5 ms. The fastest drives spin at 15,000 rpm, so the average rotational latency (half a revolution) is 2 ms. At 70 MB/s, the transfer itself takes about 150 μs, almost nothing compared to the positioning time. In such a case, the effective transfer rate drops to a little over 1 MB/s and is clearly highly dependent on the transfer size.

The traditional and obvious solution to this bottleneck is “more spindles”: rather than using one large disk, it uses several smaller disks with the same aggregate storage space. Each disk is capable of positioning and transferring independently, so the effective throughput increases by a factor close to the number of disks used.

The exact throughput improvement is, of course, smaller than the number of disks involved: although each drive is capable of transferring in parallel, there is no way to ensure that the requests are evenly distributed across the drives. Inevitably the load on one drive will be higher than on another.

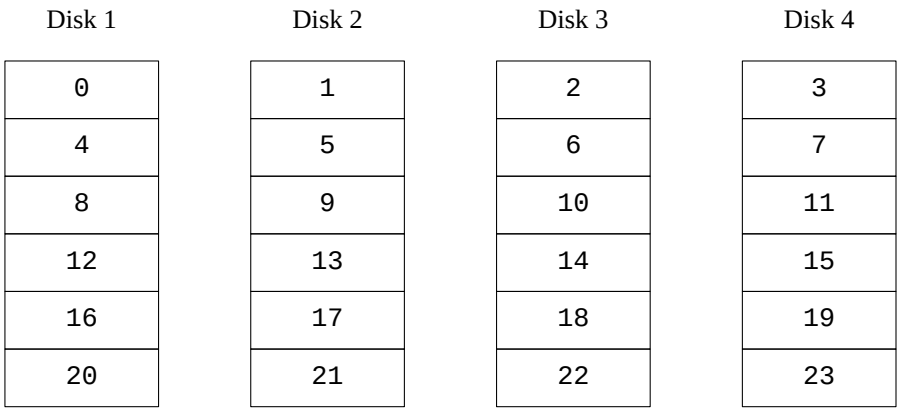
The evenness of the load on the disks is strongly dependent on the way the data is shared across the drives. In the following discussion, it is convenient to think of the disk storage as a large number of data sectors which are addressable by number, rather like the pages in a book. The most obvious method is to divide the virtual disk into groups of consecutive sectors the size of the individual physical disks and store them in this manner, rather like taking a large book and tearing it into smaller sections. This method is called *concatenation* and has the advantage that the disks are not required to have any specific size relationships. It works well when the access to the virtual disk is spread evenly about its address space. When access is concentrated on a smaller area, the improvement is less marked. Figure 21-1 illustrates the sequence in which storage units are allocated in a concatenated organization.

Figure 21-1. Concatenated Organization



An alternative mapping is to divide the address space into smaller, equal-sized components and store them sequentially on different devices. For example, the first 256 sectors may be stored on the first disk, the next 256 sectors on the next disk and so on. After filling the last disk, the process repeats until the disks are full. This mapping is called *striping* or RAID-0 [1]. Striping requires somewhat more effort to locate the data, and it can cause additional I/O load where a transfer is spread over multiple disks, but it can also provide a more constant load across the disks. Figure 21-2 illustrates the sequence in which storage units are allocated in a striped organization.

Figure 21-2. Striped Organization



21.4 Data Integrity

The final problem with current disks is that they are unreliable. Although disk drive reliability has increased tremendously over the last few years, they are still the most likely core component of a server to fail. When they do, the results can be catastrophic: replacing a failed disk drive and restoring data to it can take days.

The traditional way to approach this problem has been *mirroring*, keeping two copies of the data on different physical hardware. Since the advent of the RAID levels, this technique has also been called RAID level 1 or RAID-1. Any write to the volume writes to both locations; a read can be satisfied from either, so if one drive fails, the data is still available on the other drive.

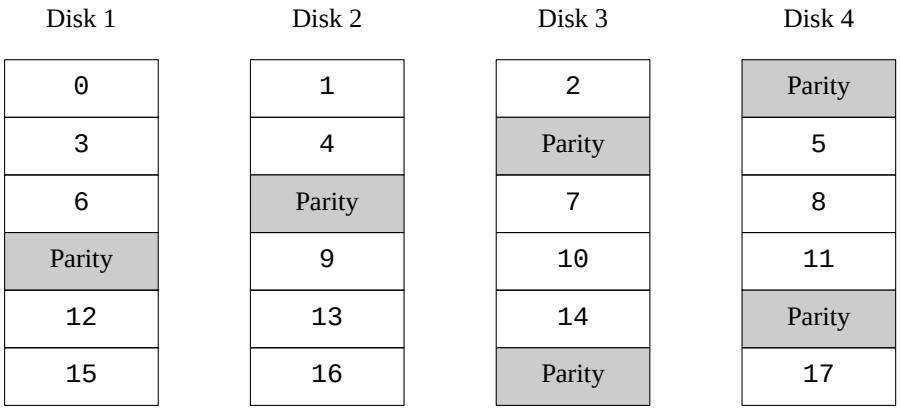
Mirroring has two problems:

- The price. It requires twice as much disk storage as a non-redundant solution.
- The performance impact. Writes must be performed to both drives, so they take up twice the bandwidth of a non-mirrored volume. Reads do not suffer from a performance penalty: it even looks as if they are faster.

An alternative solution is *parity*, implemented in the RAID levels 2, 3, 4 and 5. Of these, RAID-5 is the most interesting. As implemented in Vinum, it is a variant on a striped organization which dedicates one block of each stripe to parity one of the other blocks. As implemented by Vinum, a RAID-5 plex is similar to a striped plex, except

that it implements RAID-5 by including a parity block in each stripe. As required by RAID-5, the location of this parity block changes from one stripe to the next. The numbers in the data blocks indicate the relative block numbers.

Figure 21-3. RAID-5 Organization



Compared to mirroring, RAID-5 has the advantage of requiring significantly less storage space. Read access is similar to that of striped organizations, but write access is significantly slower, approximately 25% of the read performance. If one drive fails, the array can continue to operate in degraded mode: a read from one of the remaining accessible drives continues normally, but a read from the failed drive is recalculated from the corresponding block from all the remaining drives.

21.5 Vinum Objects

In order to address these problems, Vinum implements a four-level hierarchy of objects:

- The most visible object is the virtual disk, called a *volume*. Volumes have essentially the same properties as a UNIX disk drive, though there are some minor differences. They have no size limitations.
- Volumes are composed of *plexes*, each of which represent the total address space of a volume. This level in the hierarchy thus provides redundancy. Think of plexes as individual disks in a mirrored array, each containing the same data.
- Since Vinum exists within the UNIX disk storage framework, it would be possible to use UNIX partitions as the building block for multi-disk plexes, but in fact this turns out to be too inflexible: UNIX disks can have only a limited number of partitions. Instead, Vinum subdivides a single UNIX partition (the *drive*) into contiguous areas called *subdisks*, which it uses as building blocks for plexes.
- Subdisks reside on Vinum *drives*, currently UNIX partitions. Vinum drives can contain any number of subdisks. With the exception of a small area at the beginning of the drive, which is used for storing configuration and state information, the entire drive is available for data storage.

The following sections describe the way these objects provide the functionality required of Vinum.

21.5.1 Volume Size Considerations

Plexes can include multiple subdisks spread over all drives in the Vinum configuration. As a result, the size of an individual drive does not limit the size of a plex, and thus of a volume.

21.5.2 Redundant Data Storage

Vinum implements mirroring by attaching multiple plexes to a volume. Each plex is a representation of the data in a volume. A volume may contain between one and eight plexes.

Although a plex represents the complete data of a volume, it is possible for parts of the representation to be physically missing, either by design (by not defining a subdisk for parts of the plex) or by accident (as a result of the failure of a drive). As long as at least one plex can provide the data for the complete address range of the volume, the volume is fully functional.

21.5.3 Performance Issues

Vinum implements both concatenation and striping at the plex level:

- A *concatenated plex* uses the address space of each subdisk in turn.
- A *striped plex* stripes the data across each subdisk. The subdisks must all have the same size, and there must be at least two subdisks in order to distinguish it from a concatenated plex.

21.5.4 Which Plex Organization?

The version of Vinum supplied with FreeBSD 8.2 implements two kinds of plex:

- Concatenated plexes are the most flexible: they can contain any number of subdisks, and the subdisks may be of different length. The plex may be extended by adding additional subdisks. They require less CPU time than striped plexes, though the difference in CPU overhead is not measurable. On the other hand, they are most susceptible to hot spots, where one disk is very active and others are idle.
- The greatest advantage of striped (RAID-0) plexes is that they reduce hot spots: by choosing an optimum sized stripe (about 256 kB), you can even out the load on the component drives. The disadvantages of this approach are (fractionally) more complex code and restrictions on subdisks: they must be all the same size, and extending a plex by adding new subdisks is so complicated that Vinum currently does not implement it. Vinum imposes an additional, trivial restriction: a striped plex must have at least two subdisks, since otherwise it is indistinguishable from a concatenated plex.

Table 21-1 summarizes the advantages and disadvantages of each plex organization.

Table 21-1. Vinum Plex Organizations

Plex type	Minimum subdisks	Can add subdisks	Must be equal size	Application

Plex type	Minimum subdisks	Can add subdisks	Must be equal size	Application
concatenated	1	yes	no	Large data storage with maximum placement flexibility and moderate performance
striped	2	no	yes	High performance in combination with highly concurrent access

21.6 Some Examples

Vinum maintains a *configuration database* which describes the objects known to an individual system. Initially, the user creates the configuration database from one or more configuration files with the aid of the gvinum(8) utility program. Vinum stores a copy of its configuration database on each disk slice (which Vinum calls a *device*) under its control. This database is updated on each state change, so that a restart accurately restores the state of each Vinum object.

21.6.1 The Configuration File

The configuration file describes individual Vinum objects. The definition of a simple volume might be:

```
drive a device /dev/da3h
volume myvol
  plex org concat
    sd length 512m drive a
```

This file describes four Vinum objects:

- The *drive* line describes a disk partition (*drive*) and its location relative to the underlying hardware. It is given the symbolic name *a*. This separation of the symbolic names from the device names allows disks to be moved from one location to another without confusion.
- The *volume* line describes a volume. The only required attribute is the name, in this case *myvol*.
- The *plex* line defines a plex. The only required parameter is the organization, in this case *concat*. No name is necessary: the system automatically generates a name from the volume name by adding the suffix *.px*, where *x* is the number of the plex in the volume. Thus this plex will be called *myvol.p0*.
- The *sd* line describes a subdisk. The minimum specifications are the name of a drive on which to store it, and the length of the subdisk. As with plexes, no name is necessary: the system automatically assigns names derived from the plex name by adding the suffix *.sx*, where *x* is the number of the subdisk in the plex. Thus Vinum gives this subdisk the name *myvol.p0.s0*.

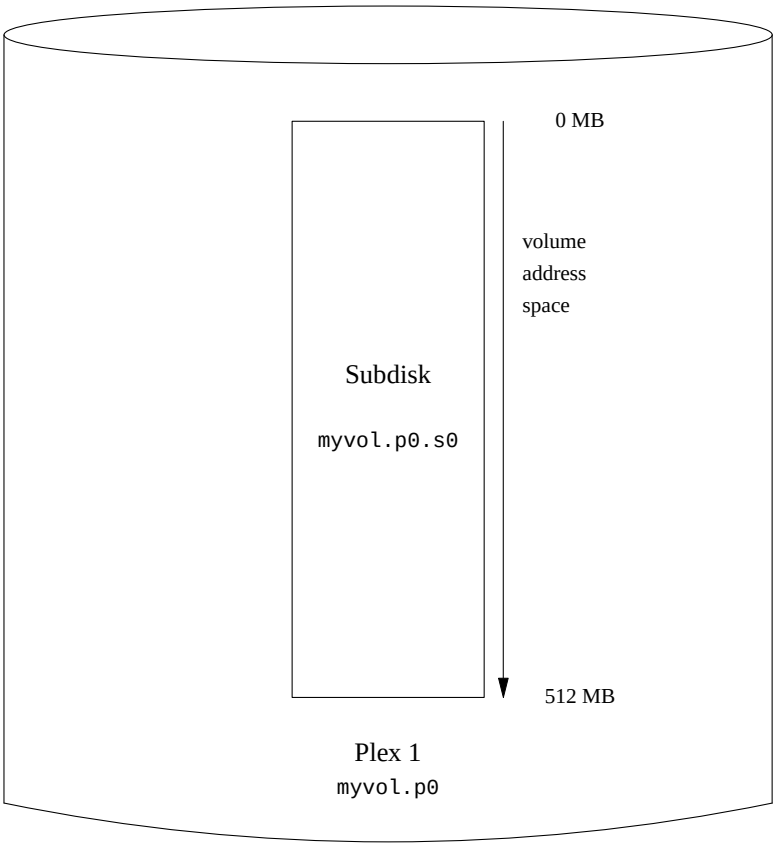
After processing this file, gvinum(8) produces the following output:

```
# 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
```

This output shows the brief listing format of gvinum(8). It is represented graphically in Figure 21-4.

Figure 21-4. A Simple Vinum Volume



This figure, and the ones which follow, represent a volume, which contains the plexes, which in turn contain the subdisks. In this trivial example, the volume contains one plex, and the plex contains one subdisk.

This particular volume has no specific advantage over a conventional disk partition. It contains a single plex, so it is not redundant. The plex contains a single subdisk, so there is no difference in storage allocation from a conventional disk partition. The following sections illustrate various more interesting configuration methods.

21.6.2 Increased Resilience: Mirroring

The resilience of a volume can be increased by mirroring. When laying out a mirrored volume, it is important to ensure that the subdisks of each plex are on different drives, so that a drive failure will not take down both plexes. The following configuration mirrors a volume:

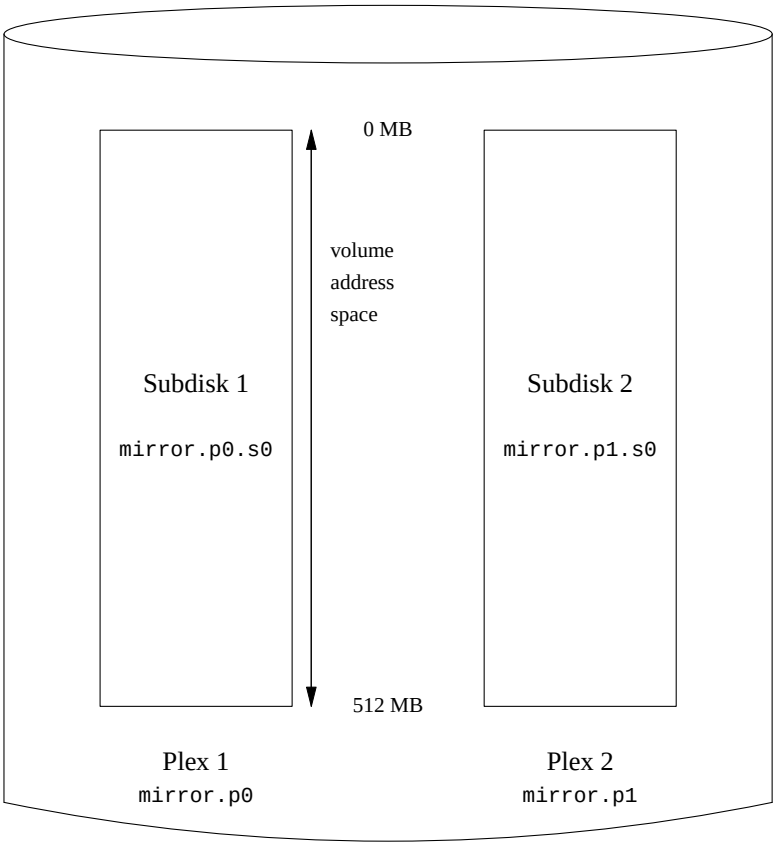
```
drive b device /dev/da4h
volume mirror
     plex org concat
       sd length 512m drive a
  plex org concat
    sd length 512m drive b
```

In this example, it was not necessary to specify a definition of drive *a* again, since Vinum keeps track of all objects in its configuration database. After processing this definition, the configuration looks like:

```
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.p1           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.p1.s0          State: empty    PO:        0  B Size:        512 MB
```

Figure 21-5 shows the structure graphically.

Figure 21-5. A Mirrored Vinum Volume



In this example, each plex contains the full 512 MB of address space. As in the previous example, each plex contains only a single subdisk.

21.6.3 Optimizing Performance

The mirrored volume in the previous example is more resistant to failure than an unmirrored volume, but its performance is less: each write to the volume requires a write to both drives, using up a greater proportion of the total disk bandwidth. Performance considerations demand a different approach: instead of mirroring, the data is striped across as many disk drives as possible. The following configuration shows a volume with a plex striped across four disk drives:

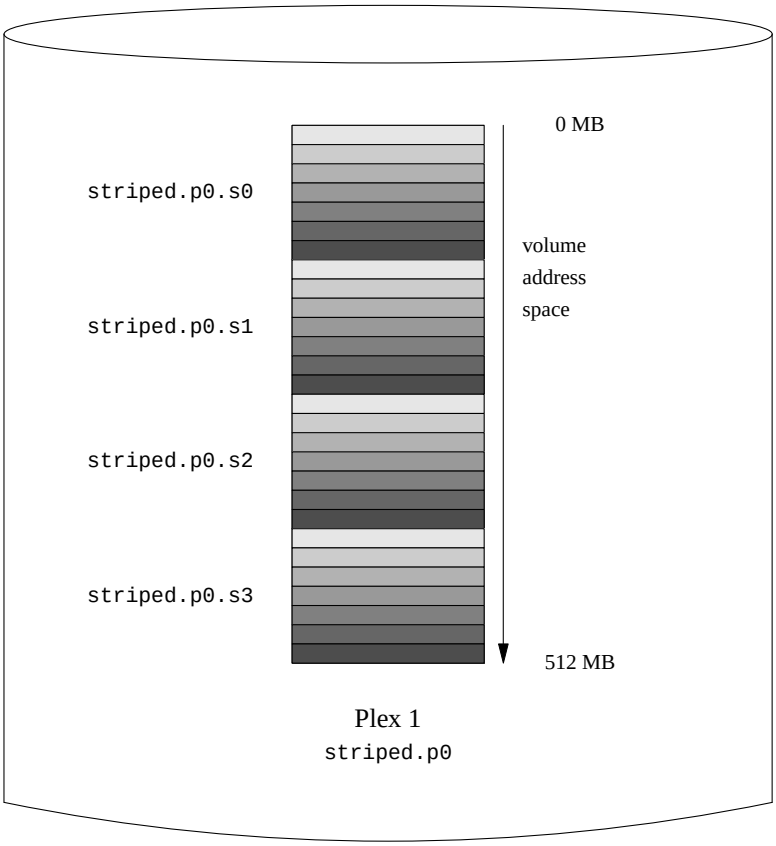
```
drive c device /dev/da5h
drive d device /dev/da6h
volume stripe
plex org striped 512k
  sd length 128m drive a
```

```
sd length 128m drive b
sd length 128m drive c
sd length 128m drive d
```

As before, it is not necessary to define the drives which are already known to Vinum. After processing this definition, the configuration looks like:

```
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
```

Figure 21-6. A Striped Vinum Volume



This volume is represented in Figure 21-6. The darkness of the stripes indicates the position within the plex address space: the lightest stripes come first, the darkest last.

21.6.4 Resilience and Performance

With sufficient hardware, it is possible to build volumes which show both increased resilience and increased performance compared to standard UNIX partitions. A typical configuration file might be:

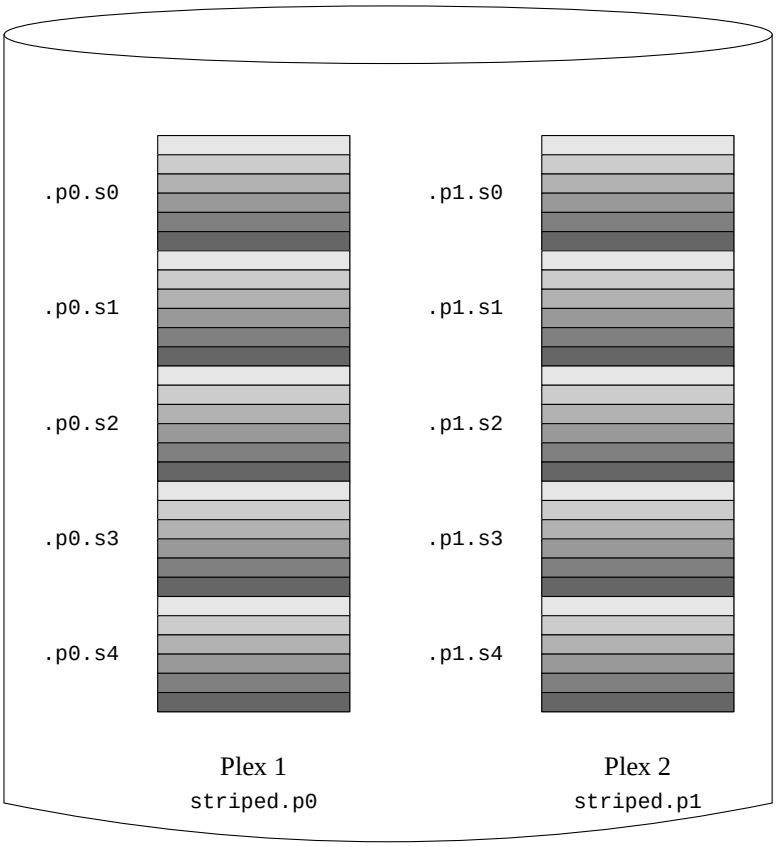
```
volume raid10
     plex org striped 512k
       sd length 102480k drive a
       sd length 102480k drive b
       sd length 102480k drive c
       sd length 102480k drive d
       sd length 102480k drive e
     plex org striped 512k
       sd length 102480k drive c
```

```
sd length 102480k drive d
sd length 102480k drive e
sd length 102480k drive a
sd length 102480k drive b
```

The subdisks of the second plex are offset by two drives from those of the first plex: this helps ensure that writes do not go to the same subdisks even if a transfer goes over two drives.

Figure 21-7 represents the structure of this volume.

Figure 21-7. A Mirrored, Striped Vinum Volume



21.7 Object Naming

As described above, Vinum assigns default names to plexes and subdisks, although they may be overridden. Overriding the default names is not recommended: experience with the VERITAS volume manager, which allows

arbitrary naming of objects, has shown that this flexibility does not bring a significant advantage, and it can cause confusion.

Names may contain any non-blank character, but it is recommended to restrict them to letters, digits and the underscore characters. The names of volumes, plexes and subdisks may be up to 64 characters long, and the names of drives may be up to 32 characters long.

Vinum objects are assigned device nodes in the hierarchy `/dev/gvinum`. The configuration shown above would cause Vinum to create the following device nodes:

- Device entries for each volume. These are the main devices used by Vinum. Thus the configuration above would include the devices `/dev/gvinum/myvol`, `/dev/gvinum/mirror`, `/dev/gvinum/striped`, `/dev/gvinum/raid5` and `/dev/gvinum/raid10`.
- All volumes get direct entries under `/dev/gvinum/`.
- The directories `/dev/gvinum/plex`, and `/dev/gvinum/sd`, which contain device nodes for each plex and for each subdisk, respectively.

For example, consider the following configuration file:

```
drive drive1 device /dev/sd1h
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
```

After processing this file, gvinum(8) creates the following structure in `/dev/gvinum`:

```
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
```

Although it is recommended that plexes and subdisks should not be allocated specific names, Vinum drives must be named. This makes it possible to move a drive to a different location and still recognize it automatically. Drive names may be up to 32 characters long.

21.7.1 Creating File Systems

Volumes appear to the system to be identical to disks, with one exception. Unlike UNIX drives, Vinum does not partition volumes, which thus do not contain a partition table. This has required modification to some disk utilities, notably newfs(8), which previously tried to interpret the last letter of a Vinum volume name as a partition identifier. For example, a disk drive may have a name like `/dev/ad0a` or `/dev/da2h`. These names represent the first partition (`a`) on the first (0) IDE disk (`ad`) and the eighth partition (`h`) on the third (2) SCSI disk (`da`) respectively. By contrast, a Vinum volume might be called `/dev/gvinum/concat`, a name which has no relationship with a partition name.

Normally, newfs(8) interprets the name of the disk and complains if it cannot understand it. For example:

```
# newfs /dev/gvinum/concat
newfs: /dev/gvinum/concat: can’t figure out file system partition
```

In order to create a file system on this volume, use newfs(8):

```
# newfs /dev/gvinum/concat
```

21.8 Configuring Vinum

The `GENERIC` kernel does not contain Vinum. It is possible to build a special kernel which includes Vinum, but this is not recommended. The standard way to start Vinum is as a kernel module (kld). You do not even need to use kldload(8) for Vinum: when you start gvinum(8), it checks whether the module has been loaded, and if it is not, it loads it automatically.

21.8.1 Startup

Vinum stores configuration information on the disk slices in essentially the same form as in the configuration files. When reading from the configuration database, Vinum recognizes a number of keywords which are not allowed in the configuration files. For example, a disk configuration might contain the following text:

```
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
```

The obvious differences here are the presence of explicit location information and naming (both of which are also allowed, but discouraged, for use by the user) and the information on the states (which are not available to the user). Vinum does not store information about drives in the configuration information: it finds the drives by scanning the configured disk drives for partitions with a Vinum label. This enables Vinum to identify drives correctly even if they have been assigned different UNIX drive IDs.

21.8.1.1 Automatic Startup

Gvinum always features an automatic startup once the kernel module is loaded, via loader.conf(5). To load the *Gvinum* module at boot time, add `geom_vinum_load="YES"` to `/boot/loader.conf`.

When you start Vinum with the `gvinum start` command, Vinum reads the configuration database from one of the Vinum drives. Under normal circumstances, each drive contains an identical copy of the configuration database, so it does not matter which drive is read. After a crash, however, Vinum must determine which drive was updated most recently and read the configuration from this drive. It then updates the configuration if necessary from progressively older drives.

21.9 Using Vinum for the Root Filesystem

For a machine that has fully-mirrored filesystems using Vinum, it is desirable to also mirror the root filesystem. Setting up such a configuration is less trivial than mirroring an arbitrary filesystem because:

- The root filesystem must be available very early during the boot process, so the Vinum infrastructure must already be available at this time.
- The volume containing the root filesystem also contains the system bootstrap and the kernel, which must be read using the host system's native utilities (e. g. the BIOS on PC-class machines) which often cannot be taught about the details of Vinum.

In the following sections, the term "root volume" is generally used to describe the Vinum volume that contains the root filesystem. It is probably a good idea to use the name `"root"` for this volume, but this is not technically required in any way. All command examples in the following sections assume this name though.

21.9.1 Starting up Vinum Early Enough for the Root Filesystem

There are several measures to take for this to happen:

- Vinum must be available in the kernel at boot-time. Thus, the method to start Vinum automatically described in Section 21.8.1.1 is not applicable to accomplish this task, and the `start_vinum` parameter must actually *not* be set when the following setup is being arranged. The first option would be to compile Vinum statically into the kernel, so it is available all the time, but this is usually not desirable. There is another option as well, to have `/boot/loader` (Section 12.3.3) load the vinum kernel module early, before starting the kernel. This can be accomplished by putting the line:

  ```
  geom_vinum_load="YES"
  ```

  into the file `/boot/loader.conf`.
- For *Gvinum*, all startup is done automatically once the kernel module has been loaded, so the procedure described above is all that is needed.

21.9.2 Making a Vinum-based Root Volume Accessible to the Bootstrap

Since the current FreeBSD bootstrap is only 7.5 KB of code, and already has the burden of reading files (like `/boot/loader`) from the UFS filesystem, it is sheer impossible to also teach it about internal Vinum structures so it

could parse the Vinum configuration data, and figure out about the elements of a boot volume itself. Thus, some tricks are necessary to provide the bootstrap code with the illusion of a standard `"a"` partition that contains the root filesystem.

For this to be possible at all, the following requirements must be met for the root volume:

- The root volume must not be striped or RAID-5.
- The root volume must not contain more than one concatenated subdisk per plex.

Note that it is desirable and possible that there are multiple plexes, each containing one replica of the root filesystem. The bootstrap process will, however, only use one of these replica for finding the bootstrap and all the files, until the kernel will eventually mount the root filesystem itself. Each single subdisk within these plexes will then need its own `"a"` partition illusion, for the respective device to become bootable. It is not strictly needed that each of these faked `"a"` partitions is located at the same offset within its device, compared with other devices containing plexes of the root volume. However, it is probably a good idea to create the Vinum volumes that way so the resulting mirrored devices are symmetric, to avoid confusion.

In order to set up these `"a"` partitions, for each device containing part of the root volume, the following needs to be done:

1. The location (offset from the beginning of the device) and size of this device's subdisk that is part of the root volume need to be examined, using the command:

   ```
   # gvinum l -rv root
   ```

   Note that Vinum offsets and sizes are measured in bytes. They must be divided by 512 in order to obtain the block numbers that are to be used in the `bsdlabel` command.

2. Run the command:

   ```
   # bsdlabel -e devname
   ```

   for each device that participates in the root volume. *devname* must be either the name of the disk (like `da0`) for disks without a slice (aka. fdisk) table, or the name of the slice (like `ad0s1`).

   If there is already an `"a"` partition on the device (presumably, containing a pre-Vinum root filesystem), it should be renamed to something else, so it remains accessible (just in case), but will no longer be used by default to bootstrap the system. Note that active partitions (like a root filesystem currently mounted) cannot be renamed, so this must be executed either when being booted from a "Fixit" medium, or in a two-step process, where (in a mirrored situation) the disk that has not been currently booted is being manipulated first.

   Then, the offset of the Vinum partition on this device (if any) must be added to the offset of the respective root volume subdisk on this device. The resulting value will become the `"offset"` value for the new `"a"` partition. The `"size"` value for this partition can be taken verbatim from the calculation above. The `"fstype"` should be `4.2BSD`. The `"fsize"`, `"bsize"`, and `"cpg"` values should best be chosen to match the actual filesystem, though they are fairly unimportant within this context.

   That way, a new `"a"` partition will be established that overlaps the Vinum partition on this device. Note that the `bsdlabel` will only allow for this overlap if the Vinum partition has properly been marked using the `"vinum"` fstype.

3. That's all! A faked `"a"` partition does exist now on each device that has one replica of the root volume. It is highly recommendable to verify the result again, using a command like:

   ```
   # fsck -n /dev/devnamea
   ```

It should be remembered that all files containing control information must be relative to the root filesystem in the Vinum volume which, when setting up a new Vinum root volume, might not match the root filesystem that is currently active. So in particular, the files `/etc/fstab` and `/boot/loader.conf` need to be taken care of.

At next reboot, the bootstrap should figure out the appropriate control information from the new Vinum-based root filesystem, and act accordingly. At the end of the kernel initialization process, after all devices have been announced, the prominent notice that shows the success of this setup is a message like:

```
Mounting root from ufs:/dev/gvinum/root
```

21.9.3 Example of a Vinum-based Root Setup

After the Vinum root volume has been set up, the output of `gvinum l -rv root` could look like:

```
...
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/da1h) at offset 135680 (132 kB)
```

The values to note are `135680` for the offset (relative to partition `/dev/da0h`). This translates to 265 512-byte disk blocks in `bsdlabel`'s terms. Likewise, the size of this root volume is 245760 512-byte blocks. `/dev/da1h`, containing the second replica of this root volume, has a symmetric setup.

The bsdlabel for these devices might look like:

```
...
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         # (Cyl.    0 - 4467*)
  h: 71771672       16     vinum                       # (Cyl.    0*- 4467*)
```

It can be observed that the `"size"` parameter for the faked `"a"` partition matches the value outlined above, while the `"offset"` parameter is the sum of the offset within the Vinum partition `"h"`, and the offset of this partition within the device (or slice). This is a typical setup that is necessary to avoid the problem described in Section 21.9.4.3. It can also be seen that the entire `"a"` partition is completely within the `"h"` partition containing all the Vinum data for this device.

Note that in the above example, the entire device is dedicated to Vinum, and there is no leftover pre-Vinum root partition, since this has been a newly set-up disk that was only meant to be part of a Vinum configuration, ever.

21.9.4 Troubleshooting

If something goes wrong, a way is needed to recover from the situation. The following list contains few known pitfalls and solutions.

21.9.4.1 System Bootstrap Loads, but System Does Not Boot

If for any reason the system does not continue to boot, the bootstrap can be interrupted with by pressing the **space** key at the 10-seconds warning. The loader variables (like `vinum.autostart`) can be examined using the `show`, and manipulated using `set` or `unset` commands.

If the only problem was that the Vinum kernel module was not yet in the list of modules to load automatically, a simple `load geom_vinum` will help.

When ready, the boot process can be continued with a `boot -as`. The options `-as` will request the kernel to ask for the root filesystem to mount (`-a`), and make the boot process stop in single-user mode (`-s`), where the root filesystem is mounted read-only. That way, even if only one plex of a multi-plex volume has been mounted, no data inconsistency between plexes is being risked.

At the prompt asking for a root filesystem to mount, any device that contains a valid root filesystem can be entered. If `/etc/fstab` had been set up correctly, the default should be something like `ufs:/dev/gvinum/root`. A typical alternate choice would be something like `ufs:da0d` which could be a hypothetical partition that contains the pre-Vinum root filesystem. Care should be taken if one of the alias `"a"` partitions are entered here that are actually reference to the subdisks of the Vinum root device, because in a mirrored setup, this would only mount one piece of a mirrored root device. If this filesystem is to be mounted read-write later on, it is necessary to remove the other plex(es) of the Vinum root volume since these plexes would otherwise carry inconsistent data.

21.9.4.2 Only Primary Bootstrap Loads

If `/boot/loader` fails to load, but the primary bootstrap still loads (visible by a single dash in the left column of the screen right after the boot process starts), an attempt can be made to interrupt the primary bootstrap at this point, using the **space** key. This will make the bootstrap stop in stage two, see Section 12.3.2. An attempt can be made here to boot off an alternate partition, like the partition containing the previous root filesystem that has been moved away from `"a"` above.

21.9.4.3 Nothing Boots, the Bootstrap Panics

This situation will happen if the bootstrap had been destroyed by the Vinum installation. Unfortunately, Vinum accidentally currently leaves only 4 KB at the beginning of its partition free before starting to write its Vinum header information. However, the stage one and two bootstraps plus the bsdlabel embedded between them currently require 8 KB. So if a Vinum partition was started at offset 0 within a slice or disk that was meant to be bootable, the Vinum setup will trash the bootstrap.

Similarly, if the above situation has been recovered, for example by booting from a “Fixit” medium, and the bootstrap has been re-installed using `bsdlabel -B` as described in Section 12.3.2, the bootstrap will trash the Vinum header, and Vinum will no longer find its disk(s). Though no actual Vinum configuration data or data in Vinum volumes will be trashed by this, and it would be possible to recover all the data by entering exact the same Vinum configuration data again, the situation is hard to fix at all. It would be necessary to move the entire Vinum partition by at least 4 KB off, in order to have the Vinum header and the system bootstrap no longer collide.

Notes

1. RAID stands for *Redundant Array of Inexpensive Disks* and offers various forms of fault tolerance, though the latter term is somewhat misleading: it provides no redundancy.

Chapter 22
Virtualization

22.1 Synopsis

Virtualization software allows multiple operating systems to run simultaneously on the same computer. Such software systems for PCs often involve a host operating system which runs the virtualization software and supports any number of guest operating systems.

After reading this chapter, you will know:

- The difference between a host operating system and a guest operating system.
- How to install FreeBSD on an Intel-based Apple Macintosh computer.
- How to install FreeBSD on Microsoft Windows with **Virtual PC**.
- How to tune a FreeBSD system for best performance under virtualization.

Before reading this chapter, you should:

- Understand the basics of UNIX and FreeBSD (Chapter 3).
- Know how to install FreeBSD (Chapter 2).
- Know how to set up your network connection (Chapter 31).
- Know how to install additional third-party software (Chapter 4).

22.2 FreeBSD as a Guest OS

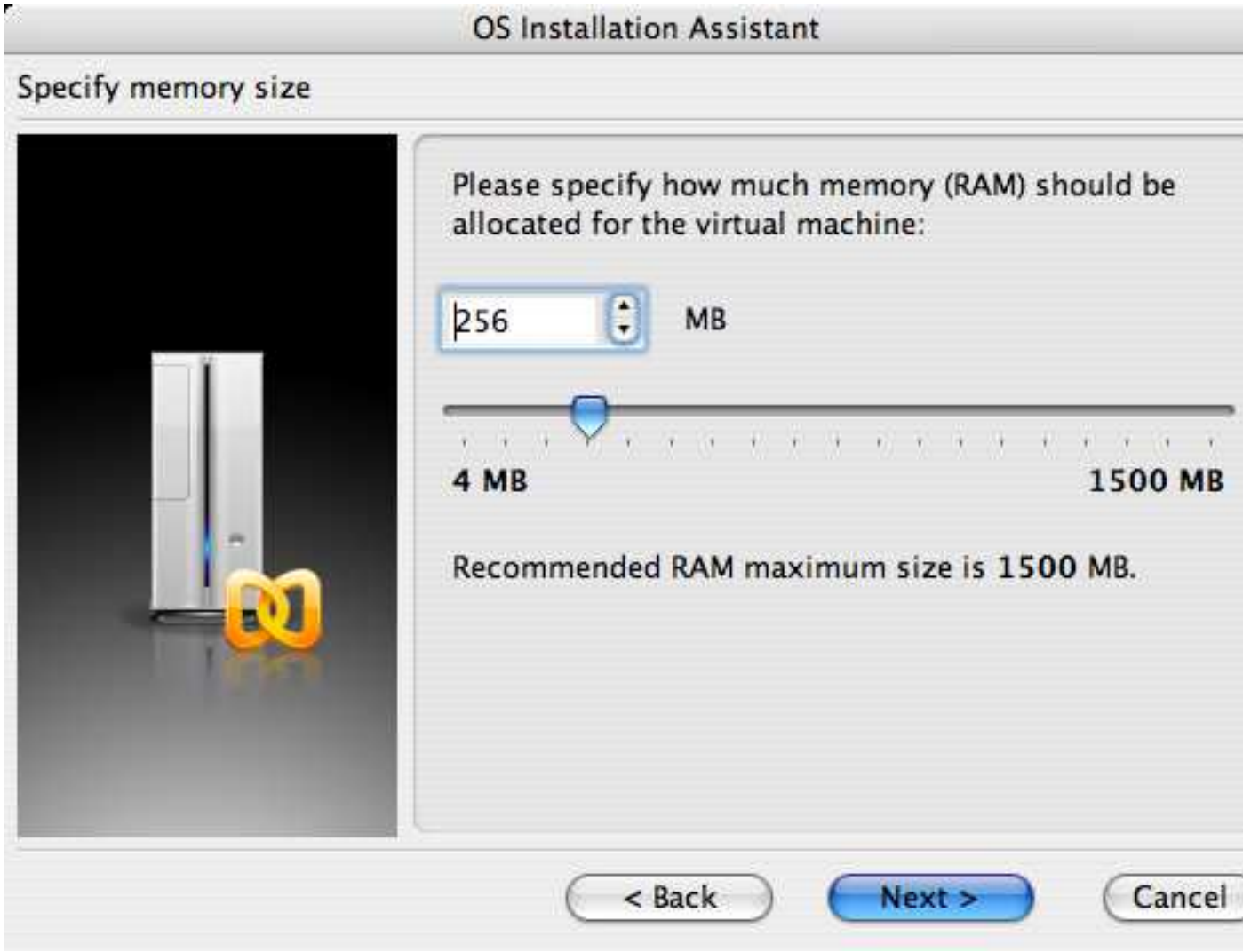
22.2.1 Parallels on MacOS

Parallels Desktop for Mac is a commercial software product available for Intel based Apple Mac computers running Mac OS 10.4.6 or higher. FreeBSD is a fully supported guest operating system. Once **Parallels** has been installed on Mac OS X, the user must configure a virtual machine and then install the desired guest operating system.

22.2.1.1 Installing FreeBSD on Parallels/Mac OS® X

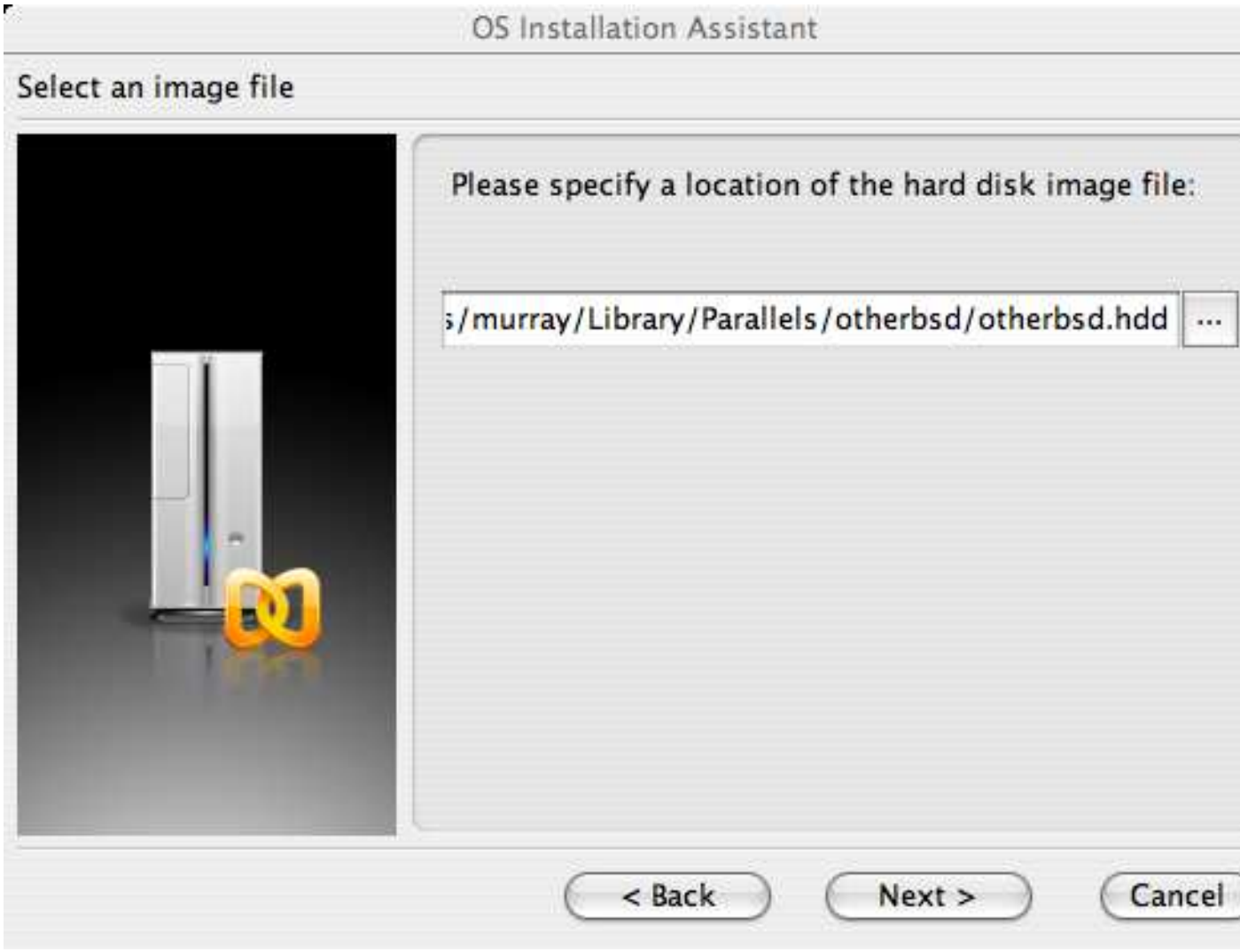
The first step in installing FreeBSD on Mac OS X/**Parallels** is to create a new virtual machine for installing FreeBSD. Select FreeBSD as the Guest OS Type when prompted:

And choose a reasonable amount of disk and memory depending on your plans for this virtual FreeBSD instance. 4GB of disk space and 512MB of RAM work well for most uses of FreeBSD under **Parallels**:

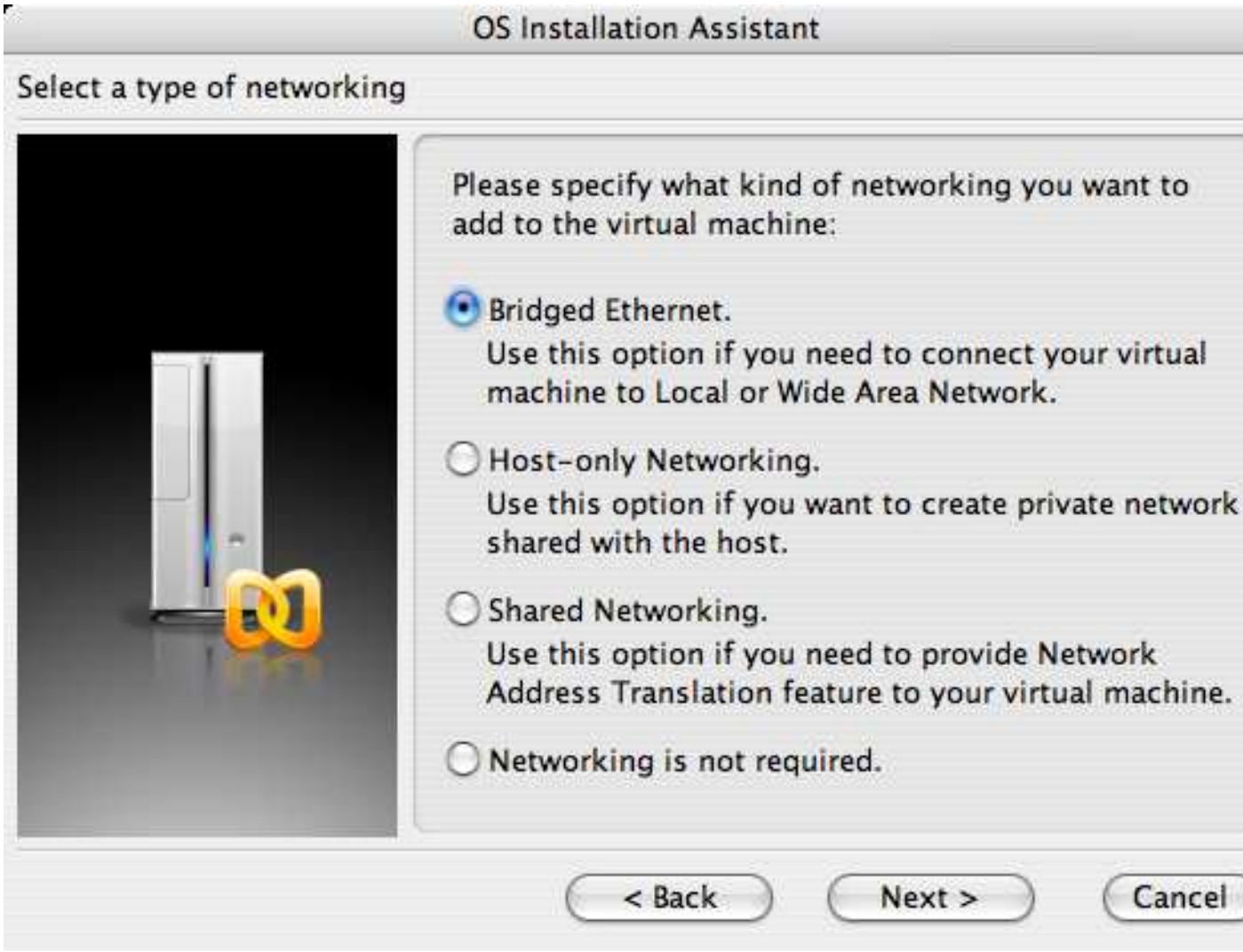

OS Installation Assistant
Specify memory size
Please specify how much memory (RAM) should be allocated for the virtual machine:
256
MB
4 MB
1500 MB
Recommended RAM maximum size is 1500 MB.
< Back
Next >
Cancel

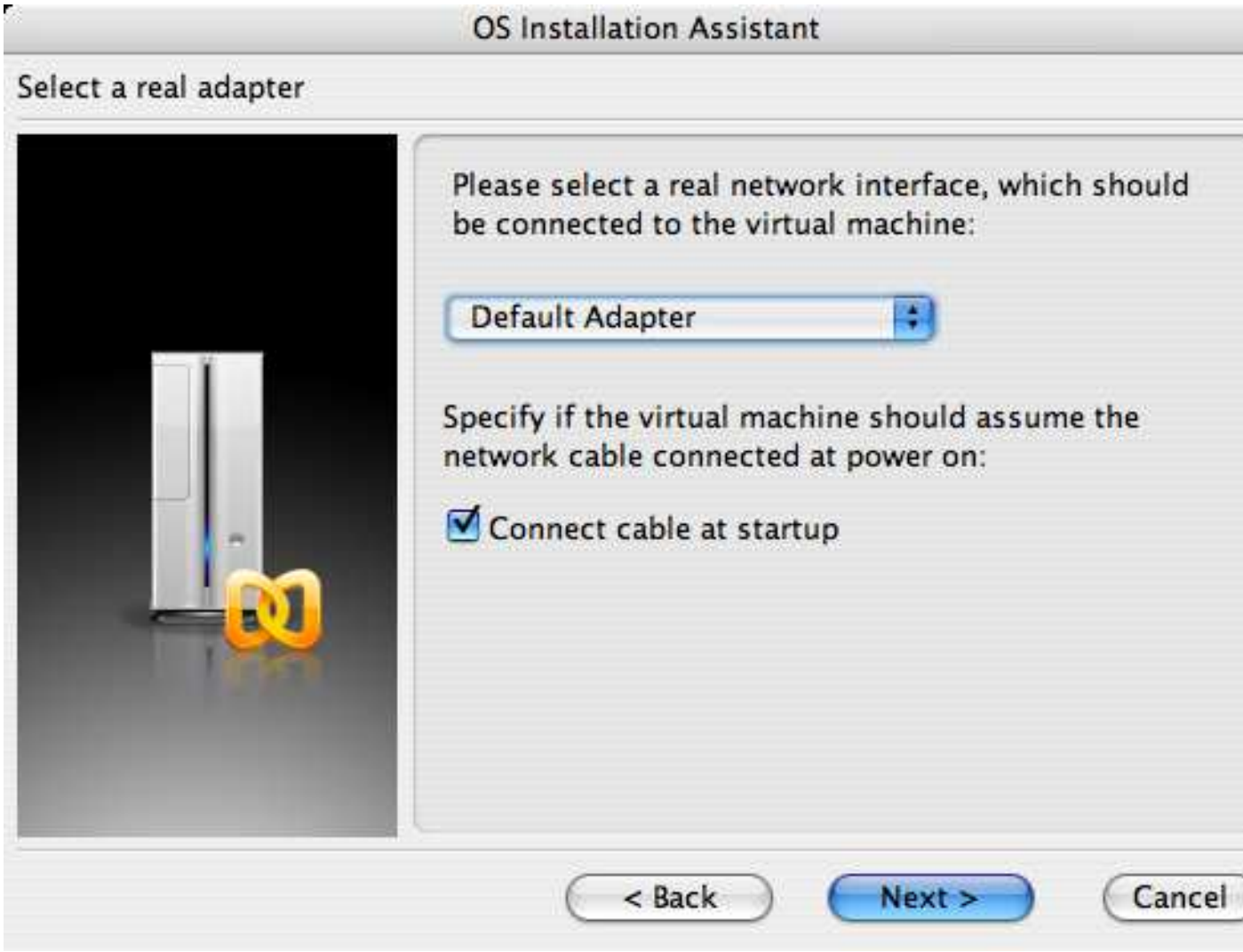

OS Installation Assistant
Select action type
Please specify what kind of hard disk you want to install to the virtual machine. If you do not want to add a hard disk now, select "Do not add hard disk" option. You will be able to add it later using a Configuration Editor.
Create a new virtual hard disk
Use an existing hard disk image
Do not add hard disk
< Back
Next >
Cancel


OS Installation Assistant
Specify hard disk options
Please specify a preferred virtual disk size:
8000
MB
Select disk format:
Expanding (recommended).
Disk image file is small initially and grows as you add more data to the virtual machine. This disk format takes less time to create and saves disk space on the host.
Plain.
Disk image file consumes all the allocated space right from the start. It takes more time to create but allows guest OS to operate faster.
< Back
Next >
Cancel



Select the type of networking and a network interface:

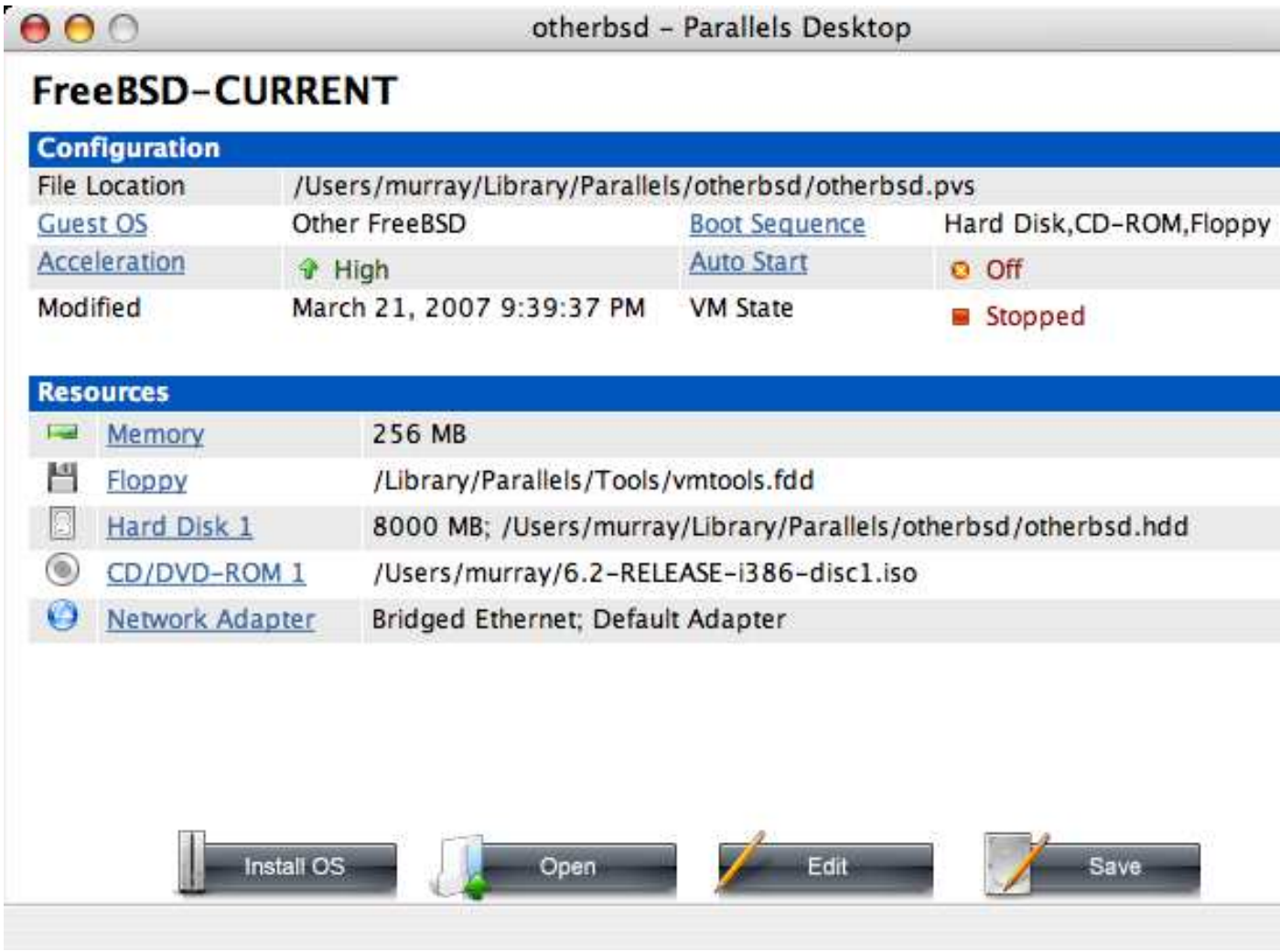

OS Installation Assistant
Select a type of networking
Please specify what kind of networking you want to add to the virtual machine:
Bridged Ethernet.
Use this option if you need to connect your virtual machine to Local or Wide Area Network.
Host-only Networking.
Use this option if you want to create private network shared with the host.
Shared Networking.
Use this option if you need to provide Network Address Translation feature to your virtual machine.
Networking is not required.
< Back
Next >
Cancel



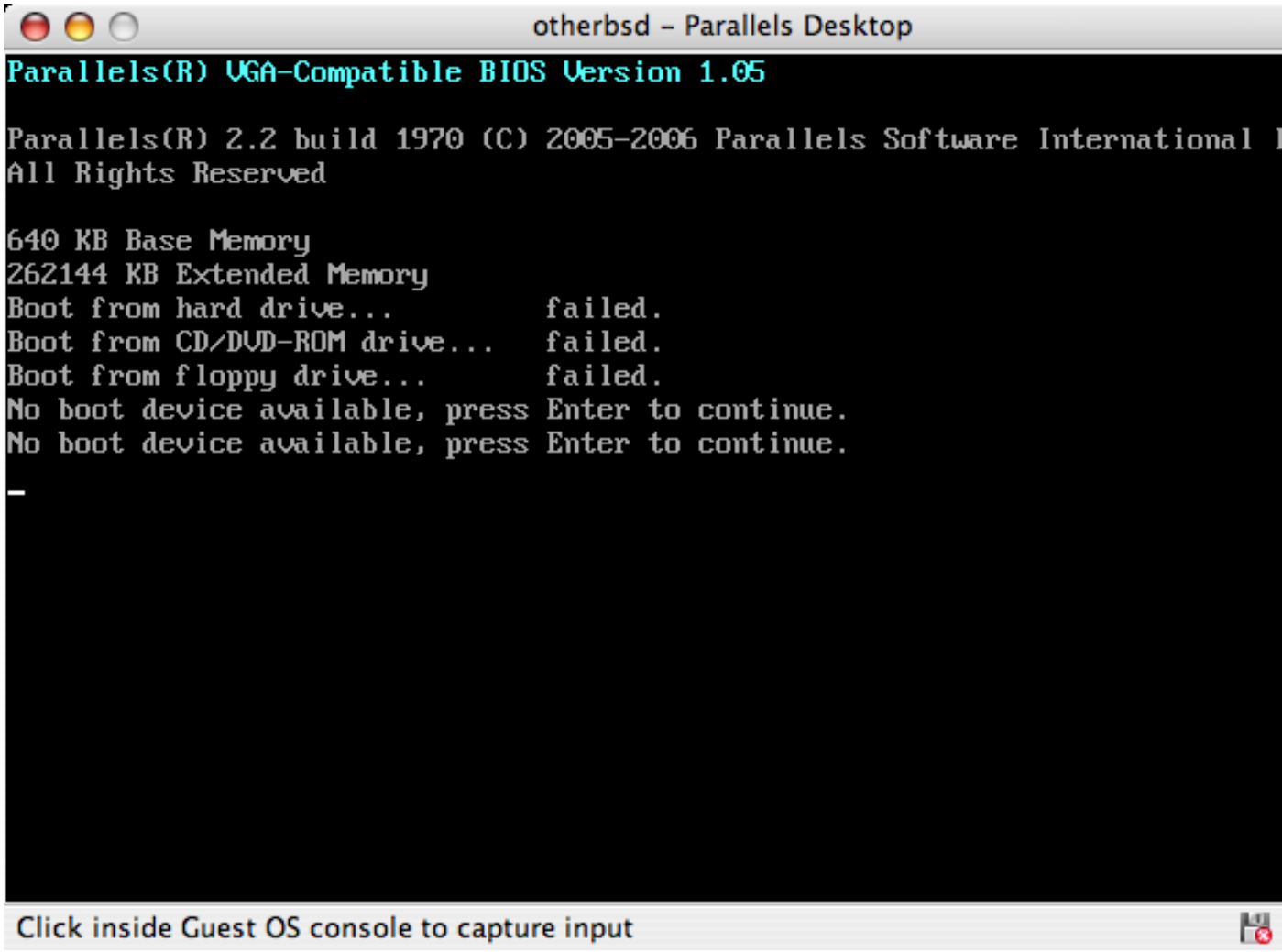
Save and finish the configuration:


OS Installation Assistant
Select virtual machine configuration file
Please specify a virtual machine name and location of the virtual machine configuration file.
Virtual Machine Name:
FreeBSD-CURRENT
Configuration File:
s/murray/Library/Parallels/otherbsd/otherbsd.pvs
...
< Back
Next >
Cancel

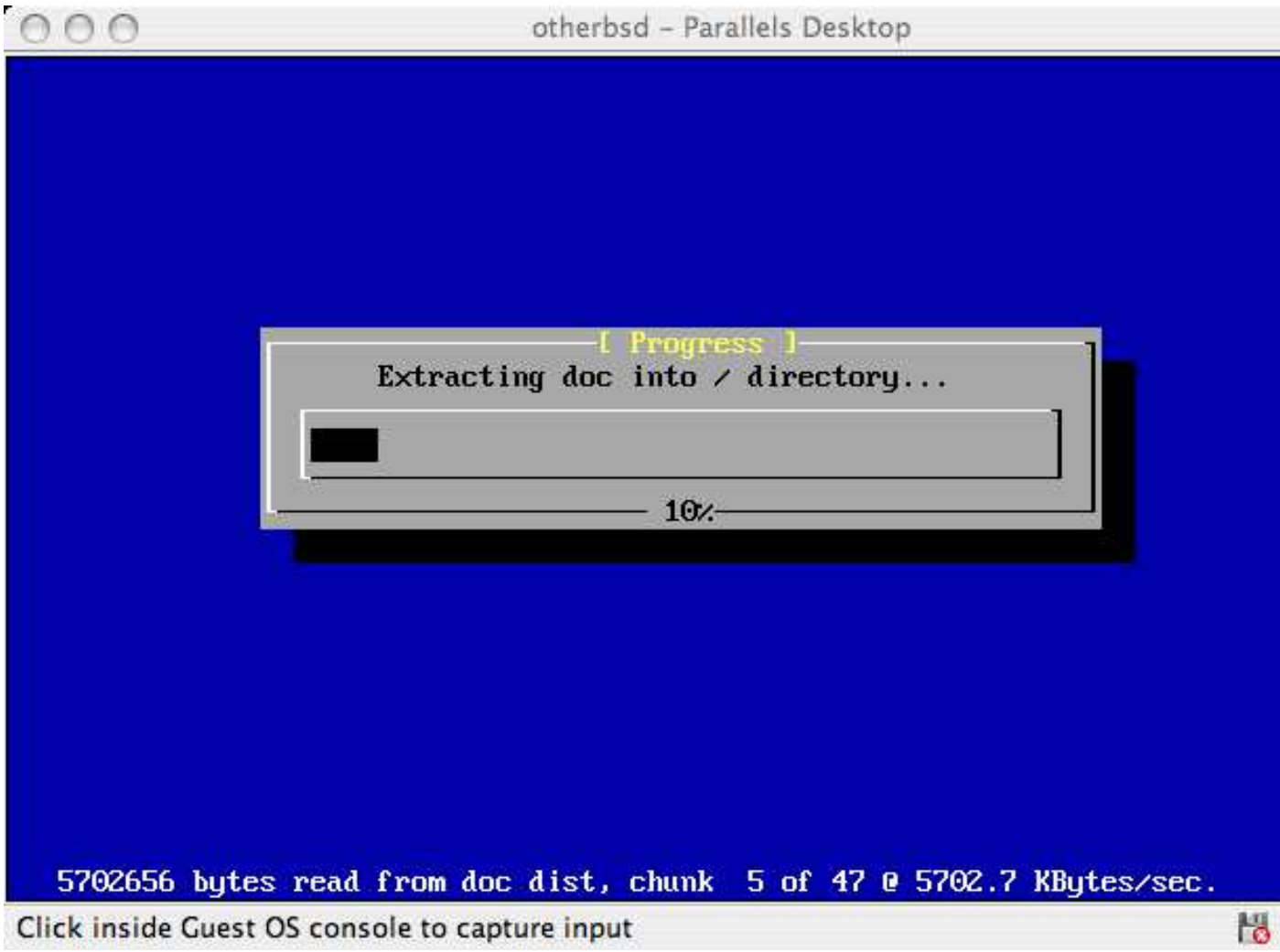
After your FreeBSD virtual machine has been created, you will need to install FreeBSD on it. This is best done with an official FreeBSD CDROM or with an ISO image downloaded from an official FTP site. When you have the appropriate ISO image on your local Mac filesystem or a CDROM in your Mac's CD drive, click on the disc icon in the bottom right corner of your FreeBSD **Parallels** window. This will bring up a window that allows you to associate the CDROM drive in your virtual machine with an ISO file on disk or with your real CDROM drive.



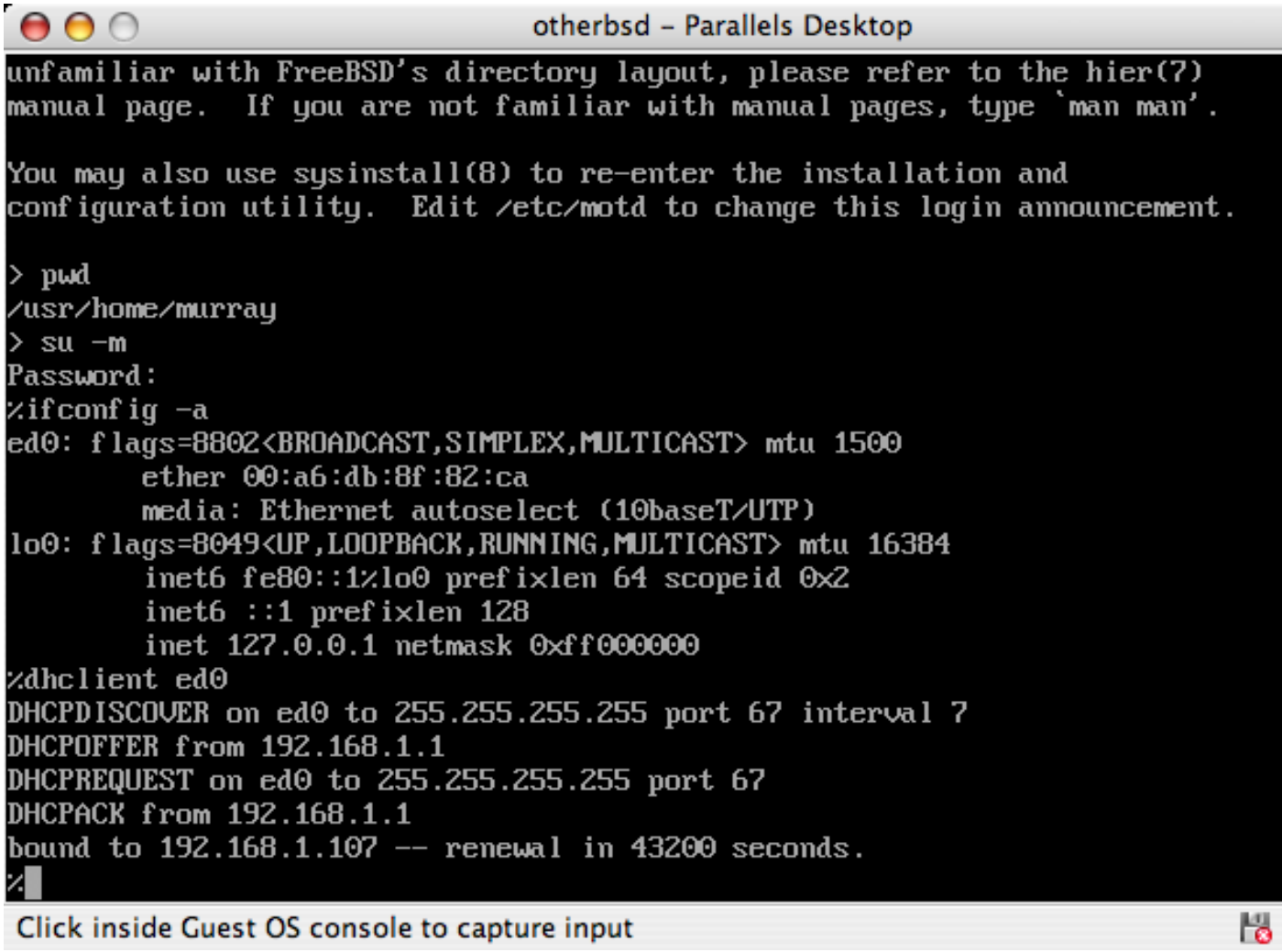
Once you have made this association with your CDROM source, reboot your FreeBSD virtual machine as normal by clicking the reboot icon. **Parallels** will reboot with a special BIOS that first checks if you have a CDROM just as a normal BIOS would do.



In this case it will find the FreeBSD installation media and begin a normal **sysinstall** based installation as described in Chapter 2. You may install, but do not attempt to configure X11 at this time.



When you have finished the installation, reboot into your newly installed FreeBSD virtual machine.



22.2.1.2 Configuring FreeBSD on Mac OS X/Parallels

After FreeBSD has been successfully installed on Mac OS X with **Parallels**, there are a number of configuration steps that can be taken to optimize the system for virtualized operation.

1. Set boot loader variables

   The most important step is to reduce the `kern.hz` tunable to reduce the CPU utilization of FreeBSD under the **Parallels** environment. This is accomplished by adding the following line to `/boot/loader.conf`:

   ```
   kern.hz=100
   ```

   Without this setting, an idle FreeBSD **Parallels** guest OS will use roughly 15% of the CPU of a single processor iMac®. After this change the usage will be closer to a mere 5%.

2. Create a new kernel configuration file

   You can remove all of the SCSI, FireWire, and USB device drivers. **Parallels** provides a virtual network adapter used by the ed(4) driver, so all other network devices except for ed(4) and miibus(4) can be removed from the kernel.

3. Setup networking

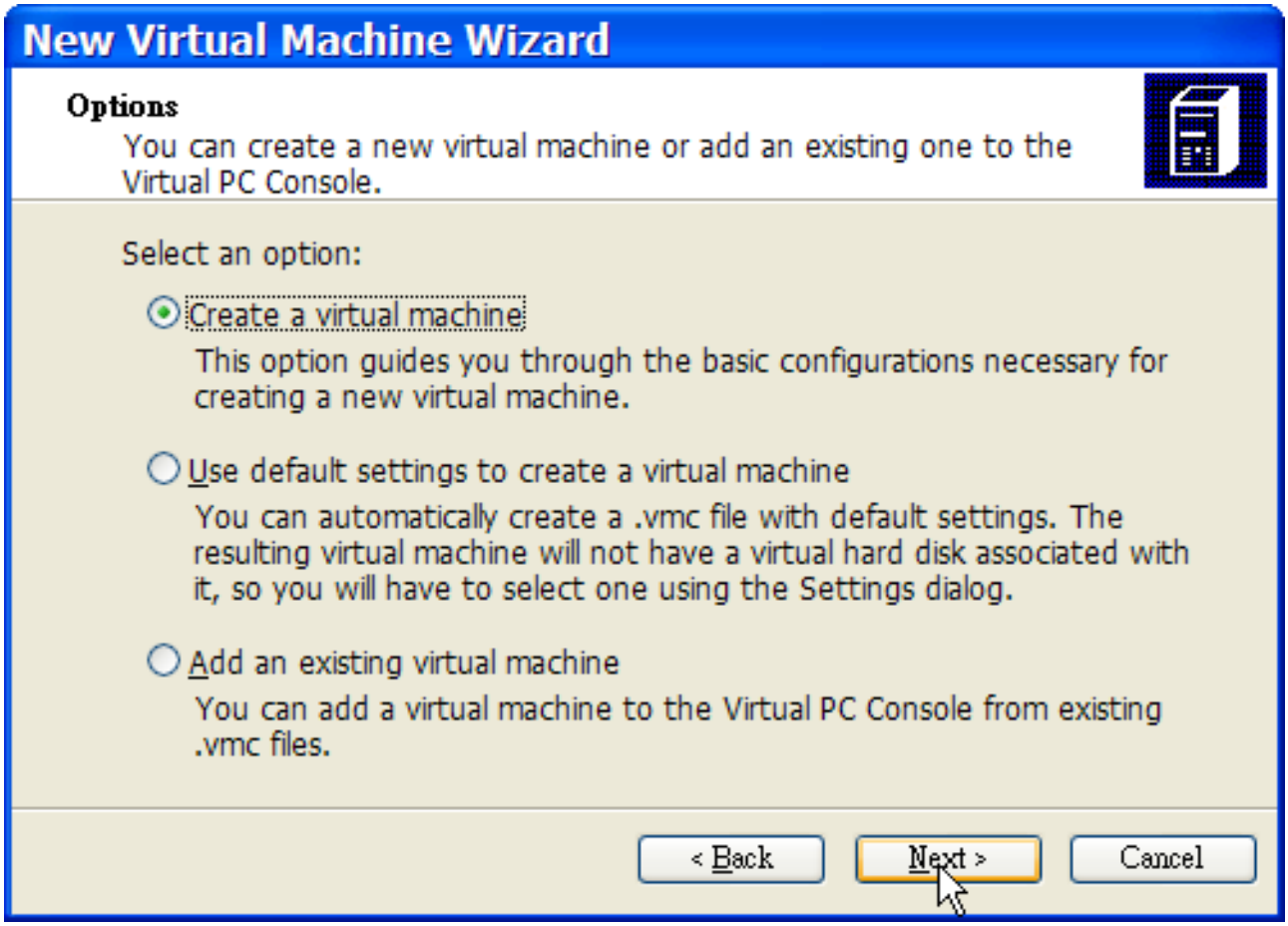
   The most basic networking setup involves simply using DHCP to connect your virtual machine to the same local area network as your host Mac. This can be accomplished by adding `ifconfig_ed0="DHCP"` to `/etc/rc.conf`. More advanced networking setups are described in Chapter 31.

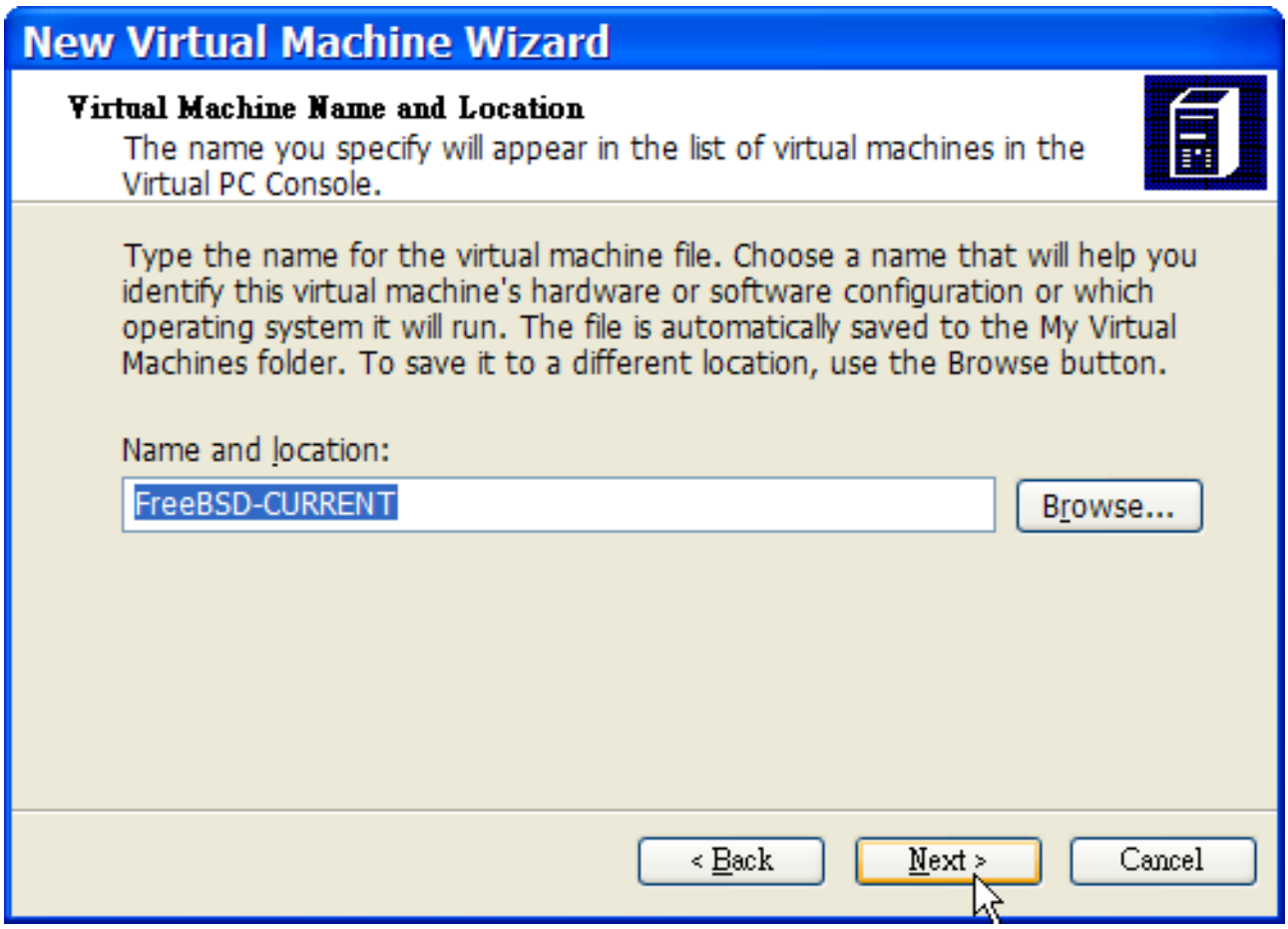
22.2.2 Virtual PC on Windows

Virtual PC for Windows is a Microsoft software product available for free download. See system requirements (http://www.microsoft.com/windows/downloads/virtualpc/sysreq.mspx). Once **Virtual PC** has been installed on Microsoft Windows, the user must configure a virtual machine and then install the desired guest operating system.

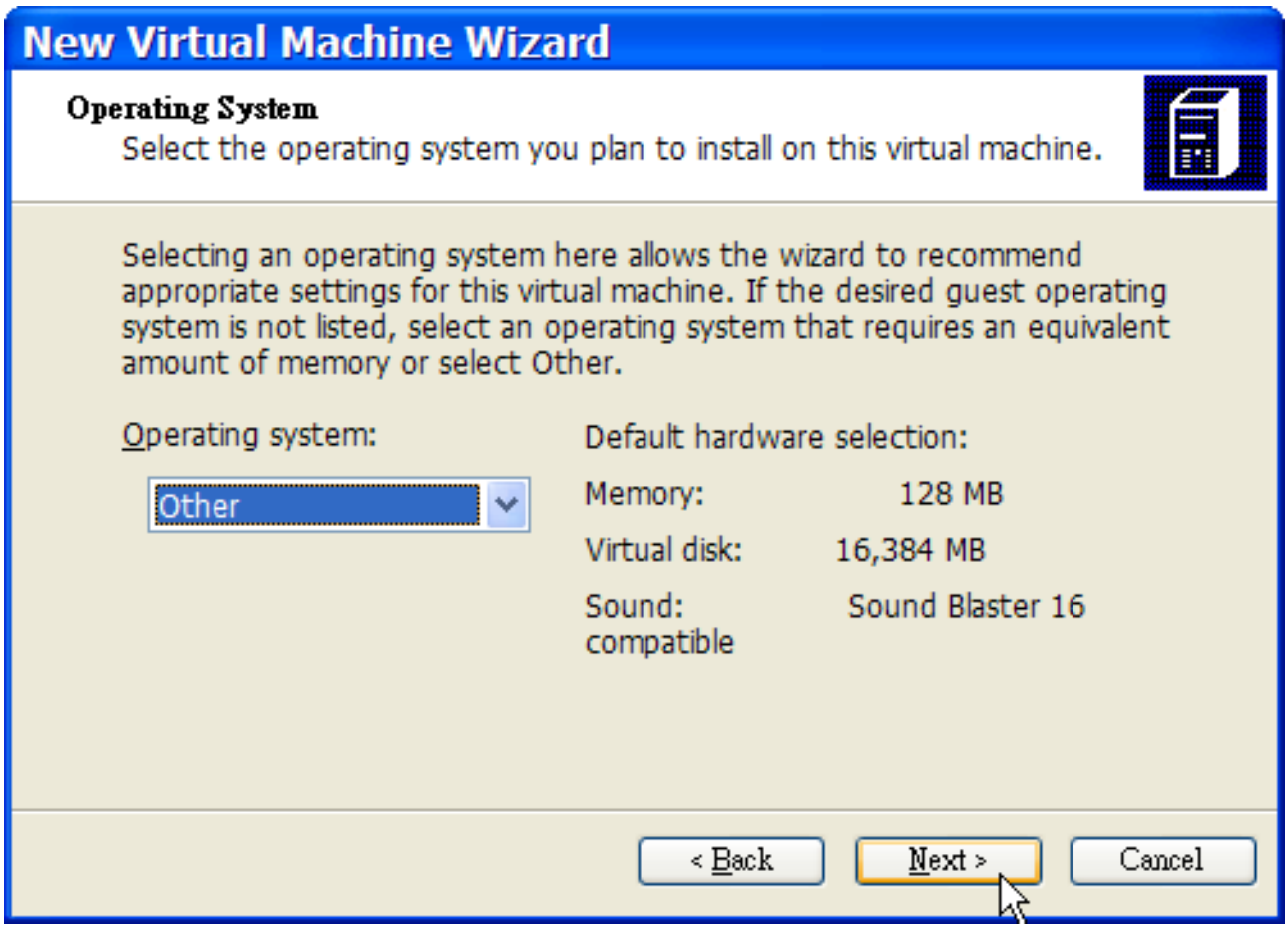
22.2.2.1 Installing FreeBSD on Virtual PC/Microsoft® Windows

The first step in installing FreeBSD on Microsoft Windows /**Virtual PC** is to create a new virtual machine for installing FreeBSD. Select Create a virtual machine when prompted:

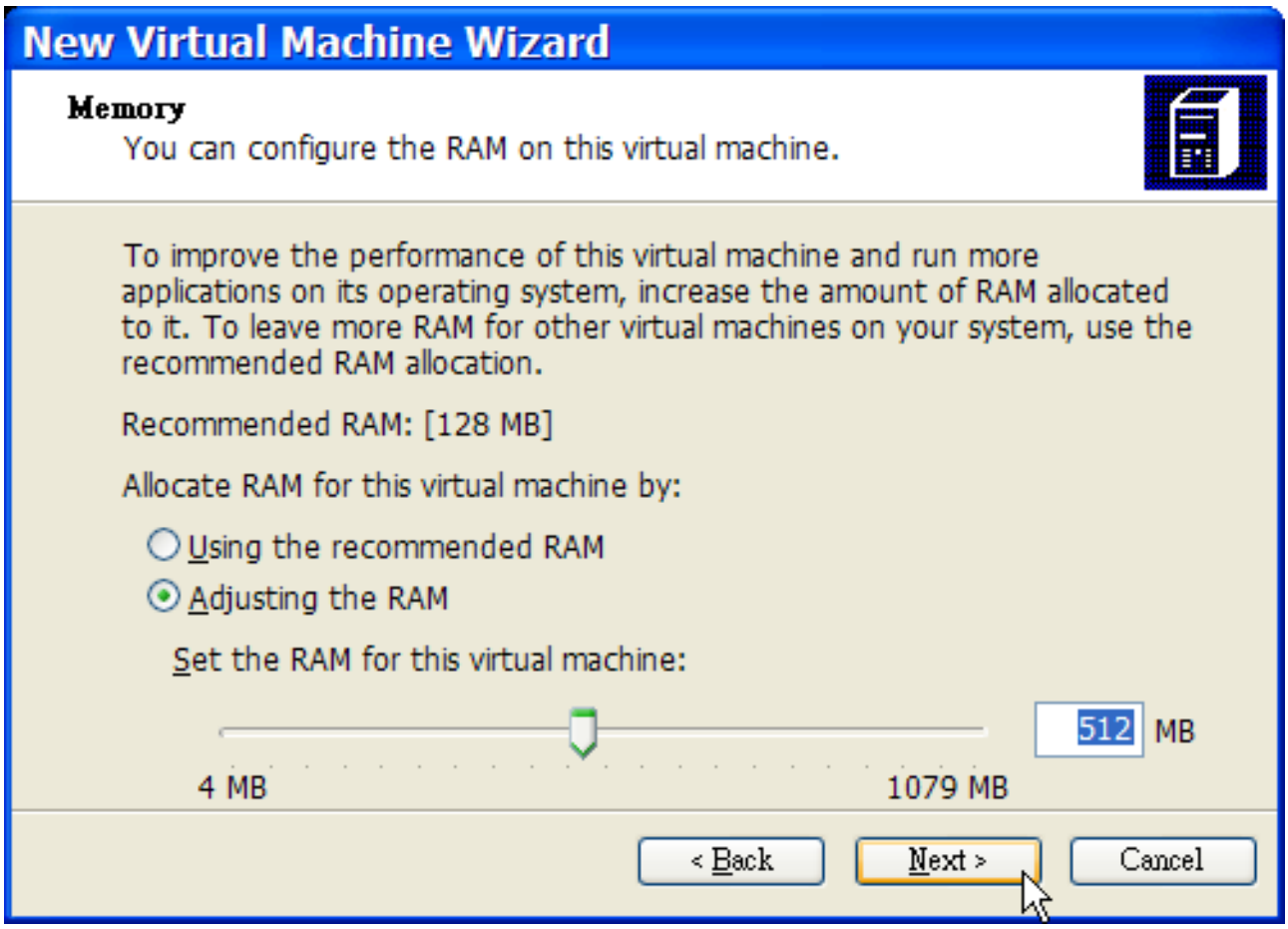


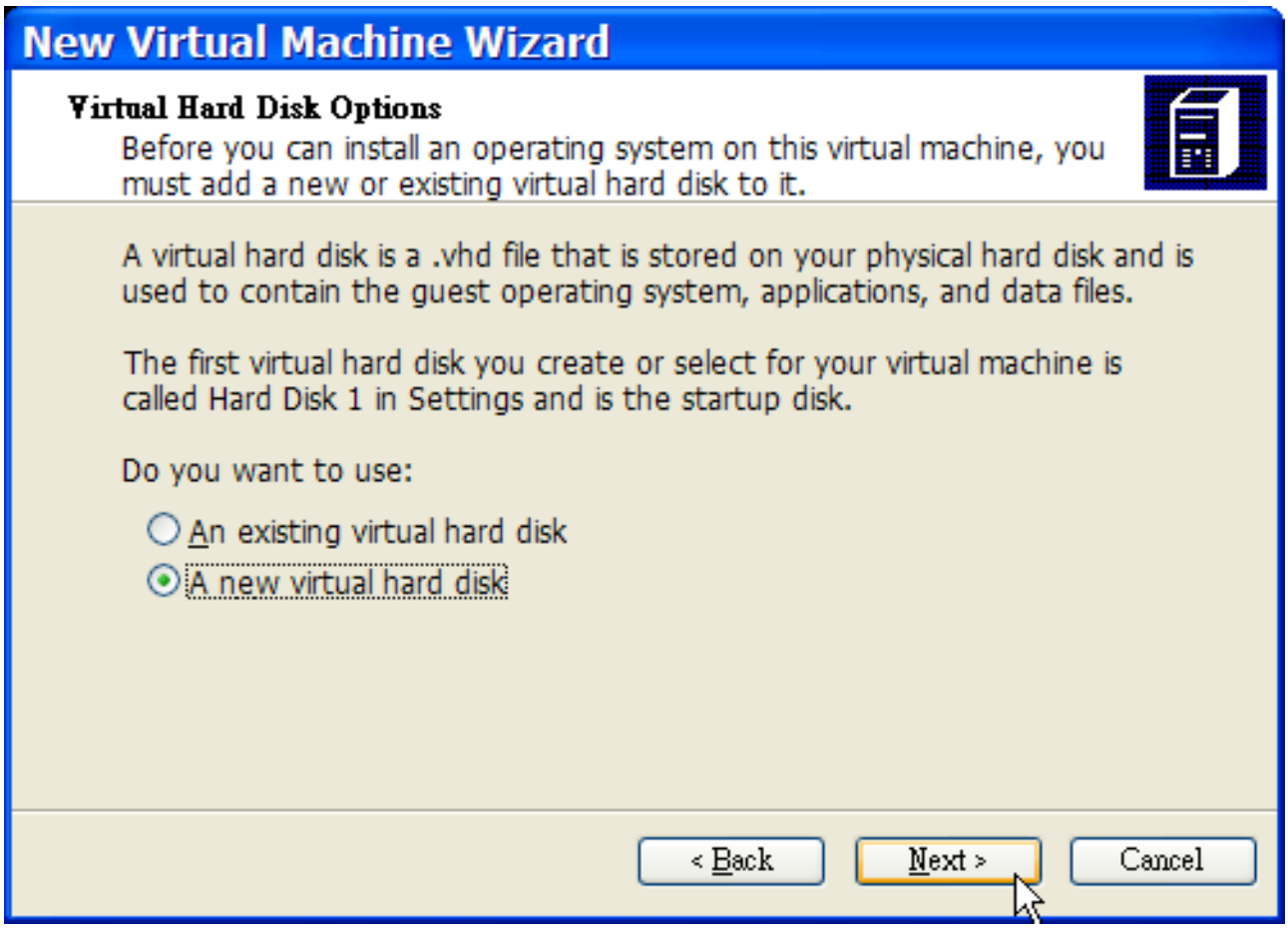


And select Other as the Operating system when prompted:

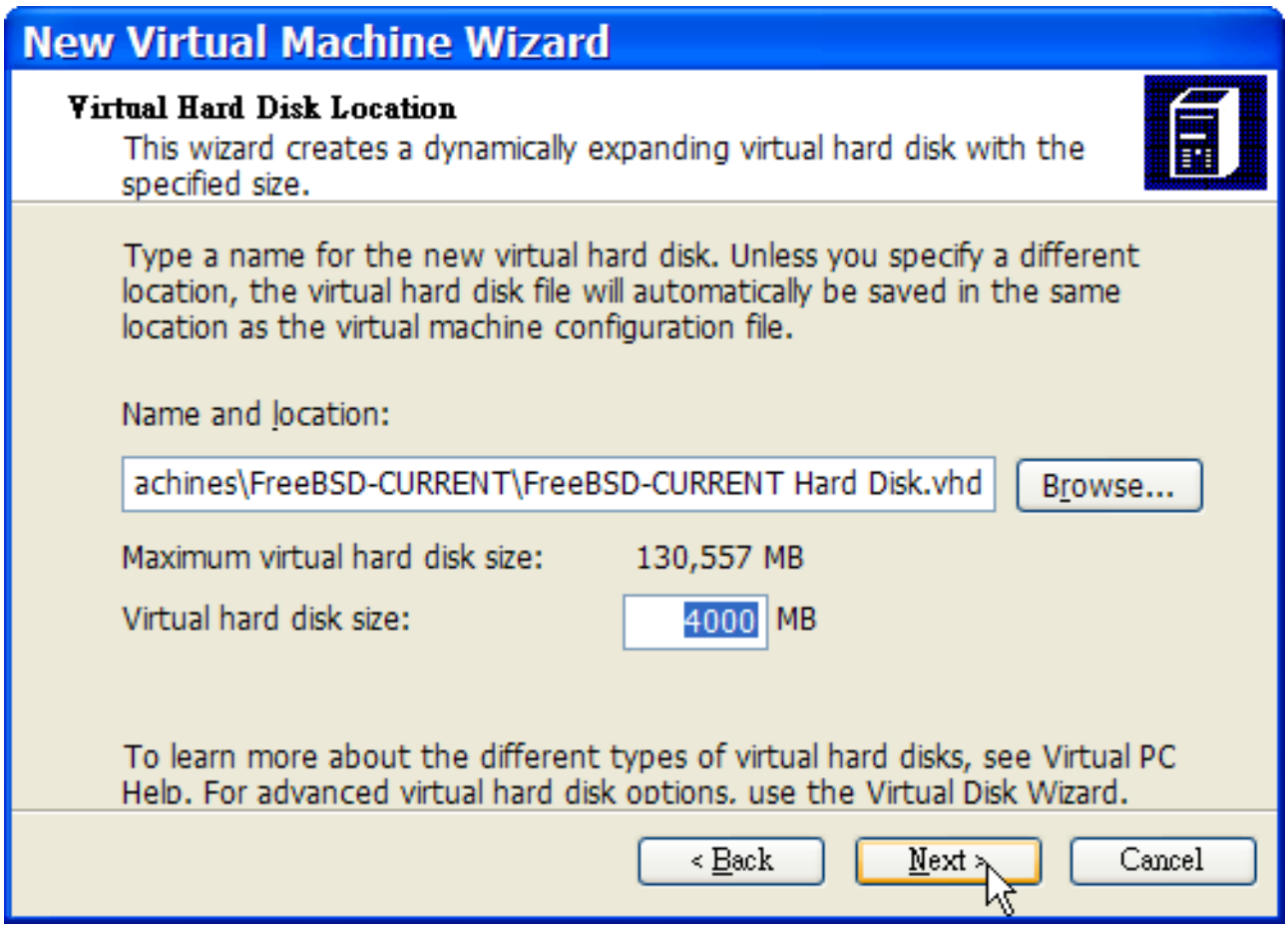


Then, choose a reasonable amount of disk and memory depending on your plans for this virtual FreeBSD instance. 4GB of disk space and 512MB of RAM work well for most uses of FreeBSD under **Virtual PC**:

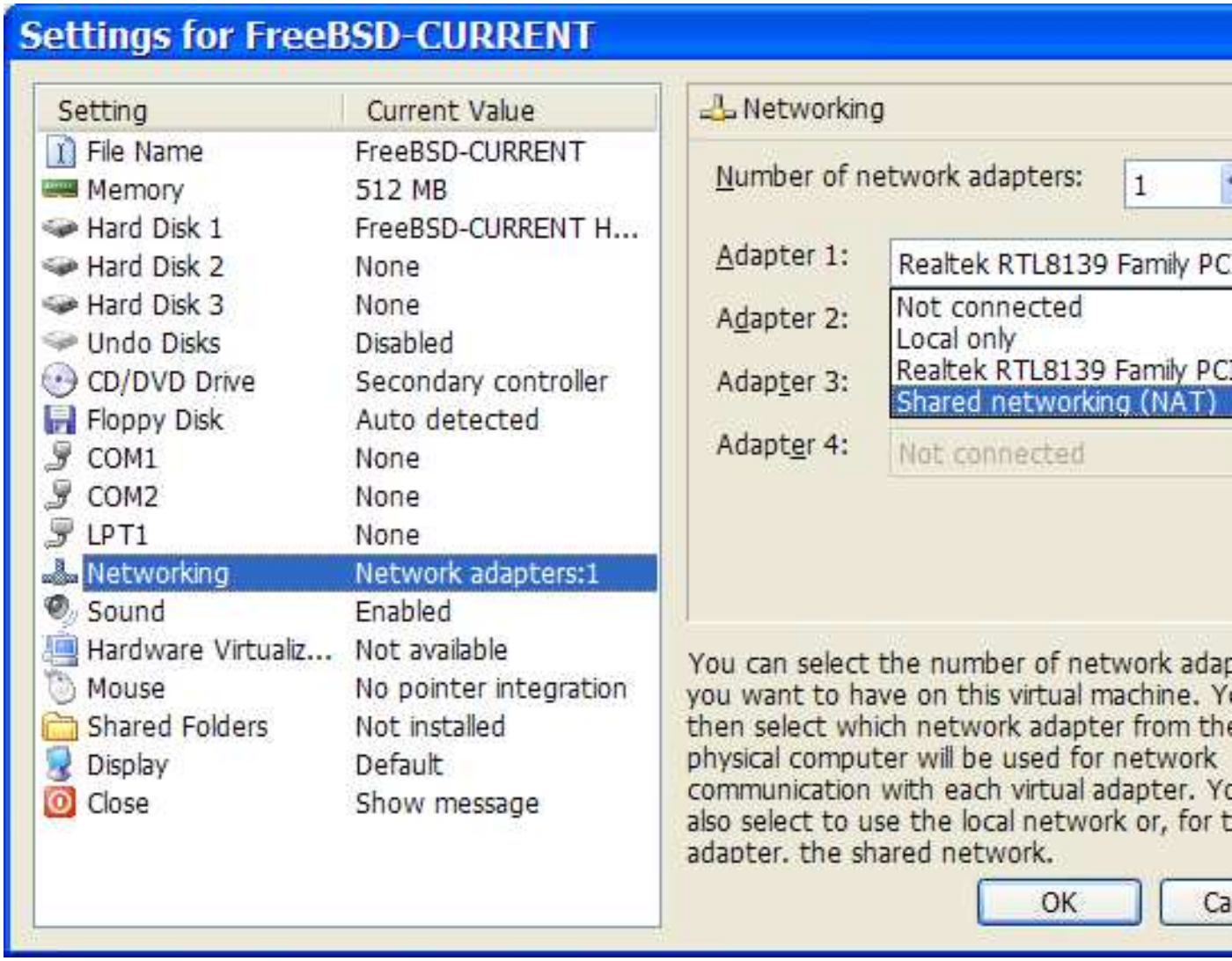

New Virtual Machine Wizard
Memory
You can configure the RAM on this virtual machine.
To improve the performance of this virtual machine and run more applications on its operating system, increase the amount of RAM allocated to it. To leave more RAM for other virtual machines on your system, use the recommended RAM allocation.
Recommended RAM: [128 MB]
Allocate RAM for this virtual machine by:
Using the recommended RAM
Adjusting the RAM
Set the RAM for this virtual machine:
512
MB
4 MB
1079 MB
< Back
Next >
Cancel



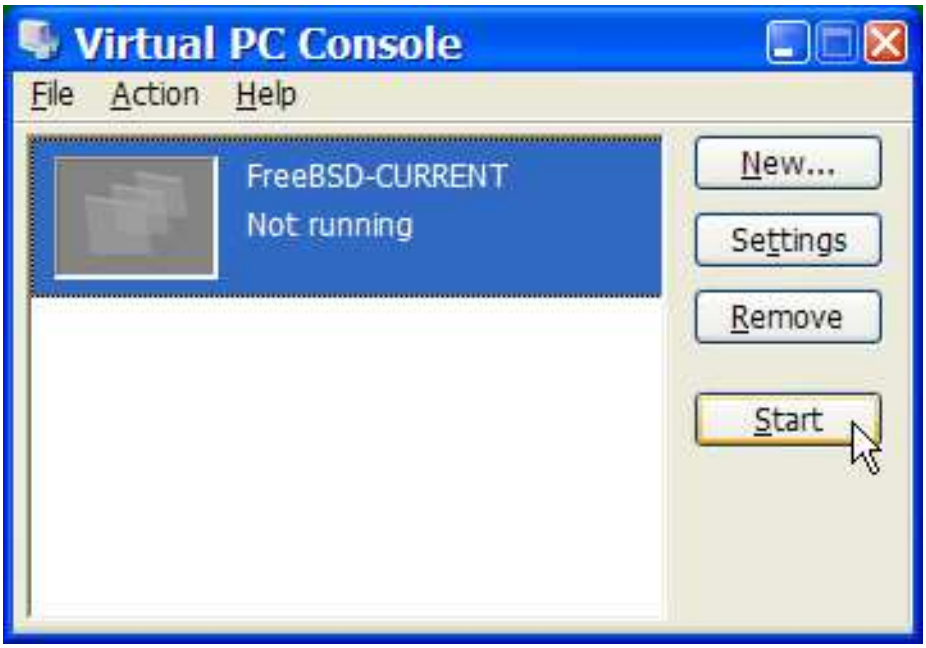
Save and finish the configuration:

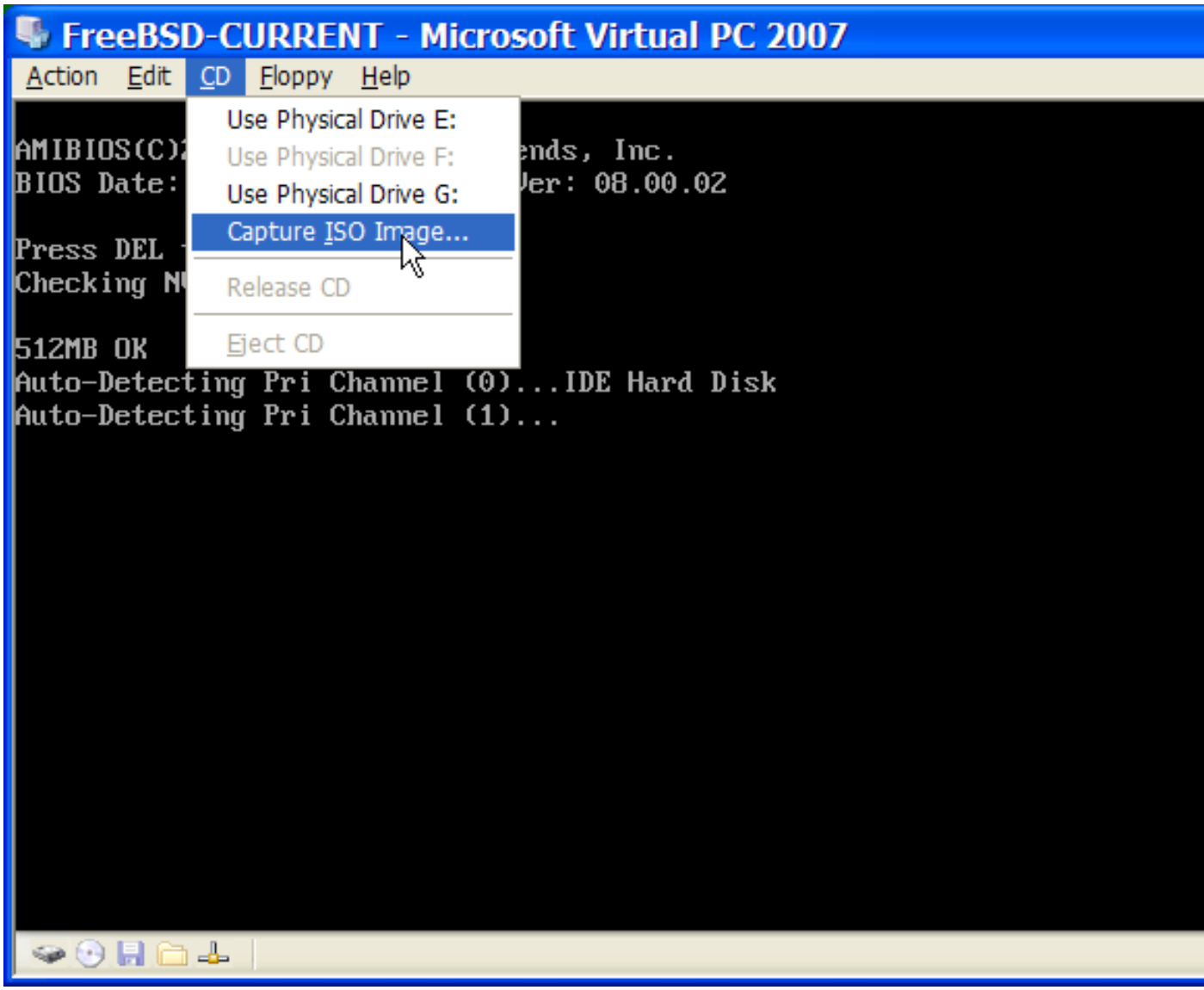


Select your FreeBSD virtual machine and click Settings, then set the type of networking and a network interface:

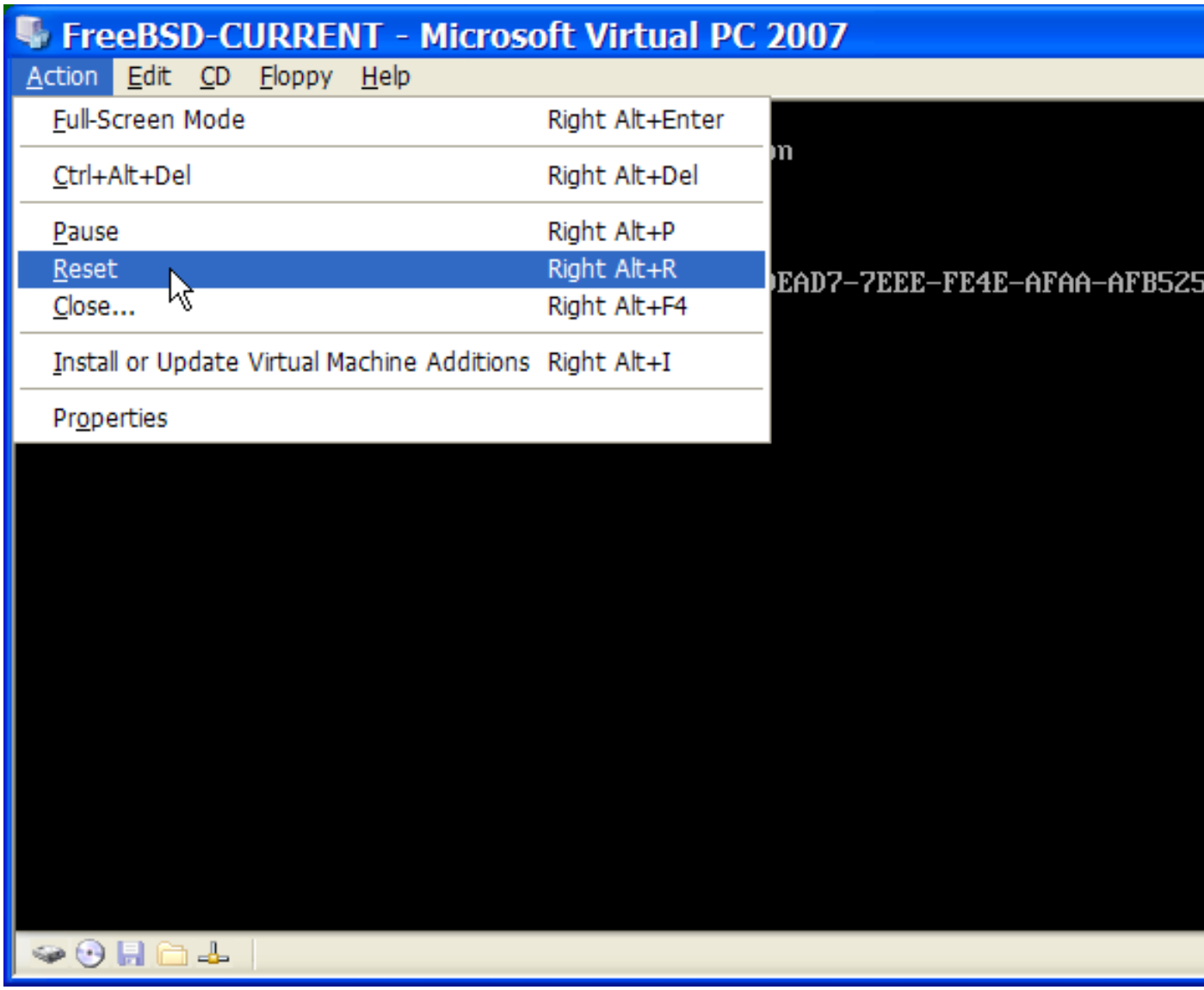


After your FreeBSD virtual machine has been created, you will need to install FreeBSD on it. This is best done with an official FreeBSD CDROM or with an ISO image downloaded from an official FTP site. When you have the appropriate ISO image on your local Windows filesystem or a CDROM in your CD drive, double click on your FreeBSD virtual machine to boot. Then, click CD and choose Capture ISO Image... on **Virtual PC** window. This will bring up a window that allows you to associate the CDROM drive in your virtual machine with an ISO file on disk or with your real CDROM drive.

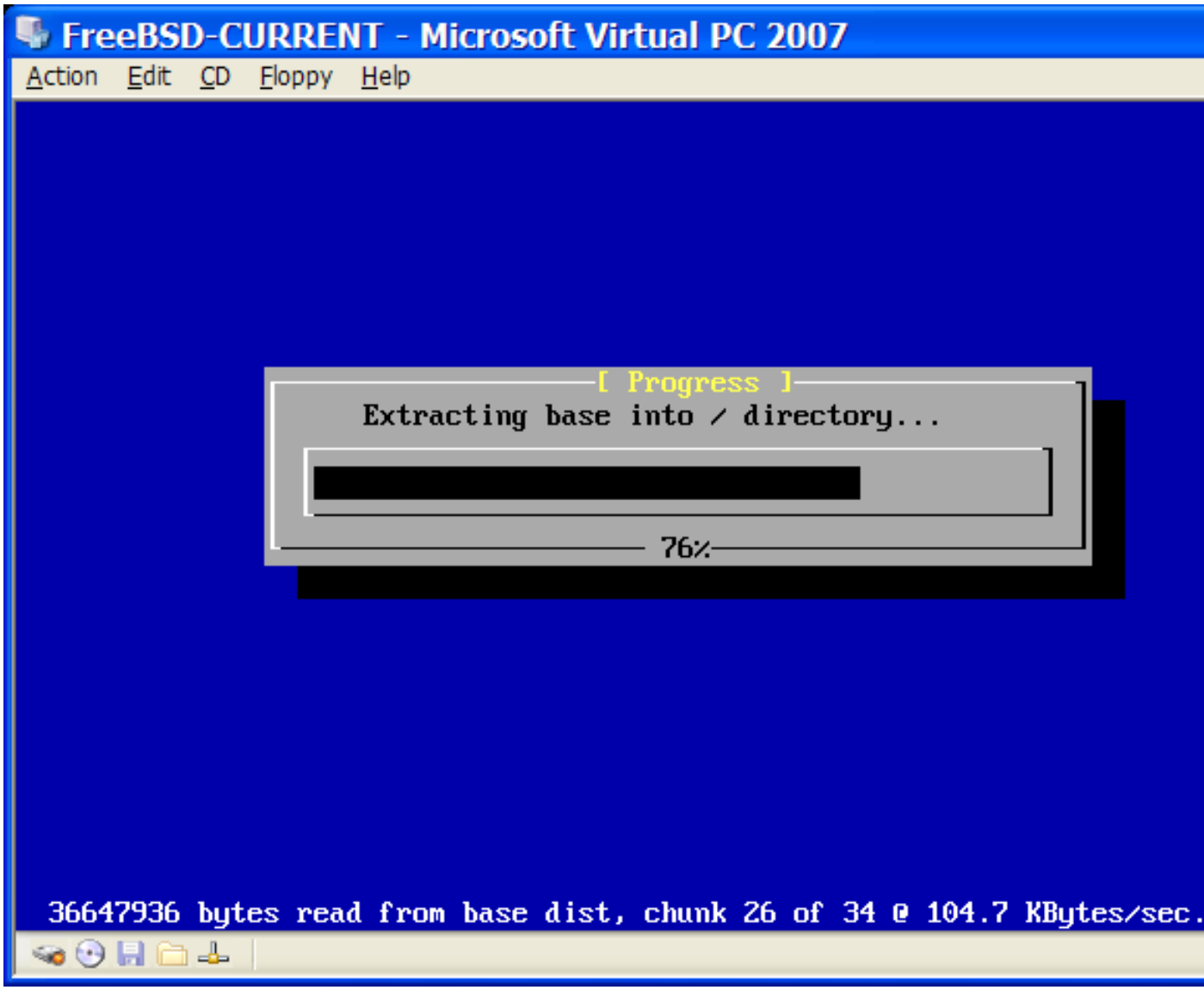

Virtual PC Console
File Action Help
FreeBSD-CURRENT
Not running
New...
Settings
Remove
Start



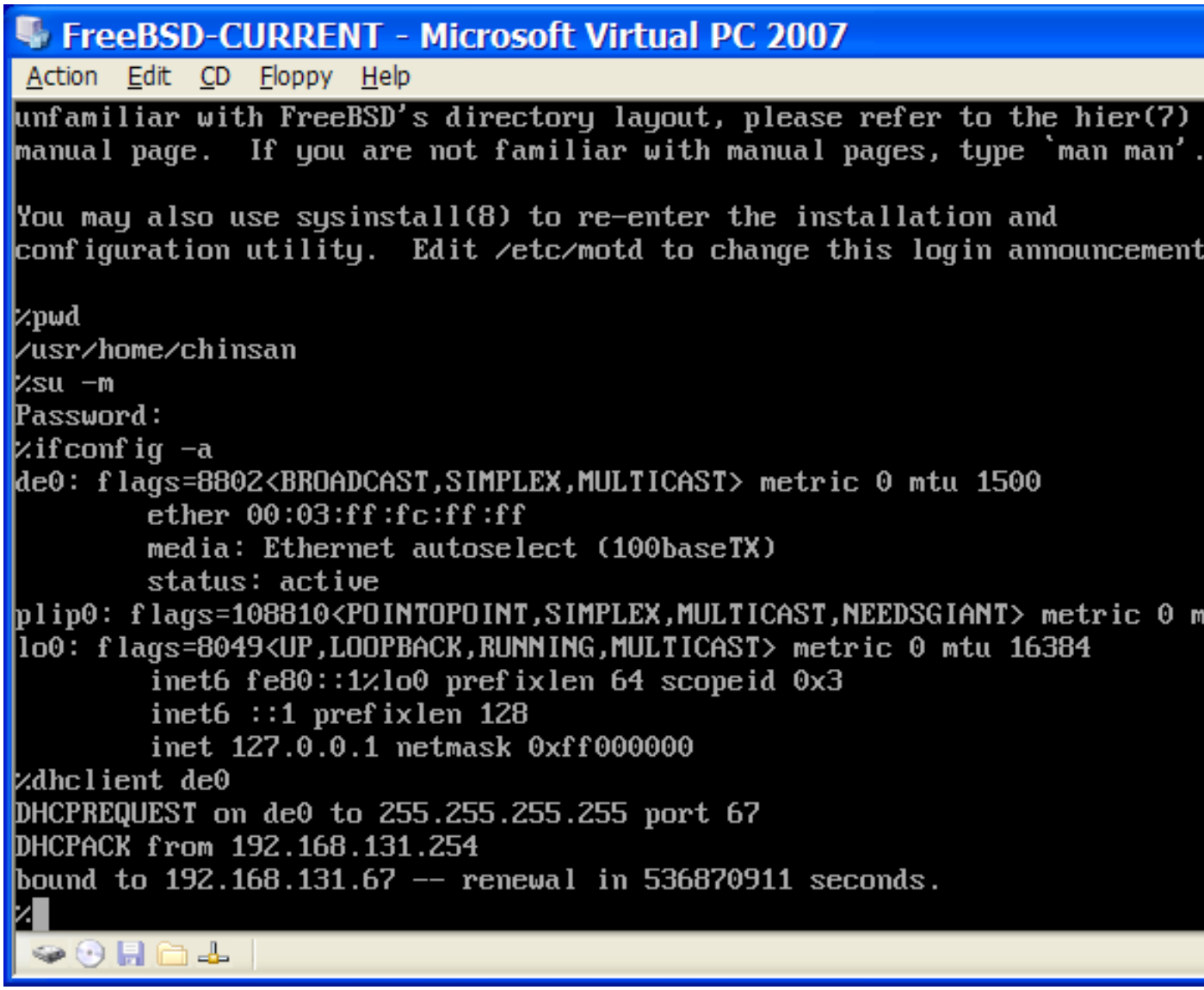
Once you have made this association with your CDROM source, reboot your FreeBSD virtual machine as normal by clicking the Action and Reset. **Virtual PC** will reboot with a special BIOS that first checks if you have a CDROM just as a normal BIOS would do.



In this case it will find the FreeBSD installation media and begin a normal **sysinstall** based installation as described in Chapter 2. You may install, but do not attempt to configure X11 at this time.



When you have finished the installation, remember to eject CDROM or release ISO image. Finally, reboot into your newly installed FreeBSD virtual machine.



22.2.2.2 Configuring FreeBSD on Microsoft Windows/Virtual PC

After FreeBSD has been successfully installed on Microsoft Windows with **Virtual PC**, there are a number of configuration steps that can be taken to optimize the system for virtualized operation.

1. Set boot loader variables

   The most important step is to reduce the `kern.hz` tunable to reduce the CPU utilization of FreeBSD under the **Virtual PC** environment. This is accomplished by adding the following line to `/boot/loader.conf`:

   ```
   kern.hz=100
   ```

   Without this setting, an idle FreeBSD **Virtual PC** guest OS will use roughly 40% of the CPU of a single processor computer. After this change the usage will be closer to a mere 3%.

2. Create a new kernel configuration file

You can remove all of the SCSI, FireWire, and USB device drivers. **Virtual PC** provides a virtual network adapter used by the de(4) driver, so all other network devices except for de(4) and miibus(4) can be removed from the kernel.

3. Setup networking

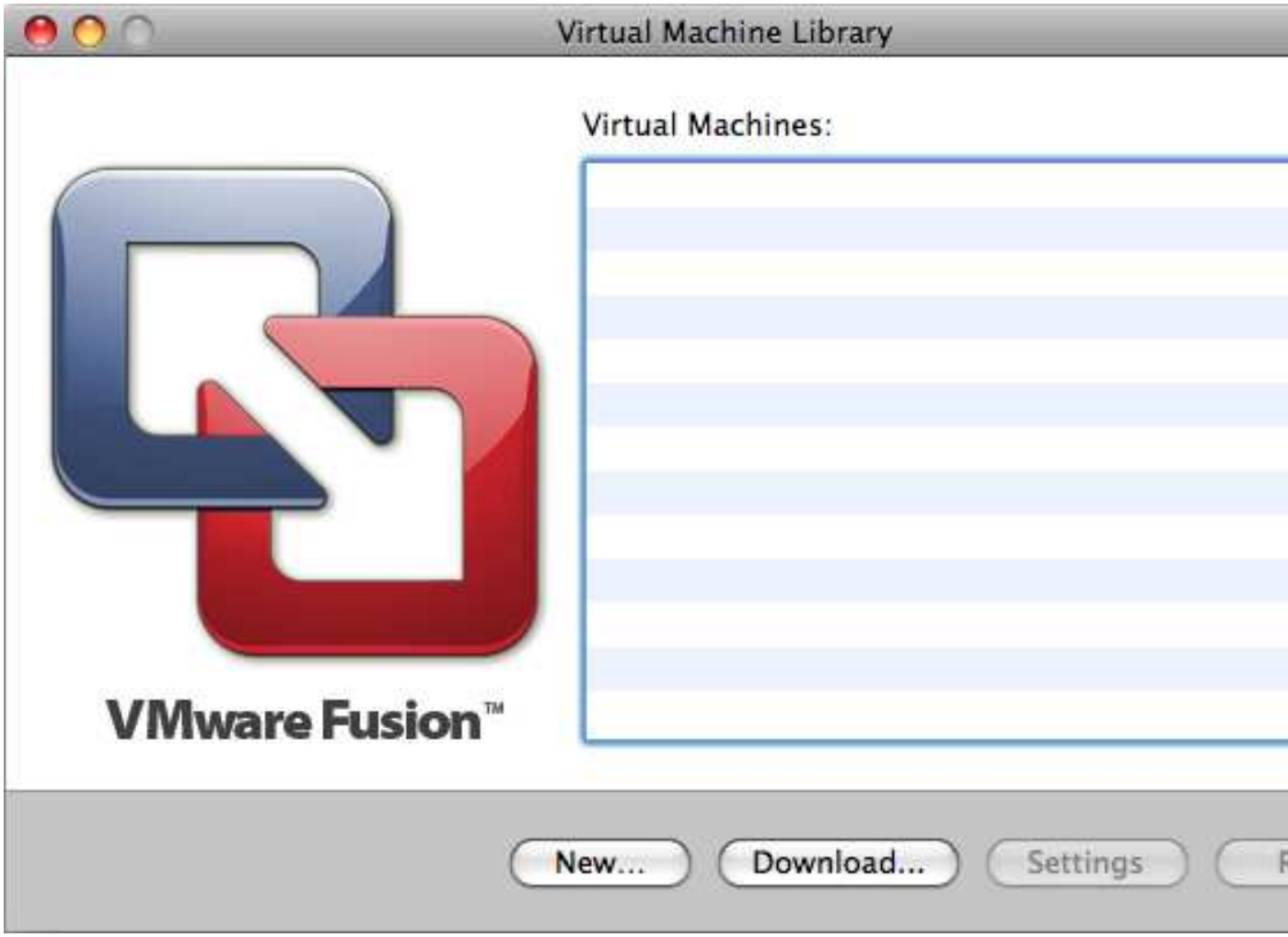
   The most basic networking setup involves simply using DHCP to connect your virtual machine to the same local area network as your host Microsoft Windows. This can be accomplished by adding `ifconfig_de0="DHCP"` to `/etc/rc.conf`. More advanced networking setups are described in Chapter 31.

22.2.3 VMware on MacOS

VMware Fusion for Mac is a commercial software product available for Intel based Apple Mac computers running Mac OS 10.4.9 or higher. FreeBSD is a fully supported guest operating system. Once **VMware Fusion** has been installed on Mac OS X, the user must configure a virtual machine and then install the desired guest operating system.

22.2.3.1 Installing FreeBSD on VMware/Mac OS X

The first step is to start VMware Fusion, the Virtual Machine Library will load. Click "New" to create the VM:



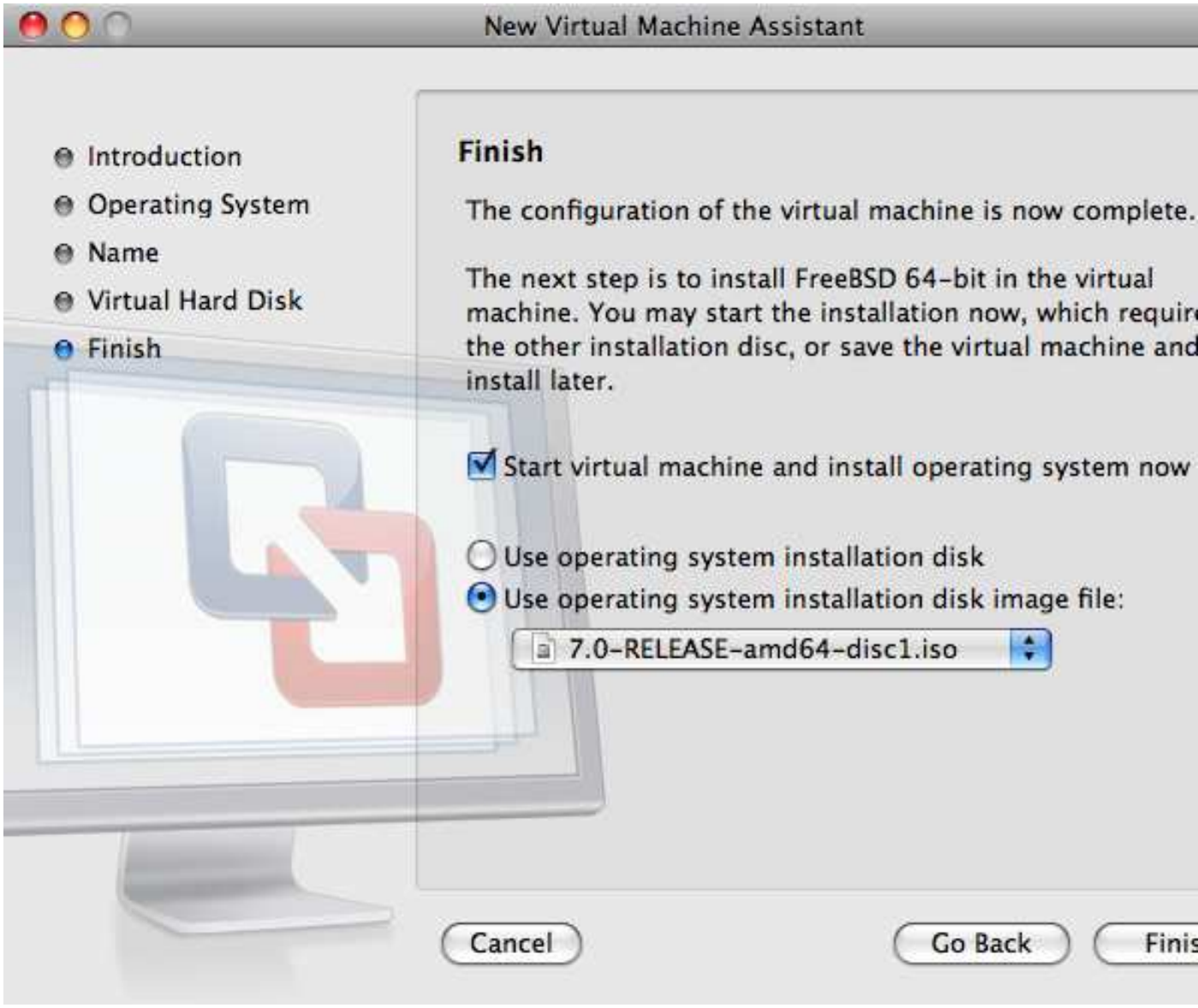
This will load the New Virtual Machine Assistant to help you create the VM, click Continue to proceed:

Select Other as the Operating System and FreeBSD or FreeBSD 64-bit, depending on if you want 64-bit support, as the Version when prompted:

Choose the Name of the VM Image and the Directory where you would like it saved:

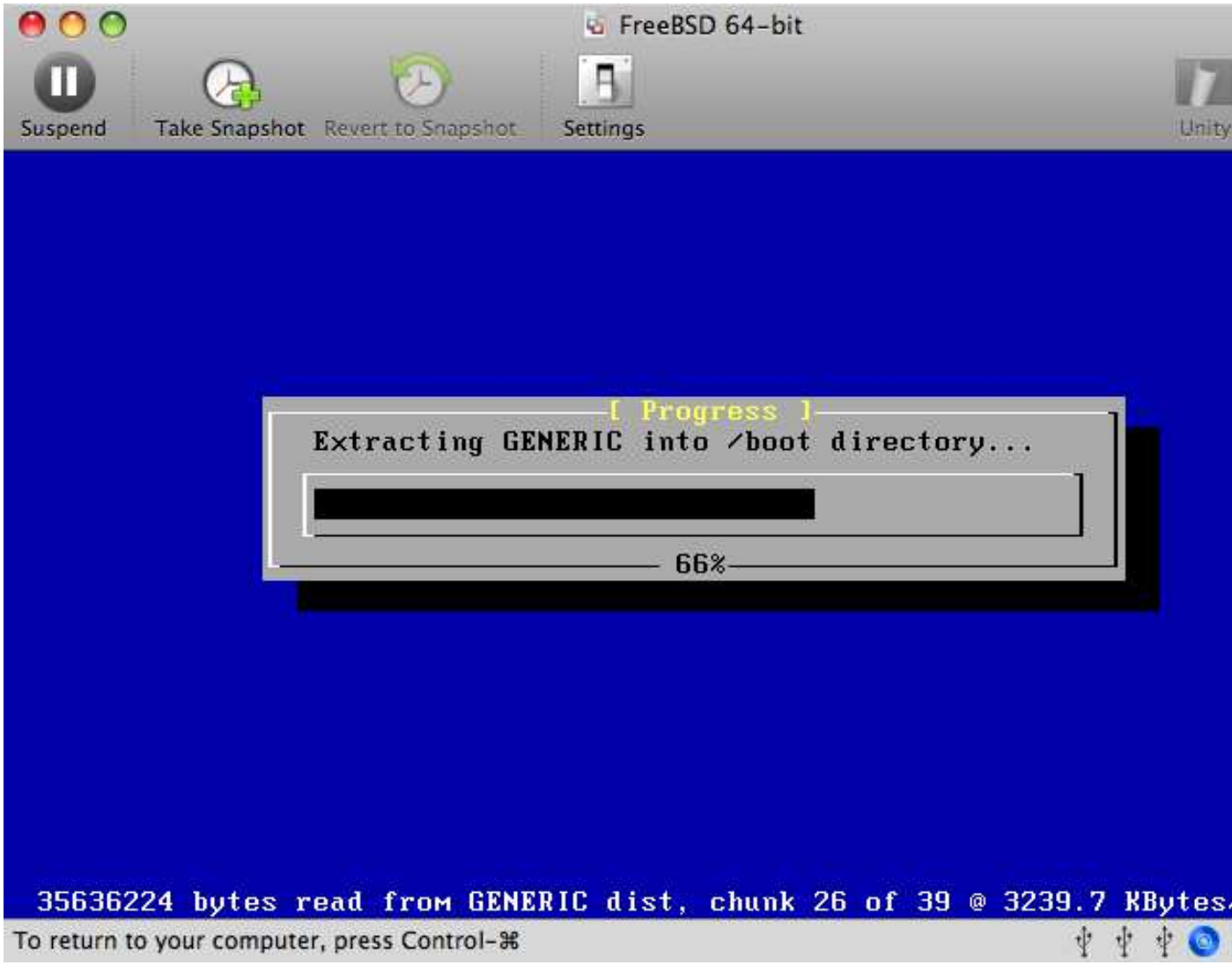
Choose the size of the Virtual Hard Disk for the VM:

Choose the method you would like to install the VM, either from an ISO image or from a CD:



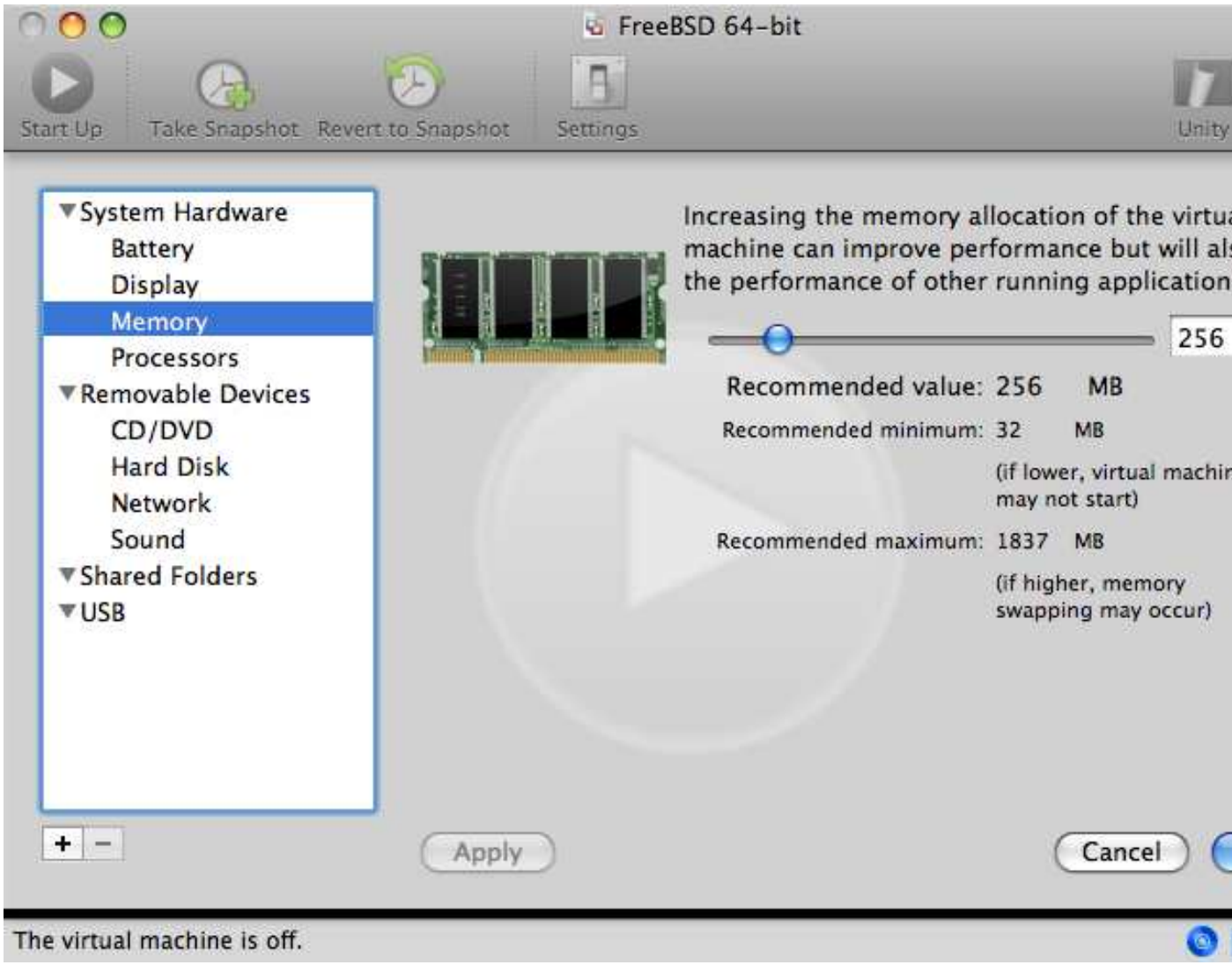
Once you click Finish, the VM will boot:

Install FreeBSD like you normally would, or by following the directions in Chapter 2:



Once the install is complete you can modify the settings of the VM, such as Memory Usage:

Note: The System Hardware settings of the VM cannot be modified while the VM is running.



The number of CPUs the VM will have access to:

The status of the CD-Rom Device. Normally you can disconnect the CD-Rom/ISO from the VM if you will not be needing it anymore.

The last thing to change is how the VM will connect to the Network. If you want to allow connections to the VM from other machines besides the Host, make sure you choose the Connect directly to the physical network (Bridged). Otherwise Share the host's internet connection (NAT) is preferred so that the VM can have access to the Internet, but the network cannot access the VM.

After you have finished modifying the settings, boot the newly installed FreeBSD virtual machine.

22.2.3.2 Configuring FreeBSD on Mac OS X/VMware

After FreeBSD has been successfully installed on Mac OS X with **VMware**, there are a number of configuration steps that can be taken to optimize the system for virtualized operation.

1. Set boot loader variables

   The most important step is to reduce the `kern.hz` tunable to reduce the CPU utilization of FreeBSD under the **VMware** environment. This is accomplished by adding the following line to `/boot/loader.conf`:

   ```
   kern.hz=100
   ```

   Without this setting, an idle FreeBSD **VMware** guest OS will use roughly 15% of the CPU of a single processor iMac. After this change the usage will be closer to a mere 5%.

2. Create a new kernel configuration file

You can remove all of the FireWire, and USB device drivers. **VMware** provides a virtual network adapter used by the em(4) driver, so all other network devices except for em(4) can be removed from the kernel.

3. Setup networking

   The most basic networking setup involves simply using DHCP to connect your virtual machine to the same local area network as your host Mac. This can be accomplished by adding `ifconfig_em0="DHCP"` to `/etc/rc.conf`. More advanced networking setups are described in Chapter 31.

22.3 FreeBSD as a Host OS

For a number of years, FreeBSD was not officially supported as a host OS by any of the available virtualization solutions. Some people were using older and mostly obsolete versions of **VMware** (like `emulators/vmware3`), which utilized the Linux binary compatibility layer. Shortly after the release of FreeBSD 7.2, the Open Source Edition (OSE) of Sun's **VirtualBox™** appeared in the Ports Collection as a native FreeBSD program.

VirtualBox is an actively developed, complete virtualization package, that is available for most operating systems including Windows, Mac OS, Linux and FreeBSD. It is equally capable at running Windows or UNIX like guests. It comes in two flavors, an open source and a proprietary edition. From the user's point of view, perhaps the most important limitation of the OSE is the lack of USB support. Other differences may be found in the "Editions" page of the **VirtualBox** wiki, at http://www.virtualbox.org/wiki/Editions. Currently, only the OSE is available for FreeBSD.

22.3.1 Installing VirtualBox™

VirtualBox is available as a FreeBSD port in `emulators/virtualbox-ose`, and may be installed using the following commands:

```
# cd /usr/ports/emulators/virtualbox-ose
# make install clean
```

One useful option in the configuration dialog is the `GuestAdditions` suite of programs. These provide a number of useful features in guest operating systems, like mouse pointer integration (allowing the mouse to be shared between host and guest without the need to press a special keyboard shortcut to switch) and faster video rendering, especially in Windows guests. The guest additions are available in the Devices menu, after the installation of the guest OS is finished.

A few configuration changes are needed before **VirtualBox** is started for the first time. The port installs a kernel module in `/boot/modules` which must be loaded into the running kernel:

```
# kldload vboxdrv
```

To ensure the module always gets loaded after a reboot, add the following line to `/boot/loader.conf`:

```
vboxdrv_load="YES"
```

Versions of **VirtualBox** prior to 3.1.2 require the `proc` file system to be mounted. This is not needed in recent versions, which utilize the functions provided by the sysctl(3) library.

When using an older version of the port, follow the instructions below to make sure `proc` is mounted properly:

```
# mount -t procfs proc /proc
```

To allow this setting to persist reboots, the following line is needed in `/etc/fstab`:

```
proc /proc procfs rw 0 0
```

> **Note:** If an error message similar to the following is observed when **VirtualBox** is run from the terminal:
>
> ```
> VirtualBox: supR3HardenedExecDir: couldn't read "", errno=2 cchLink=-1
> ```
>
> The most likely culprit will be the `proc` file system. Please use the `mount` command to check whether it is mounted properly.

The `vboxusers` group is created during the installation of **VirtualBox**. All users that need access to **VirtualBox** will have to be added as members of this group. The `pw` command may be used to add new members:

```
# pw groupmod vboxusers -m yourusername
```

To launch **VirtualBox**, either select the Sun VirtualBox item from your graphic environment's menu, or type the following in a terminal:

```
% VirtualBox
```

For more information on configuring and using **VirtualBox**, please visit the official website at http://www.virtualbox.org. As the FreeBSD port is very recent, it is under heavy development. For the latest information and troubleshooting instructions, please visit the relevant page in the FreeBSD wiki, at http://wiki.FreeBSD.org/VirtualBox.

Chapter 23
Localization - I18N/L10N Usage and Setup

23.1 Synopsis

FreeBSD is a very distributed project with users and contributors located all over the world. This chapter discusses the internationalization and localization features of FreeBSD that allow non-English speaking users to get real work done. There are many aspects of the i18n implementation in both the system and application levels, so where applicable we refer the reader to more specific sources of documentation.

After reading this chapter, you will know:

- How different languages and locales are encoded on modern operating systems.
- How to set the locale for your login shell.
- How to configure your console for non-English languages.
- How to use X Window System effectively with different languages.
- Where to find more information about writing i18n-compliant applications.

Before reading this chapter, you should:

- Know how to install additional third-party applications (Chapter 4).

23.2 The Basics

23.2.1 What Is I18N/L10N?

Developers shortened internationalization into the term I18N, counting the number of letters between the first and the last letters of internationalization. L10N uses the same naming scheme, coming from “localization”. Combined together, I18N/L10N methods, protocols, and applications allow users to use languages of their choice.

I18N applications are programmed using I18N kits under libraries. It allows for developers to write a simple file and translate displayed menus and texts to each language. We strongly encourage programmers to follow this convention.

23.2.2 Why Should I Use I18N/L10N?

I18N/L10N is used whenever you wish to either view, input, or process data in non-English languages.

23.2.3 What Languages Are Supported in the I18N Effort?

I18N and L10N are not FreeBSD specific. Currently, one can choose from most of the major languages of the World, including but not limited to: Chinese, German, Japanese, Korean, French, Russian, Vietnamese and others.

23.3 Using Localization

In all its splendor, I18N is not FreeBSD-specific and is a convention. We encourage you to help FreeBSD in following this convention.

Localization settings are based on three main terms: Language Code, Country Code, and Encoding. Locale names are constructed from these parts as follows:

```
LanguageCode_CountryCode.Encoding
```

23.3.1 Language and Country Codes

In order to localize a FreeBSD system to a specific language (or any other I18N-supporting UNIX like systems), the user needs to find out the codes for the specific country and language (country codes tell applications what variation of given language to use). In addition, web browsers, SMTP/POP servers, web servers, etc. make decisions based on them. The following are examples of language/country codes:

Language/Country Code	Description
en_US	English - United States
ru_RU	Russian for Russia
zh_TW	Traditional Chinese for Taiwan

23.3.2 Encodings

Some languages use non-ASCII encodings that are 8-bit, wide or multibyte characters, see multibyte(3) for more details. Older applications do not recognize them and mistake them for control characters. Newer applications usually do recognize 8-bit characters. Depending on the implementation, users may be required to compile an application with wide or multibyte characters support, or configure it correctly. To be able to input and process wide or multibyte characters, the FreeBSD Ports Collection (http://www.FreeBSD.org/ports/index.html) has provided each language with different programs. Refer to the I18N documentation in the respective FreeBSD Port.

Specifically, the user needs to look at the application documentation to decide on how to configure it correctly or to pass correct values into the configure/Makefile/compiler.

Some things to keep in mind are:

- Language specific single C chars character sets (see multibyte(3)), e.g. ISO8859-1, ISO8859-15, KOI8-R, CP437.
- Wide or multibyte encodings, e.g. EUC, Big5.

You can check the active list of character sets at the IANA Registry (http://www.iana.org/assignments/character-sets).

> **Note:** FreeBSD use X11-compatible locale encodings instead.

23.3.3 I18N Applications

In the FreeBSD Ports and Package system, I18N applications have been named with `I18N` in their names for easy identification. However, they do not always support the language needed.

23.3.4 Setting Locale

Usually it is sufficient to export the value of the locale name as `LANG` in the login shell. This could be done in the user's `~/.login_conf` file or in the startup file of the user's shell (`~/.profile`, `~/.bashrc`, `~/.cshrc`). There is no need to set the locale subsets such as `LC_CTYPE`, `LC_CTIME`. Please refer to language-specific FreeBSD documentation for more information.

You should set the following two environment variables in your configuration files:

- `LANG` for POSIX setlocale(3) family functions
- `MM_CHARSET` for applications' MIME character set

This includes the user shell configuration, the specific application configuration, and the X11 configuration.

23.3.4.1 Setting Locale Methods

There are two methods for setting locale, and both are described below. The first (recommended one) is by assigning the environment variables in login class, and the second is by adding the environment variable assignments to the system's shell startup file.

23.3.4.1.1 Login Classes Method

This method allows environment variables needed for locale name and MIME character sets to be assigned once for every possible shell instead of adding specific shell assignments to each shell's startup file. User Level Setup can be done by an user himself and Administrator Level Setup require superuser privileges.

23.3.4.1.1.1 User Level Setup

Here is a minimal example of a `.login_conf` file in user's home directory which has both variables set for Latin-1 encoding:

```
me:\
 :charset=ISO-8859-1:\
 :lang=de_DE.ISO8859-1:
```

Here is an example of a `.login_conf` that sets the variables for Traditional Chinese in BIG-5 encoding. Notice the many more variables set because some software does not respect locale variables correctly for Chinese, Japanese, and Korean.

```
#Users who do not wish to use monetary units or time formats
#of Taiwan can manually change each variable
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": #Set gcin as the XIM Input Server
```

See Administrator Level Setup and login.conf(5) for more details.

23.3.4.1.1.2 Administrator Level Setup

Verify that the user's login class in `/etc/login.conf` sets the correct language. Make sure these settings appear in `/etc/login.conf`:

```
language_name|Account Type Description:\
 :charset=MIME_charset:\
 :lang=locale_name:\
 :tc=default:
```

So sticking with our previous example using Latin-1, it would look like this:

```
german|German Users Accounts:\
 :charset=ISO-8859-1:\
 :lang=de_DE.ISO8859-1:\
 :tc=default:
```

Before changing users Login Classes execute the following command:

```
# cap_mkdb /etc/login.conf
```

to make new configuration in `/etc/login.conf` visible to the system.

Changing Login Classes with vipw(8)

Use `vipw` to add new users, and make the entry look like this:

```
user:password:1111:11:language:0:0:User Name:/home/user:/bin/sh
```

Changing Login Classes with adduser(8)

Use `adduser` to add new users, and do the following:

- Set `defaultclass = language` in `/etc/adduser.conf`. Keep in mind you must enter a `default` class for all users of other languages in this case.
- An alternative variant is answering the specified language each time that

  ```
  Enter login class: default []:
  ```

  appears from adduser(8).
- Another alternative is to use the following for each user of a different language that you wish to add:

```
# adduser -class language
```

Changing Login Classes with pw(8)

If you use pw(8) for adding new users, call it in this form:

```
# pw useradd user_name -L language
```

23.3.4.1.2 Shell Startup File Method

> **Note:** This method is not recommended because it requires a different setup for each possible shell program chosen. Use the Login Class Method instead.

To add the locale name and MIME character set, just set the two environment variables shown below in the `/etc/profile` and/or `/etc/csh.login` shell startup files. We will use the German language as an example below:

In `/etc/profile`:

```
LANG=de_DE.ISO8859-1; export LANG
MM_CHARSET=ISO-8859-1; export MM_CHARSET
```

Or in `/etc/csh.login`:

```
setenv LANG de_DE.ISO8859-1
setenv MM_CHARSET ISO-8859-1
```

Alternatively, you can add the above instructions to `/usr/share/skel/dot.profile` (similar to what was used in `/etc/profile` above), or `/usr/share/skel/dot.login` (similar to what was used in `/etc/csh.login` above).

For X11:

In `$HOME/.xinitrc`:

```
LANG=de_DE.ISO8859-1; export LANG
```

Or:

```
setenv LANG de_DE.ISO8859-1
```

Depending on your shell (see above).

23.3.5 Console Setup

For all single C chars character sets, set the correct console fonts in `/etc/rc.conf` for the language in question with:

```
font8x16=font_name
font8x14=font_name
font8x8=font_name
```

The *font_name* here is taken from the `/usr/share/syscons/fonts` directory, without the `.fnt` suffix.

If required, set the keymap and screenmap for your single C chars character set through `sysinstall`. Once inside **sysinstall**, choose Configure, then Console. Alternatively, you can add the following to `/etc/rc.conf`:

```
scrnmap=screenmap_name
keymap=keymap_name
keychange="fkey_number sequence"
```

The *screenmap_name* here is taken from the `/usr/share/syscons/scrnmaps` directory, without the `.scm` suffix. A screenmap with a corresponding mapped font is usually needed as a workaround for expanding bit 8 to bit 9 on a VGA adapter's font character matrix in pseudographics area, i.e., to move letters out of that area if screen font uses a bit 8 column.

If you have the **moused** daemon enabled by setting the following in your `/etc/rc.conf`:

```
moused_enable="YES"
```

then examine the mouse cursor information in the next paragraph.

By default the mouse cursor of the syscons(4) driver occupies the 0xd0-0xd3 range in the character set. If your language uses this range, you need to move the cursor's range outside of it. To enable the workaround for FreeBSD, add the following line to `/etc/rc.conf`:

```
mousechar_start=3
```

The *keymap_name* here is taken from the `/usr/share/syscons/keymaps` directory, without the `.kbd` suffix. If you are uncertain which keymap to use, you use can kbdmap(1) to test keymaps without rebooting.

The `keychange` is usually needed to program function keys to match the selected terminal type because function key sequences cannot be defined in the key map.

Also be sure to set the correct console terminal type in `/etc/ttys` for all `ttyv*` entries. Current pre-defined correspondences are:

Character Set	Terminal Type
ISO8859-1 or ISO8859-15	`cons25l1`
ISO8859-2	`cons25l2`
ISO8859-7	`cons25l7`
KOI8-R	`cons25r`
KOI8-U	`cons25u`
CP437 (VGA default)	`cons25`
US-ASCII	`cons25w`

For wide or multibyte characters languages, use the correct FreeBSD port in your `/usr/ports/`*language* directory. Some ports appear as console while the system sees it as serial vtty's, hence you must reserve enough vtty's for both X11 and the pseudo-serial console. Here is a partial list of applications for using other languages in console:

Language	Location
Traditional Chinese (BIG-5)	`chinese/big5con`

Language	Location
Japanese	`japanese/kon2-16dot` or `japanese/mule-freewnn`
Korean	`korean/han`

23.3.6 X11 Setup

Although X11 is not part of the FreeBSD Project, we have included some information here for FreeBSD users. For more details, refer to the Xorg web site (http://www.x.org/) or whichever X11 Server you use.

In `~/.Xresources`, you can additionally tune application specific I18N settings (e.g., fonts, menus, etc.).

23.3.6.1 Displaying Fonts

Install **Xorg** server (`x11-servers/xorg-server`), then install the language TrueType fonts. Setting the correct locale should allow you to view your selected language in menus and such.

23.3.6.2 Inputting Non-English Characters

The X11 Input Method (XIM) Protocol is a new standard for all X11 clients. All X11 applications should be written as XIM clients that take input from XIM Input servers. There are several XIM servers available for different languages.

23.3.7 Printer Setup

Some single C chars character sets are usually hardware coded into printers. Wide or multibyte character sets require special setup and we recommend using **apsfilter**. You may also convert the document to PostScript or PDF formats using language specific converters.

23.3.8 Kernel and File Systems

The FreeBSD fast filesystem (FFS) is 8-bit clean, so it can be used with any single C chars character set (see multibyte(3)), but there is no character set name stored in the filesystem; i.e., it is raw 8-bit and does not know anything about encoding order. Officially, FFS does not support any form of wide or multibyte character sets yet. However, some wide or multibyte character sets have independent patches for FFS enabling such support. They are only temporary unportable solutions or hacks and we have decided to not include them in the source tree. Refer to respective languages' web sites for more information and the patch files.

The FreeBSD MS-DOS filesystem has the configurable ability to convert between MS-DOS, Unicode character sets and chosen FreeBSD filesystem character sets. See mount_msdosfs(8) for details.

23.4 Compiling I18N Programs

Many FreeBSD Ports have been ported with I18N support. Some of them are marked with -I18N in the port name. These and many other programs have built in support for I18N and need no special consideration.

However, some applications such as **MySQL** need to have their `Makefile` configured with the specific charset. This is usually done in the `Makefile` or done by passing a value to **configure** in the source.

23.5 Localizing FreeBSD to Specific Languages

23.5.1 Russian Language (KOI8-R Encoding)

For more information about KOI8-R encoding, see the KOI8-R References (Russian Net Character Set) (http://koi8.pp.ru/).

23.5.1.1 Locale Setup

Put the following lines into your `~/.login_conf` file:

```
me:My Account:\
 :charset=KOI8-R:\
 :lang=ru_RU.KOI8-R:
```

See earlier in this chapter for examples of setting up the locale.

23.5.1.2 Console Setup

- Add the following line to your `/etc/rc.conf` file:

  ```
  mousechar_start=3
  ```

- Also, use following settings in `/etc/rc.conf`:

  ```
  keymap="ru.koi8-r"
  scrnmap="koi8-r2cp866"
  font8x16="cp866b-8x16"
  font8x14="cp866-8x14"
  font8x8="cp866-8x8"
  ```

- For each `ttyv*` entry in `/etc/ttys`, use `cons25r` as the terminal type.

See earlier in this chapter for examples of setting up the console.

23.5.1.3 Printer Setup

Since most printers with Russian characters come with hardware code page CP866, a special output filter is needed to convert from KOI8-R to CP866. Such a filter is installed by default as `/usr/libexec/lpr/ru/koi2alt`. A Russian printer `/etc/printcap` entry should look like:

```
lp|Russian local line printer:\
 :sh:of=/usr/libexec/lpr/ru/koi2alt:\
```

```
:lp=/dev/lpt0:sd=/var/spool/output/lpd:lf=/var/log/lpd-errs:
```

See printcap(5) for a detailed description.

23.5.1.4 MS-DOS FS and Russian Filenames

The following example fstab(5) entry enables support for Russian filenames in mounted MS-DOS filesystems:

```
/dev/ad0s2      /dos/c  msdos   rw,-Wkoi2dos,-Lru_RU.KOI8-R 0 0
```

The option `-L` selects the locale name used, and `-W` sets the character conversion table. To use the `-W` option, be sure to mount `/usr` before the MS-DOS partition because the conversion tables are located in `/usr/libdata/msdosfs`. For more information, see the mount_msdosfs(8) manual page.

23.5.1.5 X11 Setup

1. Do non-X locale setup first as described.
2. If you use **Xorg**, install `x11-fonts/xorg-fonts-cyrillic` package.

   Check the `"Files"` section in your `/etc/X11/xorg.conf` file. The following line must be added *before* any other `FontPath` entries:

   ```
   FontPath   "/usr/local/lib/X11/fonts/cyrillic"
   ```

   Note: See ports for more cyrillic fonts.

3. To activate a Russian keyboard, add the following to the `"Keyboard"` section of your `xorg.conf` file:

   ```
   Option "XkbLayout"   "us,ru"
   Option "XkbOptions"  "grp:toggle"
   ```

   Also make sure that `XkbDisable` is turned off (commented out) there.

   For `grp:toggle` the RUS/LAT switch will be **Right Alt**, for `grp:ctrl_shift_toggle` switch will be **Ctrl**+**Shift**. For `grp:caps_toggle` the RUS/LAT switch will be **CapsLock**. The old **CapsLock** function is still available via **Shift**+**CapsLock** (in LAT mode only). `grp:caps_toggle` does not work in **Xorg** for unknown reason.

   If you have “Windows” keys on your keyboard, and notice that some non-alphabetical keys are mapped incorrectly in RUS mode, add the following line in your `xorg.conf` file:

   ```
   Option "XkbVariant" ",winkeys"
   ```

   Note: The Russian XKB keyboard may not work with non-localized applications.

Note: Minimally localized applications should call a `XtSetLanguageProc (NULL, NULL, NULL);` function early in the program.

See KOI8-R for X Window (http://koi8.pp.ru/xwin.html) for more instructions on localizing X11 applications.

23.5.2 Traditional Chinese Localization for Taiwan

The FreeBSD-Taiwan Project has an Chinese HOWTO for FreeBSD at http://netlab.cse.yzu.edu.tw/~statue/freebsd/zh-tut/ using many Chinese ports. Current editor for the `FreeBSD Chinese HOWTO` is Shen Chuan-Hsing <`statue@freebsd.sinica.edu.tw`>.

Chuan-Hsing Shen <`statue@freebsd.sinica.edu.tw`> has created the Chinese FreeBSD Collection (CFC) (http://netlab.cse.yzu.edu.tw/~statue/cfc/) using FreeBSD-Taiwan's `zh-L10N-tut`. The packages and the script files are available at ftp://freebsd.csie.nctu.edu.tw/pub/taiwan/CFC/.

23.5.3 German Language Localization (for All ISO 8859-1 Languages)

Slaven Rezic <`eserte@cs.tu-berlin.de`> wrote a tutorial on using umlauts on a FreeBSD machine. The tutorial is written in German and is available at http://user.cs.tu-berlin.de/~eserte/FreeBSD/doc/umlaute/umlaute.html.

23.5.4 Greek Language Localization

Nikos Kokkalis <`nickkokkalis@gmail.com`> has written a complete article on Greek support in FreeBSD. It is available as part of the official FreeBSD Greek documentation, in http://www.freebsd.org/doc/el_GR.ISO8859-7/articles/greek-language-support/index.html (http://www.FreeBSD.org/doc/el_GR.ISO8859-7/articles/greek-language-support/index.html). Please note this is in Greek *only*.

23.5.5 Japanese and Korean Language Localization

For Japanese, refer to http://www.jp.FreeBSD.org/, and for Korean, refer to http://www.kr.FreeBSD.org/.

23.5.6 Non-English FreeBSD Documentation

Some FreeBSD contributors have translated parts of FreeBSD documentation to other languages. They are available through links on the main site (http://www.FreeBSD.org/index.html) or in `/usr/share/doc`.

Chapter 24
Updating and Upgrading FreeBSD

24.1 Synopsis

FreeBSD is under constant development between releases. Some people prefer to use the officially released versions, while others prefer to keep in sync with the latest developments. However, even official releases are often updated with security and other critical fixes. Regardless of the version used, FreeBSD provides all necessary tools to keep your system updated, and also allows for easy upgrades between versions. This chapter will help you decide if you want to track the development system, or stick with one of the released versions. The basic tools for keeping your system up to date are also presented.

After reading this chapter, you will know:

- What utilities may be used to update the system and the Ports Collection.
- How to keep your system up to date with **freebsd-update**, **CVSup**, **CVS**, or **CTM**.
- How to compare the state of an installed system against a known pristine copy.
- How to keep your documentation up to date with **CVSup** or documentation ports.
- The difference between the two development branches: FreeBSD-STABLE and FreeBSD-CURRENT.
- How to rebuild and reinstall the entire base system with `make buildworld` (etc).

Before reading this chapter, you should:

- Properly set up your network connection (Chapter 31).
- Know how to install additional third-party software (Chapter 4).

> **Note:** Throughout this chapter, the `cvsup` command is used to obtain and update FreeBSD sources. To use it, you will need to install the port or the package for `net/cvsup` (if you do not want to install the graphical `cvsup` client, you can just install the port `net/cvsup-without-gui`). You may wish to substitute this with csup(1), which is part of the base system.

24.2 FreeBSD Update

Applying security patches is an important part of maintaining computer software, especially the operating system. For the longest time on FreeBSD this process was not an easy one. Patches had to be applied to the source code, the code rebuilt into binaries, and then the binaries had to be re-installed.

This is no longer the case as FreeBSD now includes a utility simply called `freebsd-update`. This utility provides two separate functions. First, it allows for binary security and errata updates to be applied to the FreeBSD base system without the build and install requirements. Second, the utility supports minor and major release upgrades.

> **Note:** Binary updates are available for all architectures and releases currently supported by the security team. Before updating to a new release, the current release announcements should be reviewed as they may contain important information pertinent to the desired release. These announcements may be viewed at the following link: http://www.FreeBSD.org/releases/.

If a `crontab` utilizing the features of `freebsd-update` exists, it must be disabled before the following operation is started.

24.2.1 The Configuration File

Some users may wish to tweak the default configuration file in `/etc/freebsd-update.conf`, allowing better control of the process. The options are very well documented, but the following few may require a bit more explanation:

```
# Components of the base system which should be kept updated.
Components src world kernel
```

This parameter controls what parts of FreeBSD will be kept up to date. The default is to update the source code, the entire base system, and the kernel. Components are the same as those available during the install, for instance, adding "world/games" here would allow game patches to be applied. Using "src/bin" would allow the source code in `src/bin` to be updated.

The best option is to leave this at the default as changing it to include specific items will require the user to list every item they prefer to be updated. This could have disastrous consequences as source code and binaries may become out of sync.

```
# Paths which start with anything matching an entry in an IgnorePaths
# statement will be ignored.
IgnorePaths
```

Add paths, such as `/bin` or `/sbin` to leave these specific directories untouched during the update process. This option may be used to prevent `freebsd-update` from overwriting local modifications.

```
# Paths which start with anything matching an entry in an UpdateIfUnmodified
# statement will only be updated if the contents of the file have not been
# modified by the user (unless changes are merged; see below).
UpdateIfUnmodified /etc/ /var/ /root/ /.cshrc /.profile
```

Update configuration files in the specified directories only if they have not been modified. Any changes made by the user will invalidate the automatic updating of these files. There is another option, `KeepModifiedMetadata`, which will instruct `freebsd-update` to save the changes during the merge.

```
# When upgrading to a new FreeBSD release, files which match MergeChanges
# will have any local changes merged into the version from the new release.
MergeChanges /etc/ /var/named/etc/
```

List of directories with configuration files that `freebsd-update` should attempt merges in. The file merge process is a series of diff(1) patches similar to mergemaster(8) with fewer options, the merges are either accepted, open an editor, or `freebsd-update` will abort. When in doubt, backup `/etc` and just accept the merges. See Section 24.7.11.1 for more information about the `mergemaster` command.

```
# Directory in which to store downloaded updates and temporary
# files used by FreeBSD Update.
# WorkDir /var/db/freebsd-update
```

This directory is where all patches and temporary files will be placed. In cases where the user is doing a version upgrade, this location should have a least a gigabyte of disk space available.

```
# When upgrading between releases, should the list of Components be
# read strictly (StrictComponents yes) or merely as a list of components
# which *might* be installed of which FreeBSD Update should figure out
# which actually are installed and upgrade those (StrictComponents no)?
# StrictComponents no
```

When set to `yes`, `freebsd-update` will assume that the `Components` list is complete and will not attempt to make changes outside of the list. Effectively, `freebsd-update` will attempt to update every file which belongs to the `Components` list.

24.2.2 Security Patches

Security patches are stored on a remote machine and may be downloaded and installed using the following command:

```
# freebsd-update fetch
# freebsd-update install
```

If any kernel patches have been applied the system will need a reboot. If all went well the system should be patched and `freebsd-update` may be run as a nightly cron(8) job. An entry in `/etc/crontab` would be sufficient to accomplish this task:

```
@daily                                  root    freebsd-update cron
```

This entry states that once every day, the `freebsd-update` utility will be run. In this way, using the `cron` argument, `freebsd-update` will only check if updates exist. If patches exist, they will automatically be downloaded to the local disk but not applied. The `root` user will be sent an email so they may install them manually.

If anything went wrong, `freebsd-update` has the ability to roll back the last set of changes with the following command:

```
# freebsd-update rollback
```

Once complete, the system should be restarted if the kernel or any kernel modules were modified. This will allow FreeBSD to load the new binaries into memory.

The `freebsd-update` utility can automatically update the `GENERIC` kernel only. If a custom kernel is in use, it will have to be rebuilt and reinstalled after `freebsd-update` finishes installing the rest of the updates. However, `freebsd-update` will detect and update the `GENERIC` kernel in `/boot/GENERIC` (if it exists), even if it is not the current (running) kernel of the system.

> **Note:** It is a good idea to always keep a copy of the `GENERIC` kernel in `/boot/GENERIC`. It will be helpful in diagnosing a variety of problems, and in performing version upgrades using `freebsd-update` as described in Section 24.2.3.

Unless the default configuration in `/etc/freebsd-update.conf` has been changed, `freebsd-update` will install the updated kernel sources along with the rest of the updates. Rebuilding and reinstalling your new custom kernel can then be performed in the usual way.

> **Note:** The updates distributed via `freebsd-update`, do not always involve the kernel. It will not be necessary to rebuild your custom kernel if the kernel sources have not been modified by the execution of `freebsd-update install`. However, `freebsd-update` will always update the `/usr/src/sys/conf/newvers.sh` file. The current patch level (as indicated by the `-p` number reported by `uname -r`) is obtained from this file. Rebuilding your custom kernel, even if nothing else changed, will allow uname(1) to accurately report the current patch level of the system. This is particularly helpful when maintaining multiple systems, as it allows for a quick assessment of the updates installed in each one.

24.2.3 Major and Minor Upgrades

This process will remove old object files and libraries which will break most third party applications. It is recommended that all installed ports either be removed and re-installed or upgraded later using the `ports-mgmt/portupgrade` utility. Most users will want to run a test build using the following command:

```
# portupgrade -af
```

This will ensure everything will be re-installed correctly. Note that setting the `BATCH` environment variable to `yes` will answer `yes` to any prompts during this process, removing the need for manual intervention during the build process.

If a custom kernel is in use, the upgrade process is slightly more involved. A copy of the `GENERIC` kernel is needed, and it should be placed in `/boot/GENERIC`. If the `GENERIC` kernel is not already present in the system, it may be obtained using one of the following methods:

- If a custom kernel has only been built once, the kernel in `/boot/kernel.old` is actually the `GENERIC` one. Simply rename this directory to `/boot/GENERIC`.
- Assuming physical access to the machine is possible, a copy of the `GENERIC` kernel can be installed from the CD-ROM media. Insert your installation disc and use the following commands:

  ```
  # mount /cdrom
  # cd /cdrom/X.Y-RELEASE/kernels
  # ./install.sh GENERIC
  ```

Replace *X.Y-RELEASE* with the actual version of the release you are using. The GENERIC kernel will be installed in /boot/GENERIC by default.

- Failing all the above, the GENERIC kernel may be rebuilt and installed from the sources:

```
# cd /usr/src
# env DESTDIR=/boot/GENERIC make kernel
# mv /boot/GENERIC/boot/kernel/* /boot/GENERIC
# rm -rf /boot/GENERIC/boot
```

For this kernel to be picked up as GENERIC by freebsd-update, the GENERIC configuration file must not have been modified in any way. It is also suggested that it is built without any other special options (preferably with an empty /etc/make.conf).

Rebooting to the GENERIC kernel is not required at this stage.

Major and minor version updates may be performed by providing freebsd-update with a release version target, for example, the following command will update to FreeBSD 8.1:

```
# freebsd-update -r 8.1-RELEASE upgrade
```

After the command has been received, freebsd-update will evaluate the configuration file and current system in an attempt to gather the information necessary to update the system. A screen listing will display what components have been detected and what components have not been detected. For example:

```
Looking up update.FreeBSD.org mirrors... 1 mirrors found.
Fetching metadata signature for 8.0-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
```

At this point, freebsd-update will attempt to download all files required for the upgrade. In some cases, the user may be prompted with questions regarding what to install or how to proceed.

When using a custom kernel, the above step will produce a warning similar to the following:

```
WARNING: This system is running a "MYKERNEL" kernel, which is not a
kernel configuration distributed as part of FreeBSD 8.0-RELEASE.
This kernel will not be updated: you MUST update the kernel manually
before running "/usr/sbin/freebsd-update install"
```

This warning may be safely ignored at this point. The updated GENERIC kernel will be used as an intermediate step in the upgrade process.

After all patches have been downloaded to the local system, they will then be applied. This process may take a while depending on the speed and workload of the machine. Configuration files will then be merged — this part of the process requires some user intervention as a file may be merged or an editor may appear on screen for a manual merge. The results of every successful merge will be shown to the user as the process continues. A failed or ignored merge will cause the process to abort. Users may wish to make a backup of `/etc` and manually merge important files, such as `master.passwd` or `group` at a later time.

> **Note:** The system is not being altered yet, all patching and merging is happening in another directory. When all patches have been applied successfully, all configuration files have been merged and it seems the process will go smoothly, the changes will need to be committed by the user.

Once this process is complete, the upgrade may be committed to disk using the following command.

```
# freebsd-update install
```

The kernel and kernel modules will be patched first. At this point the machine must be rebooted. If the system was running with a custom kernel, use the nextboot(8) command to set the kernel for the next boot to `/boot/GENERIC` (which was updated):

```
# nextboot -k GENERIC
```

> **Warning:** Before rebooting with the `GENERIC` kernel, make sure it contains all drivers required for your system to boot properly (and connect to the network, if the machine that is being updated is accessed remotely). In particular, if the previously running custom kernel contained built-in functionality usually provided by kernel modules, make sure to temporarily load these modules into the `GENERIC` kernel using the `/boot/loader.conf` facility. You may also wish to disable non-essential services, disk and network mounts, etc. until the upgrade process is complete.

The machine should now be restarted with the updated kernel:

```
# shutdown -r now
```

Once the system has come back online, `freebsd-update` will need to be started again. The state of the process has been saved and thus, `freebsd-update` will not start from the beginning, but will remove all old shared libraries and object files. To continue to this stage, issue the following command:

```
# freebsd-update install
```

> **Note:** Depending on whether any libraries version numbers got bumped, there may only be two install phases instead of three.

All third party software will now need to be rebuilt and re-installed. This is required as installed software may depend on libraries which have been removed during the upgrade process. The `ports-mgmt/portupgrade` command may be used to automate this process. The following commands may be used to begin this process:

```
# 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
```

Once this has completed, finish the upgrade process with a final call to `freebsd-update`. Issue the following command to tie up all loose ends in the upgrade process:

```
# freebsd-update install
```

If the `GENERIC` kernel was temporarily used, this is the time to build and install a new custom kernel in the usual way.

Reboot the machine into the new FreeBSD version. The process is complete.

24.2.4 System State Comparison

The `freebsd-update` utility may be used to test the state of the installed FreeBSD version against a known good copy. This option evaluates the current version of system utilities, libraries, and configuration files. To begin the comparison, issue the following command:

```
# freebsd-update IDS >> outfile.ids
```

> **Warning:** While the command name is IDS it should in no way be a replacement for an intrusion detection system such as `security/snort`. As `freebsd-update` stores data on disk, the possibility of tampering is evident. While this possibility may be reduced by using the `kern.securelevel` setting and storing the `freebsd-update` data on a read only file system when not in use, a better solution would be to compare the system against a secure disk, such as a DVD or securely stored external USB disk device.

The system will now be inspected, and a list of files along with their sha256(1) hash values, both the known value in the release and the current installed value, will be printed. This is why the output has been sent to the `outfile.ids` file. It scrolls by too quickly for eye comparisons, and soon it fills up the console buffer.

These lines are also extremely long, but the output format may be parsed quite easily. For instance, to obtain a list of all files different from those in the release, issue the following command:

```
# cat outfile.ids | awk '{ print $1 }' | more
/etc/master.passwd
/etc/motd
/etc/passwd
/etc/pf.conf
```

This output has been truncated, many more files exist. Some of these files have natural modifications, the `/etc/passwd` has been modified because users have been added to the system. In some cases, there may be other files, such as kernel modules, which differ as `freebsd-update` may have updated them. To exclude specific files or directories, add them to the `IDSIgnorePaths` option in `/etc/freebsd-update.conf`.

This system may be used as part of an elaborate upgrade method, aside from the previously discussed version.

24.3 Portsnap: A Ports Collection Update Tool

The base system of FreeBSD includes a utility for updating the Ports Collection too: the portsnap(8) utility. Upon execution, it will connect to a remote site, verify the secure key, and download a new copy of the Ports Collection. The key is used to verify the integrity of all downloaded files, ensuring they have not been modified in-flight. To download the latest Ports Collection files, issue the following command:

```
# 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.
```

What this example shows is that portsnap(8) has found and verified several patches to the current ports data. This also indicates that the utility was run previously, if it was a first time run, the collection would have simply been downloaded.

When portsnap(8) successfully completes a `fetch` operation, the Ports Collection and subsequent patches exist on the local system that have passed verification. The first time `portsnap` is executed, you have to use `extract` to install the downloaded files:

```
# 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
...
```

To update an already installed Ports Collection use the command `portsnap update`:

```
# portsnap update
```

The process is now complete, and applications may be installed or upgraded using the updated Ports Collection.

The `fetch` and `extract` or `update` operations may be run consecutively, as shown in the following example:

```
# portsnap fetch update
```

This command will download the latest version of the Ports Collection and update your local version under `/usr/ports`.

24.4 Updating the Documentation Set

Besides the base system and the Ports Collection, documentation is an integral part of the FreeBSD operating system. While an up-to-date version of the FreeBSD Documentation Set is always available on the FreeBSD web site (http://www.freebsd.org/doc/), some users might have slow or no permanent network connectivity at all. Fortunately, there are several ways to update the documentation shipped with each release by maintaining a local copy of the latest FreeBSD Documentation Set.

24.4.1 Using CVSup to Update the Documentation

The sources and the installed copy of the FreeBSD documentation can be updated with **CVSup**, using a mechanism similar to the one employed for the base system sources (c.f. Section 24.7). This section describes:

- How to install the documentation toolchain, the tools that are required to rebuild the FreeBSD documentation from its source.
- How to download a copy of the documentation source at `/usr/doc`, using **CVSup**.
- How to rebuild the FreeBSD documentation from its source, and install it under `/usr/share/doc`.
- Some of the build options that are supported by the build system of the documentation, i.e. the options that build only some of the different language translations of the documentation or the options that select a specific output format.

24.4.2 Installing CVSup and the Documentation Toolchain

Rebuilding the FreeBSD documentation from source requires a fairly large collection of tools. These tools are not part of the FreeBSD base system, because they need a large amount of disk space and they are not useful to all FreeBSD users; they are only useful to those users that are actively writing new documentation for FreeBSD or are frequently updating their documentation from source.

All the required tools are available as part of the Ports Collection. The `textproc/docproj` port is a master port that has been developed by the FreeBSD Documentation Project, to ease the initial installation and future updates of these tools.

> **Note:** When no PostScript or PDF documentation required, one might consider installing the `textproc/docproj-nojadetex` port instead. This version of the documentation toolchain includes everything except the **teTeX** typesetting engine. **teTeX** is a very large collection of tools, so it may be quite sensible to omit its installation if PDF output is not really necessary.

For more information on installing and using **CVSup**, see Using CVSup.

24.4.3 Updating the Documentation Sources

The **CVSup** utility can fetch a clean copy of the documentation sources, using the `/usr/share/examples/cvsup/doc-supfile` file as a configuration template. The default update host is set to a placeholder value in `doc-supfile`, but cvsup(1) accepts a host name through the command line, so the documentation sources can be fetched from one of the **CVSup** servers by typing:

```
# cvsup -h cvsup.FreeBSD.org -g -L 2 /usr/share/examples/cvsup/doc-supfile
```

Change *cvsup.FreeBSD.org* to the nearest **CVSup** server. See Section A.6.7 for a complete listing of mirror sites.

The initial download of the documentation sources may take a while. Let it run until it completes.

Future updates of the documentation sources may be fetched by running the same command. The **CVSup** utility downloads and copies only the updates since the last time it ran, so every run of **CVSup** after the first complete run should be pretty fast.

After checking out the sources, an alternative way of updating the documentation is supported by the `Makefile` of the `/usr/doc` directory. By setting `SUP_UPDATE`, `SUPHOST` and `DOCSUPFILE` in the `/etc/make.conf` file, it is possible to run:

```
# cd /usr/doc
# make update
```

A typical set of these make(1) options for `/etc/make.conf` is:

```
SUP_UPDATE= yes
SUPHOST?= cvsup.freebsd.org
DOCSUPFILE?= /usr/share/examples/cvsup/doc-supfile
```

> **Note:** Setting the `SUPHOST` and `DOCSUPFILE` value with `?=` permits overriding them in the command-line of make. This is the recommended way of adding options to `make.conf`, to avoid having to edit the file every time a different option value has to be tested.

24.4.4 Tunable Options of the Documentation Sources

The updating and build system of the FreeBSD documentation supports a few options that ease the process of updating only parts of the documentation, or the build of specific translations. These options can be set either as system-wide options in the `/etc/make.conf` file, or as command-line options passed to the make(1) utility.

The following options are some of these:

`DOC_LANG`

The list of languages and encodings to build and install, e.g. `en_US.ISO8859-1` for the English documentation only.

`FORMATS`

A single format or a list of output formats to be built. Currently, `html`, `html-split`, `txt`, `ps`, `pdf`, and `rtf` are supported.

`SUPHOST`

The hostname of the **CVSup** server to use when updating.

`DOCDIR`

Where to install the documentation. It defaults to `/usr/share/doc`.

For more make variables supported as system-wide options in FreeBSD, see make.conf(5).

For more make variables supported by the build system of the FreeBSD documentation, please refer to the FreeBSD Documentation Project Primer for New Contributors (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/fdp-primer).

24.4.5 Installing the FreeBSD Documentation from Source

When an up-to-date snapshot of the documentation sources has been fetched in `/usr/doc`, everything is ready for an update of the installed documentation.

A full update of all the languages defined in the `DOC_LANG` makefile option may be done by typing:

```
# cd /usr/doc
# make install clean
```

If `make.conf` has been set up with the correct `DOCSUPFILE`, `SUPHOST` and `SUP_UPDATE` options, the install step may be combined with an update of the documentation sources by typing:

```
# cd /usr/doc
# make update install clean
```

If an update of only a specific language is desired, make(1) can be invoked in a language specific subdirectory of `/usr/doc`, i.e.:

```
# cd /usr/doc/en_US.ISO8859-1
# make update install clean
```

The output formats that will be installed may be specified by setting the `FORMATS` make variable, i.e.:

```
# cd /usr/doc
# make FORMATS='html html-split' install clean
```

24.4.6 Using Documentation Ports

In the previous section, we have presented a method for updating the FreeBSD documentation from sources. Source based updates may not be feasible or practical for all FreeBSD systems though. Building the documentation sources requires a fairly large collection of tools and utilities, the *documentation toolchain*, a certain level of familiarity with **CVS** and source checkouts from a repository, and a few manual steps to build the checked out sources. In this section, we describe an alternative way of updating the installed copies of the FreeBSD documentation; one that uses the Ports Collection and makes it possible to:

- Download and install pre-built snaphots of the documentation, without having to locally build anything (eliminating this way the need for an installation of the entire documentation toolchain).
- Download the documentation sources and build them through the ports framework (making the checkout and build steps a bit eaiser).

These two methods of updating the FreeBSD documentation are supported by a set of *documentation ports*, updated by the Documentation Engineering Team <doceng@FreeBSD.org> on a monthly basis. These are listed in the FreeBSD Ports Collection, under the virtual category named docs (http://www.freshports.org/docs/).

24.4.6.1 Building and Installing Documentation Ports

The documentation ports use the ports building framework to make documentation builds easier. They automate the process of checking out the documentation source, running make(1) with the appropriate environment settings and command-line options, and they make the installation or deinstallation of documentation as easy as the installation of any other FreeBSD port or package.

> **Note:** As an extra feature, when the documentation ports are built locally, they record a dependency to the *documentation toolchain* ports, so the latter is automatically installed too.

Organization of the documentation ports is as follows:

- There is a “master port”, `misc/freebsd-doc-en`, where the documentation port files can be found. It is the base of all documentation ports. By default, it builds the English documentation only.
- There is an “all in one port”, `misc/freebsd-doc-all`, and it builds and installs all documentation in all available languages.
- Finally, there is a “slave port” for each translation, e.g.: `misc/freebsd-doc-hu` for the Hungarian-language documents. All of them depend on the master port and install the translated documentation of the respective language.

To install a documentation port from source, issue the following commands (as `root`):

```
# cd /usr/ports/misc/freebsd-doc-en
# make install clean
```

This will build and install the English documentation in split HTML format (the same as used on http://www.FreeBSD.org) in the `/usr/local/share/doc/freebsd` directory.

24.4.6.1.1 Common Knobs and Options

There are many options for modifying the default behavior of the documentation ports. The following is just a short list:

`WITH_HTML`

Allows the build of the HTML format: a single HTML file per document. The formatted documentation is saved to a file called `article.html`, or `book.html`, as appropriate, plus images.

`WITH_PDF`

Allows the build of the Adobe Portable Document Format, for use with Adobe Acrobat Reader, **Ghostscript** or other PDF readers. The formatted documentation is saved to a file called `article.pdf` or `book.pdf`, as appropriate.

DOCBASE

Where to install the documentation. It defaults to /usr/local/share/doc/freebsd.

> **Note:** Notice that the default target directory differs from the directory used by the **CVSup** method. This is because we are installing a port, and ports are usually installed under the /usr/local directory. This can overriden, by adding the PREFIX variable.

Here is a brief example on how to use the variables mentioned above to install the Hungarian documentation in Portable Document Format:

```
# cd /usr/ports/misc/freebsd-doc-hu
# make -DWITH_PDF DOCBASE=share/doc/freebsd/hu install clean
```

24.4.6.2 Using Documentation Packages

Building the documentation ports from source, as described in the previous section, requires a local installation of the documentation toolchain and a bit of disk space for the build of the ports. When resources are not available to install the documentation toolchain, or because the build from sources would take too much disk space, it is still possible to install pre-built snapshots of the documentation ports.

The Documentation Engineering Team <doceng@FreeBSD.org> prepares monthly snapshots of the FreeBSD documentation packages. These binary packages can be used with any of the bundled package tools, like pkg_add(1), pkg_delete(1), and so on.

> **Note:** When binary packages are used, the FreeBSD documentation will be installed in *all* available formats for the given language.

For example, the following command will install the latest pre-built package of the Hungarian documentation:

```
# pkg_add -r hu-freebsd-doc
```

> **Note:** Packages have the following name format that differs from the corresponding port's name: *lang*-freebsd-doc. Here *lang* is the short format of the language code, i.e. hu for Hungarian, or zh_cn for Simplified Chinese.

24.4.6.3 Updating Documentation Ports

To update a previously installed documentation port, any tool suitable for updating ports is sufficient. For example, the following command updates the installed Hungarian documentation via the ports-mgmt/portupgrade tool by using packages only:

```
# portupgrade -PP hu-freebsd-doc
```

24.5 Tracking a Development Branch

There are two development branches to FreeBSD: FreeBSD-CURRENT and FreeBSD-STABLE. This section will explain a bit about each and describe how to keep your system up-to-date with each respective tree. FreeBSD-CURRENT will be discussed first, then FreeBSD-STABLE.

24.5.1 Staying Current with FreeBSD

As you read this, keep in mind that FreeBSD-CURRENT is the “bleeding edge” of FreeBSD development. FreeBSD-CURRENT users are expected to have a high degree of technical skill, and should be capable of solving difficult system problems on their own. If you are new to FreeBSD, think twice before installing it.

24.5.1.1 What Is FreeBSD-CURRENT?

FreeBSD-CURRENT is the latest working sources for FreeBSD. This includes work in progress, experimental changes, and transitional mechanisms that might or might not be present in the next official release of the software. While many FreeBSD developers compile the FreeBSD-CURRENT source code daily, there are periods of time when the sources are not buildable. These problems are resolved as expeditiously as possible, but whether or not FreeBSD-CURRENT brings disaster or greatly desired functionality can be a matter of which exact moment you grabbed the source code in!

24.5.1.2 Who Needs FreeBSD-CURRENT?

FreeBSD-CURRENT is made available for 3 primary interest groups:

1. Members of the FreeBSD community who are actively working on some part of the source tree and for whom keeping “current” is an absolute requirement.
2. Members of the FreeBSD community who are active testers, willing to spend time solving problems in order to ensure that FreeBSD-CURRENT remains as sane as possible. These are also people who wish to make topical suggestions on changes and the general direction of FreeBSD, and submit patches to implement them.
3. Those who merely wish to keep an eye on things, or to use the current sources for reference purposes (e.g. for *reading*, not running). These people also make the occasional comment or contribute code.

24.5.1.3 What Is FreeBSD-CURRENT *Not*?

1. A fast-track to getting pre-release bits because you heard there is some cool new feature in there and you want to be the first on your block to have it. Being the first on the block to get the new feature means that you are the first on the block to get the new bugs.
2. A quick way of getting bug fixes. Any given version of FreeBSD-CURRENT is just as likely to introduce new bugs as to fix existing ones.
3. In any way “officially supported”. We do our best to help people genuinely in one of the 3 “legitimate” FreeBSD-CURRENT groups, but we simply *do not have the time* to provide tech support. This is not because we are mean and nasty people who do not like helping people out (we would not even be doing FreeBSD if we were). We simply cannot answer hundreds messages a day *and* work on FreeBSD! Given the choice between improving FreeBSD and answering lots of questions on experimental code, the developers opt for the former.

24.5.1.4 Using FreeBSD-CURRENT

1. Join the freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current) and the svn-src-head (http://lists.FreeBSD.org/mailman/listinfo/svn-src-head) lists. This is not just a good idea, it is *essential*. If you are not on the *freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current)* list, you will not see the comments that people are making about the current state of the system and thus will probably end up stumbling over a lot of problems that others have already found and solved. Even more importantly, you will miss out on important bulletins which may be critical to your system's continued health.

   The svn-src-head (http://lists.FreeBSD.org/mailman/listinfo/svn-src-head) list will allow you to see the commit log entry for each change as it is made, along with any pertinent information on possible side-effects.

   To join these lists, or one of the others available go to http://lists.FreeBSD.org/mailman/listinfo and click on the list that you wish to subscribe to. Instructions on the rest of the procedure are available there. If you are interested in tracking changes for the whole source tree, we would recommend subscribing to the svn-src-all (http://lists.FreeBSD.org/mailman/listinfo/svn-src-all) list.

2. Grab the sources from a FreeBSD mirror site. You can do this in one of two ways:

   a. Use the cvsup program with the `supfile` named `standard-supfile` available from `/usr/share/examples/cvsup`. This is the most recommended method, since it allows you to grab the entire collection once and then only what has changed from then on. Many people run `cvsup` from `cron` and keep their sources up-to-date automatically. You have to customize the sample `supfile` above, and configure cvsup for your environment.

      Note: The sample `standard-supfile` is intended for tracking a specific security branch of FreeBSD, and not FreeBSD-CURRENT. You will need to edit this file and replace the following line:

      ```
      *default release=cvs tag=RELENG_X_Y
      ```

      With this one:

      ```
      *default release=cvs tag=.
      ```

      For a detailed explanation of usable tags, please refer to the Handbook's CVS Tags section.

   b. Use the **CTM** facility. If you have very bad connectivity (high price connections or only email access) **CTM** is an option. However, it is a lot of hassle and can give you broken files. This leads to it being rarely used, which again increases the chance of it not working for fairly long periods of time. We recommend using **CVSup** for anybody with a 9600 bps modem or faster connection.

3. If you are grabbing the sources to run, and not just look at, then grab *all* of FreeBSD-CURRENT, not just selected portions. The reason for this is that various parts of the source depend on updates elsewhere, and trying to compile just a subset is almost guaranteed to get you into trouble.

   Before compiling FreeBSD-CURRENT, read the `Makefile` in `/usr/src` carefully. You should at least install a new kernel and rebuild the world the first time through as part of the upgrading process. Reading the FreeBSD-CURRENT mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current) and `/usr/src/UPDATING` will keep you up-to-date on other bootstrapping procedures that sometimes become necessary as we move toward the next release.

4. Be active! If you are running FreeBSD-CURRENT, we want to know what you have to say about it, especially if you have suggestions for enhancements or bug fixes. Suggestions with accompanying code are received most enthusiastically!

24.5.2 Staying Stable with FreeBSD

24.5.2.1 What Is FreeBSD-STABLE?

FreeBSD-STABLE is our development branch from which major releases are made. Changes go into this branch at a different pace, and with the general assumption that they have first gone into FreeBSD-CURRENT for testing. This is *still* a development branch, however, and this means that at any given time, the sources for FreeBSD-STABLE may or may not be suitable for any particular purpose. It is simply another engineering development track, not a resource for end-users.

24.5.2.2 Who Needs FreeBSD-STABLE?

If you are interested in tracking or contributing to the FreeBSD development process, especially as it relates to the next "point" release of FreeBSD, then you should consider following FreeBSD-STABLE.

While it is true that security fixes also go into the FreeBSD-STABLE branch, you do not *need* to track FreeBSD-STABLE to do this. Every security advisory for FreeBSD explains how to fix the problem for the releases it affects [1], and tracking an entire development branch just for security reasons is likely to bring in a lot of unwanted changes as well.

Although we endeavor to ensure that the FreeBSD-STABLE branch compiles and runs at all times, this cannot be guaranteed. In addition, while code is developed in FreeBSD-CURRENT before including it in FreeBSD-STABLE, more people run FreeBSD-STABLE than FreeBSD-CURRENT, so it is inevitable that bugs and corner cases will sometimes be found in FreeBSD-STABLE that were not apparent in FreeBSD-CURRENT.

For these reasons, we do *not* recommend that you blindly track FreeBSD-STABLE, and it is particularly important that you do not update any production servers to FreeBSD-STABLE without first thoroughly testing the code in your development environment.

If you do not have the resources to do this then we recommend that you run the most recent release of FreeBSD, and use the binary update mechanism to move from release to release.

24.5.2.3 Using FreeBSD-STABLE

1. Join the freebsd-stable (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable) list. This will keep you informed of build-dependencies that may appear in FreeBSD-STABLE or any other issues requiring special attention. Developers will also make announcements in this mailing list when they are contemplating some controversial fix or update, giving the users a chance to respond if they have any issues to raise concerning the proposed change.

   Join the relevant **SVN** list for the branch you are tracking. For example, if you are tracking the 7-STABLE branch, join the svn-src-stable-7 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-7) list. This will allow you to view the commit log entry for each change as it is made, along with any pertinent information on possible side-effects.

To join these lists, or one of the others available go to http://lists.FreeBSD.org/mailman/listinfo and click on the list that you wish to subscribe to. Instructions on the rest of the procedure are available there. If you are interested in tracking changes for the whole source tree, we would recommend subscribing to the svn-src-all (http://lists.FreeBSD.org/mailman/listinfo/svn-src-all) list.

2. If you are going to install a new system and want it to run monthly snapshot built from FreeBSD-STABLE, please check the Snapshots (http://www.FreeBSD.org/snapshots/) web page for more information. Alternatively, it is possible to install the most recent FreeBSD-STABLE release from the mirror sites and follow the instructions below to upgrade your system to the most up to date FreeBSD-STABLE source code.

   If you are already running a previous release of FreeBSD and wish to upgrade via sources then you can easily do so from FreeBSD mirror site. This can be done in one of two ways:

   a. Use the cvsup program with the `supfile` named `stable-supfile` from the directory `/usr/share/examples/cvsup`. This is the most recommended method, since it allows you to grab the entire collection once and then only what has changed from then on. Many people run `cvsup` from `cron` to keep their sources up-to-date automatically. You have to customize the sample `supfile` above, and configure cvsup for your environment.

   b. Use the **CTM** facility. If you do not have a fast and inexpensive connection to the Internet, this is the method you should consider using.

3. Essentially, if you need rapid on-demand access to the source and communications bandwidth is not a consideration, use `cvsup` or `ftp`. Otherwise, use **CTM**.

4. Before compiling FreeBSD-STABLE, read the `Makefile` in `/usr/src` carefully. You should at least install a new kernel and rebuild the world the first time through as part of the upgrading process. Reading the FreeBSD-STABLE mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable) and `/usr/src/UPDATING` will keep you up-to-date on other bootstrapping procedures that sometimes become necessary as we move toward the next release.

24.6 Synchronizing Your Source

There are various ways of using an Internet (or email) connection to stay up-to-date with any given area of the FreeBSD project sources, or all areas, depending on what interests you. The primary services we offer are Anonymous CVS, CVSup, and CTM.

> **Warning:** While it is possible to update only parts of your source tree, the only supported update procedure is to update the entire tree and recompile both userland (i.e., all the programs that run in user space, such as those in `/bin` and `/sbin`) and kernel sources. Updating only part of your source tree, only the kernel, or only userland will often result in problems. These problems may range from compile errors to kernel panics or data corruption.

Anonymous CVS and **CVSup** use the *pull* model of updating sources. In the case of **CVSup** the user (or a `cron` script) invokes the `cvsup` program, and it interacts with a `cvsupd` server somewhere to bring your files up-to-date. The updates you receive are up-to-the-minute and you get them when, and only when, you want them. You can easily restrict your updates to the specific files or directories that are of interest to you. Updates are generated on the fly by the server, according to what you have and what you want to have. **Anonymous CVS** is quite a bit more simplistic

than **CVSup** in that it is just an extension to **CVS** which allows it to pull changes directly from a remote CVS repository. **CVSup** can do this far more efficiently, but **Anonymous CVS** is easier to use.

CTM, on the other hand, does not interactively compare the sources you have with those on the master archive or otherwise pull them across. Instead, a script which identifies changes in files since its previous run is executed several times a day on the master CTM machine, any detected changes being compressed, stamped with a sequence-number and encoded for transmission over email (in printable ASCII only). Once received, these "CTM deltas" can then be handed to the ctm_rmail(1) utility which will automatically decode, verify and apply the changes to the user's copy of the sources. This process is far more efficient than **CVSup**, and places less strain on our server resources since it is a *push* rather than a *pull* model.

There are other trade-offs, of course. If you inadvertently wipe out portions of your archive, **CVSup** will detect and rebuild the damaged portions for you. **CTM** will not do this, and if you wipe some portion of your source tree out (and do not have it backed up) then you will have to start from scratch (from the most recent CVS "base delta") and rebuild it all with **CTM** or, with **Anonymous CVS**, simply delete the bad bits and resync.

24.7 Rebuilding "world"

Once you have synchronized your local source tree against a particular version of FreeBSD (FreeBSD-STABLE, FreeBSD-CURRENT, and so on) you can then use the source tree to rebuild the system.

Make a Backup: It cannot be stressed enough how important it is to make a backup of your system *before* you do this. While rebuilding the world is (as long as you follow these instructions) an easy task to do, there will inevitably be times when you make mistakes, or when mistakes made by others in the source tree render your system unbootable.

Make sure you have taken a backup. And have a fixit floppy or bootable CD at hand. You will probably never have to use it, but it is better to be safe than sorry!

Subscribe to the Right Mailing List: The FreeBSD-STABLE and FreeBSD-CURRENT branches are, by their nature, *in development*. People that contribute to FreeBSD are human, and mistakes occasionally happen.

Sometimes these mistakes can be quite harmless, just causing your system to print a new diagnostic warning. Or the change may be catastrophic, and render your system unbootable or destroy your file systems (or worse).

If problems like these occur, a "heads up" is posted to the appropriate mailing list, explaining the nature of the problem and which systems it affects. And an "all clear" announcement is posted when the problem has been solved.

If you try to track FreeBSD-STABLE or FreeBSD-CURRENT and do not read the FreeBSD-STABLE mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable) or the FreeBSD-CURRENT mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current) respectively, then you are asking for trouble.

Do not use `make world`: A lot of older documentation recommends using `make world` for this. Doing that skips some important steps and should only be used if you are sure of what you are doing. For almost all circumstances `make world` is the wrong thing to do, and the procedure described here should be used instead.

24.7.1 The Canonical Way to Update Your System

To update your system, you should check `/usr/src/UPDATING` for any pre-buildworld steps necessary for your version of the sources and then use the procedure outlined here.

These upgrade steps assume that you are currently using an old FreeBSD version, consisting of an old compiler, old kernel, old world and old configuration files. By “world” here we mean the core system binaries, libraries and programming files. The compiler is part of “world”, but has a few special concerns.

We also assume that you have already obtained the sources to a newer system. If the sources available on the particular system are old too, see Section 24.6 for detailed help about synchronizing them to a newer version.

Updating the system from sources is a bit more subtle than it might initially seem to be, and the FreeBSD developers have found it necessary over the years to change the recommended approach fairly dramatically as new kinds of unavoidable dependencies come to light. The rest of this section describes the rationale behind the currently recommended upgrade sequence.

Any successful update sequence must deal with the following issues:

- The old compiler might not be able to compile the new kernel. (Old compilers sometimes have bugs.) So, the new kernel should be built with the new compiler. In particular, the new compiler must be built before the new kernel is built. This does not necessarily mean that the new compiler must be *installed* before building the new kernel.
- The new world might rely on new kernel features. So, the new kernel must be installed before the new world is installed.

These first two issues are the basis for the core `buildworld`, `buildkernel`, `installkernel`, `installworld` sequence that we describe in the following paragraphs. This is not an exhaustive list of all the reasons why you should prefer the currently recommended upgrade process. Some of the less obvious ones are listed below:

- The old world might not run correctly on the new kernel, so you must install the new world immediately upon installing the new kernel.
- Some configuration changes must be done before the new world is installed, but others might break the old world. Hence, two different configuration upgrade steps are generally needed.
- For the most part, the update process only replaces or adds files; existing old files are not deleted. In a few cases, this can cause problems. As a result, the update procedure will sometimes specify certain files that should be manually deleted at certain steps. This may or may not be automated in the future.

These concerns have led to the following recommended sequence. Note that the detailed sequence for particular updates may require additional steps, but this core process should remain unchanged for some time:

1. `make buildworld`

   This first compiles the new compiler and a few related tools, then uses the new compiler to compile the rest of the new world. The result ends up in `/usr/obj`.

2. `make buildkernel`

   Unlike the older approach, using config(8) and make(1), this uses the *new* compiler residing in `/usr/obj`. This protects you against compiler-kernel mismatches.

3. `make installkernel`

   Place the new kernel and kernel modules onto the disk, making it possible to boot with the newly updated kernel.

4. Reboot into single user mode.

   Single user mode minimizes problems from updating software that's already running. It also minimizes any problems from running the old world on a new kernel.

5. `mergemaster -p`

   This does some initial configuration file updates in preparation for the new world. For instance it may add new user groups to the system, or new user names to the password database. This is often necessary when new groups or special system-user accounts have been added since the last update, so that the `installworld` step will be able to use the newly installed system user or system group names without problems.

6. `make installworld`

   Copies the world from `/usr/obj`. You now have a new kernel and new world on disk.

7. `mergemaster`

   Now you can update the remaining configuration files, since you have a new world on disk.

8. Reboot.

   A full machine reboot is needed now to load the new kernel and new world with new configuration files.

Note that if you're upgrading from one release of the same FreeBSD branch to a more recent release of the same branch, i.e. from 7.0 to 7.1, then this procedure may not be absolutely necessary, since you're unlikely to run into serious mismatches between compiler, kernel, userland and configuration files. The older approach of `make world` followed by building and installing a new kernel might work well enough for minor updates.

But, when upgrading across major releases, people who don't follow this procedure should expect some problems.

It is also worth noting that many upgrades (i.e. 4.*X* to 5.0) may require specific additional steps (renaming or deleting specific files prior to installworld, for instance). Read the `/usr/src/UPDATING` file carefully, especially at the end, where the currently recommended upgrade sequence is explicitly spelled out.

This procedure has evolved over time as the developers have found it impossible to completely prevent certain kinds of mismatch problems. Hopefully, the current procedure will remain stable for a long time.

To summarize, the currently recommended way of upgrading FreeBSD from sources is:

```
# cd /usr/src
# make buildworld
# make buildkernel
# make installkernel
# shutdown -r now
```

> **Note:** There are a few rare cases when an extra run of `mergemaster -p` is needed before the `buildworld` step. These are described in `UPDATING`. In general, though, you can safely omit this step if you are not updating across one or more major FreeBSD versions.

After `installkernel` finishes successfully, you should boot in single user mode (i.e. using `boot -s` from the loader prompt). Then run:

```
# adjkerntz -i
# mount -a -t ufs
# mergemaster -p
```

```
# cd /usr/src
# make installworld
# mergemaster
# reboot
```

> **Read Further Explanations:** The sequence described above is only a short resume to help you getting started. You should however read the following sections to clearly understand each step, especially if you want to use a custom kernel configuration.

24.7.2 Read `/usr/src/UPDATING`

Before you do anything else, read `/usr/src/UPDATING` (or the equivalent file wherever you have a copy of the source code). This file should contain important information about problems you might encounter, or specify the order in which you might have to run certain commands. If `UPDATING` contradicts something you read here, `UPDATING` takes precedence.

> **Important:** Reading `UPDATING` is not an acceptable substitute for subscribing to the correct mailing list, as described previously. The two requirements are complementary, not exclusive.

24.7.3 Check `/etc/make.conf`

Examine the files `/usr/share/examples/etc/make.conf` and `/etc/make.conf`. The first contains some default defines – most of which are commented out. To make use of them when you rebuild your system from source, add them to `/etc/make.conf`. Keep in mind that anything you add to `/etc/make.conf` is also used every time you run `make`, so it is a good idea to set them to something sensible for your system.

A typical user will probably want to copy the `CFLAGS` and `NO_PROFILE` lines found in `/usr/share/examples/etc/make.conf` to `/etc/make.conf` and uncomment them.

Examine the other definitions (`COPTFLAGS`, `NOPORTDOCS` and so on) and decide if they are relevant to you.

24.7.4 Update the Files in `/etc`

The `/etc` directory contains a large part of your system's configuration information, as well as scripts that are run at system startup. Some of these scripts change from version to version of FreeBSD.

Some of the configuration files are also used in the day to day running of the system. In particular, `/etc/group`.

There have been occasions when the installation part of `make installworld` has expected certain usernames or groups to exist. When performing an upgrade it is likely that these users or groups did not exist. This caused problems when upgrading. In some cases `make buildworld` will check to see if these users or groups exist.

An example of this is when the `smmsp` user was added. Users had the installation process fail for them when mtree(8) was trying to create `/var/spool/clientmqueue`.

The solution is to run mergemaster(8) in pre-buildworld mode by providing the `-p` option. This will compare only those files that are essential for the success of `buildworld` or `installworld`. If your old version of `mergemaster` does not support `-p`, use the new version in the source tree when running for the first time:

```
# cd /usr/src/usr.sbin/mergemaster
# ./mergemaster.sh -p
```

> **Tip:** If you are feeling particularly paranoid, you can check your system to see which files are owned by the group you are renaming or deleting:
>
> ```
> # find / -group GID -print
> ```
>
> will show all files owned by group *GID* (which can be either a group name or a numeric group ID).

24.7.5 Drop to Single User Mode

You may want to compile the system in single user mode. Apart from the obvious benefit of making things go slightly faster, reinstalling the system will touch a lot of important system files, all the standard system binaries, libraries, include files and so on. Changing these on a running system (particularly if you have active users on the system at the time) is asking for trouble.

Another method is to compile the system in multi-user mode, and then drop into single user mode for the installation. If you would like to do it this way, simply hold off on the following steps until the build has completed. You can postpone dropping to single user mode until you have to `installkernel` or `installworld`.

As the superuser, you can execute:

```
# shutdown now
```

from a running system, which will drop it to single user mode.

Alternatively, reboot the system, and at the boot prompt, select the “single user” option. The system will then boot single user. At the shell prompt you should then run:

```
# fsck -p
# mount -u /
# mount -a -t ufs
# swapon -a
```

This checks the file systems, remounts `/` read/write, mounts all the other UFS file systems referenced in `/etc/fstab` and then turns swapping on.

> **Note:** If your CMOS clock is set to local time and not to GMT (this is true if the output of the date(1) command does not show the correct time and zone), you may also need to run the following command:
>
> ```
> # adjkerntz -i
> ```
>
> This will make sure that your local time-zone settings get set up correctly — without this, you may later run into some problems.

24.7.6 Remove `/usr/obj`

As parts of the system are rebuilt they are placed in directories which (by default) go under `/usr/obj`. The directories shadow those under `/usr/src`.

You can speed up the `make buildworld` process, and possibly save yourself some dependency headaches by removing this directory as well.

Some files below `/usr/obj` may have the immutable flag set (see chflags(1) for more information) which must be removed first.

```
# cd /usr/obj
# chflags -R noschg *
# rm -rf *
```

24.7.7 Recompile the Base System

24.7.7.1 Saving the Output

It is a good idea to save the output you get from running make(1) to another file. If something goes wrong you will have a copy of the error message. While this might not help you in diagnosing what has gone wrong, it can help others if you post your problem to one of the FreeBSD mailing lists.

The easiest way to do this is to use the script(1) command, with a parameter that specifies the name of the file to save all output to. You would do this immediately before rebuilding the world, and then type **`exit`** when the process has finished.

```
# script /var/tmp/mw.out
Script started, output file is /var/tmp/mw.out
# make TARGET
... compile, compile, compile ...
# exit
Script done, ...
```

If you do this, *do not* save the output in `/tmp`. This directory may be cleared next time you reboot. A better place to store it is in `/var/tmp` (as in the previous example) or in `root`'s home directory.

24.7.7.2 Compile the Base System

You must be in the `/usr/src` directory:

```
# cd /usr/src
```

(unless, of course, your source code is elsewhere, in which case change to that directory instead).

To rebuild the world you use the make(1) command. This command reads instructions from the `Makefile`, which describes how the programs that comprise FreeBSD should be rebuilt, the order in which they should be built, and so on.

The general format of the command line you will type is as follows:

```
# make -x -DVARIABLE target
```

In this example, `-x` is an option that you would pass to make(1). See the make(1) manual page for an example of the options you can pass.

`-D`*`VARIABLE`* passes a variable to the `Makefile`. The behavior of the `Makefile` is controlled by these variables. These are the same variables as are set in `/etc/make.conf`, and this provides another way of setting them.

```
# make -DNO_PROFILE target
```

is another way of specifying that profiled libraries should not be built, and corresponds with the

```
NO_PROFILE=    true     #    Avoid compiling profiled libraries
```

line in `/etc/make.conf`.

`target` tells make(1) what you want to do. Each `Makefile` defines a number of different “targets”, and your choice of target determines what happens.

Some targets are listed in the `Makefile`, but are not meant for you to run. Instead, they are used by the build process to break out the steps necessary to rebuild the system into a number of sub-steps.

Most of the time you will not need to pass any parameters to make(1), and so your command like will look like this:

```
# make target
```

Where *`target`* will be one of many build options. The first target should always be `buildworld`.

As the names imply, `buildworld` builds a complete new tree under `/usr/obj`, and `installworld`, another target, installs this tree on the current machine.

Having separate options is very useful for two reasons. First, it allows you to do the build safe in the knowledge that no components of your running system will be affected. The build is “self hosted”. Because of this, you can safely run `buildworld` on a machine running in multi-user mode with no fear of ill-effects. It is still recommended that you run the `installworld` part in single user mode, though.

Secondly, it allows you to use NFS mounts to upgrade multiple machines on your network. If you have three machines, `A`, `B` and `C` that you want to upgrade, run `make buildworld` and `make installworld` on `A`. `B` and `C` should then NFS mount `/usr/src` and `/usr/obj` from `A`, and you can then run `make installworld` to install the results of the build on `B` and `C`.

Although the `world` target still exists, you are strongly encouraged not to use it.

Run

```
# make buildworld
```

It is possible to specify a `-j` option to `make` which will cause it to spawn several simultaneous processes. This is most useful on multi-CPU machines. However, since much of the compiling process is IO bound rather than CPU bound it is also useful on single CPU machines.

On a typical single-CPU machine you would run:

```
# make -j4 buildworld
```

make(1) will then have up to 4 processes running at any one time. Empirical evidence posted to the mailing lists shows this generally gives the best performance benefit.

If you have a multi-CPU machine and you are using an SMP configured kernel try values between 6 and 10 and see how they speed things up.

24.7.7.3 Timings

Many factors influence the build time, but fairly recent machines may only take a one or two hours to build the FreeBSD-STABLE tree, with no tricks or shortcuts used during the process. A FreeBSD-CURRENT tree will take somewhat longer.

24.7.8 Compile and Install a New Kernel

To take full advantage of your new system you should recompile the kernel. This is practically a necessity, as certain memory structures may have changed, and programs like ps(1) and top(1) will fail to work until the kernel and source code versions are the same.

The simplest, safest way to do this is to build and install a kernel based on `GENERIC`. While `GENERIC` may not have all the necessary devices for your system, it should contain everything necessary to boot your system back to single user mode. This is a good test that the new system works properly. After booting from `GENERIC` and verifying that your system works you can then build a new kernel based on your normal kernel configuration file.

On FreeBSD it is important to build world before building a new kernel.

> **Note:** If you want to build a custom kernel, and already have a configuration file, just use `KERNCONF=`*`MYKERNEL`* like this:
>
> ```
> # cd /usr/src
> # make buildkernel KERNCONF=MYKERNEL
> # make installkernel KERNCONF=MYKERNEL
> ```

Note that if you have raised `kern.securelevel` above 1 *and* you have set either the `noschg` or similar flags to your kernel binary, you might find it necessary to drop into single user mode to use `installkernel`. Otherwise you should be able to run both these commands from multi user mode without problems. See init(8) for details about `kern.securelevel` and chflags(1) for details about the various file flags.

24.7.9 Reboot into Single User Mode

You should reboot into single user mode to test the new kernel works. Do this by following the instructions in Section 24.7.5.

24.7.10 Install the New System Binaries

You should now use `installworld` to install the new system binaries.

Run

```
# cd /usr/src
```

```
# make installworld
```

> **Note:** If you specified variables on the `make buildworld` command line, you must specify the same variables in the `make installworld` command line. This does not necessarily hold true for other options; for example, `-j` must never be used with `installworld`.
>
> For example, if you ran:
>
> ```
> # make -DNO_PROFILE buildworld
> ```
>
> you must install the results with:
>
> ```
> # make -DNO_PROFILE installworld
> ```
>
> otherwise it would try to install profiled libraries that had not been built during the `make buildworld` phase.

24.7.11 Update Files Not Updated by `make installworld`

Remaking the world will not update certain directories (in particular, `/etc`, `/var` and `/usr`) with new or changed configuration files.

The simplest way to update these files is to use mergemaster(8), though it is possible to do it manually if you would prefer to do that. Regardless of which way you choose, be sure to make a backup of `/etc` in case anything goes wrong.

24.7.11.1 `mergemaster`

The mergemaster(8) utility is a Bourne script that will aid you in determining the differences between your configuration files in `/etc`, and the configuration files in the source tree `/usr/src/etc`. This is the recommended solution for keeping the system configuration files up to date with those located in the source tree.

To begin simply type `mergemaster` at your prompt, and watch it start going. `mergemaster` will then build a temporary root environment, from `/` down, and populate it with various system configuration files. Those files are then compared to the ones currently installed in your system. At this point, files that differ will be shown in diff(1) format, with the + sign representing added or modified lines, and - representing lines that will be either removed completely, or replaced with a new line. See the diff(1) manual page for more information about the diff(1) syntax and how file differences are shown.

mergemaster(8) will then show you each file that displays variances, and at this point you will have the option of either deleting the new file (referred to as the temporary file), installing the temporary file in its unmodified state, merging the temporary file with the currently installed file, or viewing the diff(1) results again.

Choosing to delete the temporary file will tell mergemaster(8) that we wish to keep our current file unchanged, and to delete the new version. This option is not recommended, unless you see no reason to change the current file. You can get help at any time by typing **?** at the mergemaster(8) prompt. If the user chooses to skip a file, it will be presented again after all other files have been dealt with.

Choosing to install the unmodified temporary file will replace the current file with the new one. For most unmodified files, this is the best option.

Choosing to merge the file will present you with a text editor, and the contents of both files. You can now merge them by reviewing both files side by side on the screen, and choosing parts from both to create a finished product. When the files are compared side by side, the **l** key will select the left contents and the **r** key will select contents from your right. The final output will be a file consisting of both parts, which can then be installed. This option is customarily used for files where settings have been modified by the user.

Choosing to view the diff(1) results again will show you the file differences just like mergemaster(8) did before prompting you for an option.

After mergemaster(8) is done with the system files you will be prompted for other options. mergemaster(8) may ask if you want to rebuild the password file and will finish up with an option to remove left-over temporary files.

24.7.11.2 Manual Update

If you wish to do the update manually, however, you cannot just copy over the files from `/usr/src/etc` to `/etc` and have it work. Some of these files must be "installed" first. This is because the `/usr/src/etc` directory *is not* a copy of what your `/etc` directory should look like. In addition, there are files that should be in `/etc` that are not in `/usr/src/etc`.

If you are using mergemaster(8) (as recommended), you can skip forward to the next section.

The simplest way to do this by hand is to install the files into a new directory, and then work through them looking for differences.

> **Backup Your Existing `/etc`:** Although, in theory, nothing is going to touch this directory automatically, it is always better to be sure. So copy your existing `/etc` directory somewhere safe. Something like:
>
> ```
> # cp -Rp /etc /etc.old
> ```
>
> `-R` does a recursive copy, `-p` preserves times, ownerships on files and suchlike.

You need to build a dummy set of directories to install the new `/etc` and other files into. `/var/tmp/root` is a reasonable choice, and there are a number of subdirectories required under this as well.

```
# mkdir /var/tmp/root
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root distrib-dirs distribution
```

This will build the necessary directory structure and install the files. A lot of the subdirectories that have been created under `/var/tmp/root` are empty and should be deleted. The simplest way to do this is to:

```
# cd /var/tmp/root
# find -d . -type d | xargs rmdir 2>/dev/null
```

This will remove all empty directories. (Standard error is redirected to `/dev/null` to prevent the warnings about the directories that are not empty.)

`/var/tmp/root` now contains all the files that should be placed in appropriate locations below `/`. You now have to go through each of these files, determining how they differ with your existing files.

Note that some of the files that will have been installed in `/var/tmp/root` have a leading ".". At the time of writing the only files like this are shell startup files in `/var/tmp/root/` and `/var/tmp/root/root/`, although there may be others (depending on when you are reading this). Make sure you use `ls -a` to catch them.

The simplest way to do this is to use diff(1) to compare the two files:

```
# diff /etc/shells /var/tmp/root/etc/shells
```

This will show you the differences between your `/etc/shells` file and the new `/var/tmp/root/etc/shells` file. Use these to decide whether to merge in changes that you have made or whether to copy over your old file.

Name the New Root Directory (`/var/tmp/root`) with a Time Stamp, so You Can Easily Compare Differences Between Versions: Frequently rebuilding the world means that you have to update `/etc` frequently as well, which can be a bit of a chore.

You can speed this process up by keeping a copy of the last set of changed files that you merged into `/etc`. The following procedure gives one idea of how to do this.

1. Make the world as normal. When you want to update `/etc` and the other directories, give the target directory a name based on the current date. If you were doing this on the 14th of February 1998 you could do the following:

   ```
   # mkdir /var/tmp/root-19980214
   # cd /usr/src/etc
   # make DESTDIR=/var/tmp/root-19980214 \
       distrib-dirs distribution
   ```

2. Merge in the changes from this directory as outlined above.

   Do not remove the `/var/tmp/root-19980214` directory when you have finished.

3. When you have downloaded the latest version of the source and remade it, follow step 1. This will give you a new directory, which might be called `/var/tmp/root-19980221` (if you wait a week between doing updates).

4. You can now see the differences that have been made in the intervening week using diff(1) to create a recursive diff between the two directories:

   ```
   # cd /var/tmp
   # diff -r root-19980214 root-19980221
   ```

   Typically, this will be a much smaller set of differences than those between `/var/tmp/root-19980221/etc` and `/etc`. Because the set of differences is smaller, it is easier to migrate those changes across into your `/etc` directory.

5. You can now remove the older of the two `/var/tmp/root-*` directories:

   ```
   # rm -rf /var/tmp/root-19980214
   ```

6. Repeat this process every time you need to merge in changes to `/etc`.

You can use date(1) to automate the generation of the directory names:

```
# mkdir /var/tmp/root-‘date "+%Y%m%d"‘
```

24.7.12 Rebooting

You are now done. After you have verified that everything appears to be in the right place you can reboot the system. A simple shutdown(8) should do it:

```
# shutdown -r now
```

24.7.13 Finished

You should now have successfully upgraded your FreeBSD system. Congratulations.

If things went slightly wrong, it is easy to rebuild a particular piece of the system. For example, if you accidentally deleted `/etc/magic` as part of the upgrade or merge of `/etc`, the file(1) command will stop working. In this case, the fix would be to run:

```
# cd /usr/src/usr.bin/file
# make all install
```

24.7.14 Questions

1. Do I need to re-make the world for every change?

There is no easy answer to this one, as it depends on the nature of the change. For example, if you just ran **CVSup**, and it has shown the following files as being updated:

```
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
```

it probably is not worth rebuilding the entire world. You could just go to the appropriate sub-directories and `make all install`, and that's about it. But if something major changed, for example `src/lib/libc/stdlib` then you should either re-make the world, or at least those parts of it that are statically linked (as well as anything else you might have added that is statically linked).

At the end of the day, it is your call. You might be happy re-making the world every fortnight say, and let changes accumulate over that fortnight. Or you might want to re-make just those things that have changed, and be confident you can spot all the dependencies.

And, of course, this all depends on how often you want to upgrade, and whether you are tracking FreeBSD-STABLE or FreeBSD-CURRENT.

2. My compile failed with lots of signal 11 (or other signal number) errors. What has happened?

This is normally indicative of hardware problems. (Re)making the world is an effective way to stress test your hardware, and will frequently throw up memory problems. These normally manifest themselves as the compiler mysteriously dying on receipt of strange signals.

A sure indicator of this is if you can restart the make and it dies at a different point in the process.

In this instance there is little you can do except start swapping around the components in your machine to determine which one is failing.

3. Can I remove `/usr/obj` when I have finished?

The short answer is yes.

`/usr/obj` contains all the object files that were produced during the compilation phase. Normally, one of the first steps in the `make buildworld` process is to remove this directory and start afresh. In this case, keeping `/usr/obj` around after you have finished makes little sense, and will free up a large chunk of disk space (currently about 2 GB).

However, if you know what you are doing you can have `make buildworld` skip this step. This will make subsequent builds run much faster, since most of sources will not need to be recompiled. The flip side of this is that subtle dependency problems can creep in, causing your build to fail in odd ways. This frequently generates noise on the FreeBSD mailing lists, when one person complains that their build has failed, not realizing that it is because they have tried to cut corners.

4. Can interrupted builds be resumed?

This depends on how far through the process you got before you found a problem.

In general (and this is not a hard and fast rule) the `make buildworld` process builds new copies of essential tools (such as gcc(1), and make(1)) and the system libraries. These tools and libraries are then installed. The new tools and libraries are then used to rebuild themselves, and are installed again. The entire system (now including regular user programs, such as ls(1) or grep(1)) is then rebuilt with the new system files.

If you are at the last stage, and you know it (because you have looked through the output that you were storing) then you can (fairly safely) do:

```
... fix the problem ...
# cd /usr/src
# make -DNO_CLEAN all
```

This will not undo the work of the previous `make buildworld`.

If you see the message:

```
--------------------------------------------------------------
Building everything..
--------------------------------------------------------------
```

in the `make buildworld` output then it is probably fairly safe to do so.

If you do not see that message, or you are not sure, then it is always better to be safe than sorry, and restart the build from scratch.

5. How can I speed up making the world?

- Run in single user mode.

- Put the `/usr/src` and `/usr/obj` directories on separate file systems held on separate disks. If possible, put these disks on separate disk controllers.
- Better still, put these file systems across multiple disks using the ccd(4) (concatenated disk driver) device.
- Turn off profiling (set "NO_PROFILE=true" in `/etc/make.conf`). You almost certainly do not need it.
- Also in `/etc/make.conf`, set `CFLAGS` to something like `-O -pipe`. The optimization `-O2` is much slower, and the optimization difference between `-O` and `-O2` is normally negligible. `-pipe` lets the compiler use pipes rather than temporary files for communication, which saves disk access (at the expense of memory).
- Pass the `-j`*n* option to make(1) to run multiple processes in parallel. This usually helps regardless of whether you have a single or a multi processor machine.
- The file system holding `/usr/src` can be mounted (or remounted) with the `noatime` option. This prevents the file system from recording the file access time. You probably do not need this information anyway.

```
# mount -u -o noatime /usr/src
```

> **Warning:** The example assumes `/usr/src` is on its own file system. If it is not (if it is a part of `/usr` for example) then you will need to use that file system mount point, and not `/usr/src`.

- The file system holding `/usr/obj` can be mounted (or remounted) with the `async` option. This causes disk writes to happen asynchronously. In other words, the write completes immediately, and the data is written to the disk a few seconds later. This allows writes to be clustered together, and can be a dramatic performance boost.

> **Warning:** Keep in mind that this option makes your file system more fragile. With this option there is an increased chance that, should power fail, the file system will be in an unrecoverable state when the machine restarts.
>
> If `/usr/obj` is the only thing on this file system then it is not a problem. If you have other, valuable data on the same file system then ensure your backups are fresh before you enable this option.

```
# mount -u -o async /usr/obj
```

> **Warning:** As above, if `/usr/obj` is not on its own file system, replace it in the example with the name of the appropriate mount point.

6. What do I do if something goes wrong?

Make absolutely sure your environment has no extraneous cruft from earlier builds. This is simple enough.

```
# chflags -R noschg /usr/obj/usr
# rm -rf /usr/obj/usr
# cd /usr/src
# make cleandir
# make cleandir
```

Yes, `make cleandir` really should be run twice.

Then restart the whole process, starting with `make buildworld`.

If you still have problems, send the error and the output of `uname -a` to FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions). Be prepared to answer other questions about your setup!

24.8 Deleting obsolete files, directories and libraries

As a part of the FreeBSD development lifecycle, it happens from time to time that files and their contents become obsolete. This may be because their functionality is implemented elsewhere, the version number of the library has changed or it was removed from the system entirely. This includes old files, libraries and directories, which should be removed when updating the system. The benefit for the user is that the system is not cluttered with old files which take up unnecessary space on the storage (and backup) medium. Additionally, if the old library had a security or stability issue, you should update to the newer library to keep your system safe and prevent crashes caused by the old library implementation. The files, directories, and libraries that are considered obsolete are listed in `/usr/src/ObsoleteFiles.inc`. The following instructions will help you removing these obsolete files during the system upgrade process.

We assume you are following the steps outlined in Section 24.7.1. After the `make installworld` and the subsequent `mergemaster` commands have finished successfully, you should check for obsolete files and libraries as follows:

```
# cd /usr/src
# make check-old
```

If any obsolete files are found, they can be deleted using the following commands:

```
# make delete-old
```

> **Tip:** See `/usr/src/Makefile` for more targets of interest.

A prompt is displayed before deleting each obsolete file. You can skip the prompt and let the system remove these files automatically by using the `BATCH_DELETE_OLD_FILES` make-variable as follows:

```
# make -DBATCH_DELETE_OLD_FILES delete-old
```

You can also achieve the same goal by piping these commands through `yes` like this:

```
# yes|make delete-old
```

> **Warning:** Deleting obsolete files will break applications that still depend on those obsolete files. This is especially true for old libraries. In most cases, you need to recompile the programs, ports, or libraries that used the old library before `make delete-old-libs` is executed.

Utilities for checking shared library dependencies are available from the Ports Collection in `sysutils/libchk` or `sysutils/bsdadminscripts`.

Obsolete shared libraries can conflict with newer libraries, causing messages like these:

```
/usr/bin/ld: warning: libz.so.4, needed by /usr/local/lib/libtiff.so, may conflict with libz.so.5
/usr/bin/ld: warning: librpcsvc.so.4, needed by /usr/local/lib/libXext.so, may conflict with libr
```

To solve these problems, determine which port installed the library:

```
# pkg_info -W  /usr/local/lib/libtiff.so
/usr/local/lib/libtiff.so was installed by package tiff-3.9.4
# pkg_info -W /usr/local/lib/libXext.so
/usr/local/lib/libXext.so was installed by package libXext-1.1.1,1
```

Then deinstall, rebuild and reinstall the port. The `ports-mgmt/portmaster` and `ports-mgmt/portupgrade` utilities can be used to automate this process. After you've made sure that all ports are rebuilt and do not use the old libraries anymore, you can delete them using the following command:

```
# make delete-old-libs
```

24.9 Tracking for Multiple Machines

If you have multiple machines that you want to track the same source tree, then having all of them download sources and rebuild everything seems like a waste of resources: disk space, network bandwidth, and CPU cycles. It is, and the solution is to have one machine do most of the work, while the rest of the machines mount that work via NFS. This section outlines a method of doing so.

24.9.1 Preliminaries

First, identify a set of machines that is going to run the same set of binaries, which we will call a *build set*. Each machine can have a custom kernel, but they will be running the same userland binaries. From that set, choose a machine to be the *build machine*. It is going to be the machine that the world and kernel are built on. Ideally, it should be a fast machine that has sufficient spare CPU to run `make buildworld` and `make buildkernel`. You will also want to choose a machine to be the *test machine*, which will test software updates before they are put into production. This *must* be a machine that you can afford to have down for an extended period of time. It can be the build machine, but need not be.

All the machines in this build set need to mount `/usr/obj` and `/usr/src` from the same machine, and at the same point. Ideally, those are on two different drives on the build machine, but they can be NFS mounted on that machine as well. If you have multiple build sets, `/usr/src` should be on one build machine, and NFS mounted on the rest.

Finally make sure that `/etc/make.conf` and `/etc/src.conf` on all the machines in the build set agrees with the build machine. That means that the build machine must build all the parts of the base system that any machine in the build set is going to install. Also, each build machine should have its kernel name set with `KERNCONF` in `/etc/make.conf`, and the build machine should list them all in `KERNCONF`, listing its own kernel first. The build machine must have the kernel configuration files for each machine in `/usr/src/sys/`*`arch`*`/conf` if it is going to build their kernels.

24.9.2 The Base System

Now that all that is done, you are ready to build everything. Build the kernel and world as described in Section 24.7.7.2 on the build machine, but do not install anything. After the build has finished, go to the test machine, and install the kernel you just built. If this machine mounts `/usr/src` and `/usr/obj` via NFS, when you reboot to single user you will need to enable the network and mount them. The easiest way to do this is to boot to multi-user, then run `shutdown now` to go to single user mode. Once there, you can install the new kernel and world and run `mergemaster` just as you normally would. When done, reboot to return to normal multi-user operations for this machine.

After you are certain that everything on the test machine is working properly, use the same procedure to install the new software on each of the other machines in the build set.

24.9.3 Ports

The same ideas can be used for the ports tree. The first critical step is mounting `/usr/ports` from the same machine to all the machines in the build set. You can then set up `/etc/make.conf` properly to share distfiles. You should set `DISTDIR` to a common shared directory that is writable by whichever user `root` is mapped to by your NFS mounts. Each machine should set `WRKDIRPREFIX` to a local build directory. Finally, if you are going to be building and distributing packages, you should set `PACKAGES` to a directory similar to `DISTDIR`.

Notes

1. That is not quite true. We can not continue to support old releases of FreeBSD forever, although we do support them for many years. For a complete description of the current security policy for old releases of FreeBSD, please see http://www.FreeBSD.org/security/.

Chapter 25
DTrace

25.1 Synopsis

DTrace, also known as Dynamic Tracing, was developed by Sun as a tool for locating performance bottlenecks in production and pre-production systems. It is not, in any way, a debugging tool, but a tool for real time system analysis to locate performance and other issues.

DTrace is a remarkable profiling tool, with an impressive array of features for diagnosing system issues. It may also be used to run pre-written scripts to take advantage of its capabilities. Users may even author their own utilities using the DTrace D Language, allowing them to customize their profiling based on specific needs.

After reading this chapter, you will know:

- What DTrace is and what features it provides.
- Differences between the Solaris DTrace implementation and the one provided by FreeBSD.
- How to enable and use DTrace on FreeBSD.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Chapter 3).
- Be familiar with the basics of kernel configuration/compilation (Chapter 8).
- Have some familiarity with security and how it pertains to FreeBSD (Chapter 14).
- Understand how to obtain and rebuild the FreeBSD sources (Chapter 24).

> **Warning:** This feature is considered experimental. Some options may be lacking in functionality, other parts may not work at all. In time, this feature will be considered production ready and this documentation will be altered to fit that situation.

25.2 Implementation Differences

While the DTrace in FreeBSD is very similar to that found in Solaris, differences exist that should be explained before continuing. The primary difference users will notice is that on FreeBSD, DTrace needs to be specifically enabled. There are kernel options and modules which must be enabled for DTrace to work properly. These will be explained later.

There is a `DDB_CTF` kernel option which is used to enable support for loading the CTF data from kernel modules and the kernel itself. CTF is the Solaris Compact C Type Format which encapsulates a reduced form of debugging

information similar to DWARF and the venerable stabs. This CTF data is added to the binaries by the `ctfconvert` and `ctfmerge` build tools. The `ctfconvert` utility parses DWARF ELF debug sections created by the compiler and `ctfmerge` merges CTF ELF sections from objects into either executables or shared libraries. More on how to enable this for the kernel and FreeBSD build is forthcoming.

Some different providers exist for FreeBSD than for Solaris. Most notable is the `dtmalloc` provider, which allows tracing `malloc()` by type in the FreeBSD kernel.

Only `root` may use DTrace on FreeBSD. This is related to security differences, Solaris has a few low level security checks which do not yet exist in FreeBSD. As such, the `/dev/dtrace/dtrace` is strictly limited to `root` users only.

Finally, the DTrace software falls under Sun's CDDL license. The `Common Development and Distribution License` comes with FreeBSD, see the `/usr/src/cddl/contrib/opensolaris/OPENSOLARIS.LICENSE` or view it online at http://www.opensolaris.org/os/licensing.

This license means that a FreeBSD kernel with the DTrace options is still BSD licensed; however the CDDL kicks in when the modules are distributed in binary form, or the binaries are loaded.

25.3 Enabling DTrace Support

To enable support for DTrace, add the following lines to the kernel configuration file:

```
options         KDTRACE_HOOKS
options         DDB_CTF
```

> **Note:** Users of the AMD64 architecture will want to add the following line to their kernel configuration file:
>
> ```
> options         KDTRACE_FRAME
> ```
>
> This option provides support for the FBT feature. DTrace will work without this option; however, there will be limited support for function boundary tracing.

All sources must be rebuilt and installed with CTF options. To accomplish this task, rebuild the FreeBSD sources using:

```
# cd /usr/src
# make WITH_CTF=1 kernel
```

The system will need to be restarted.

After rebooting and allowing the new kernel to be loaded into memory, support for the Korn shell should be added. This is needed as the DTrace toolkit has several utilities written in `ksh`. Install the `shells/ksh93`. It is also possible to run these tools under `shells/pdksh` or `shells/mksh`.

Finally, obtain the current DTrace toolkit. The current version is available at http://www.opensolaris.org/os/community/dtrace/dtracetoolkit/. There is an install mechanism included; however, installation is not required to make use of the bundled utilities.

25.4 Using DTrace

Before making use of DTrace functionality, the DTrace device must exist. To load the device, issue the following command:

```
# kldload dtraceall
```

DTrace support should now be available. To view all probes the administrator may now execute the following command:

```
# dtrace -l | more
```

All output is passed to the `more` utility as it will quickly overflow the screen buffer. At this point, DTrace should be considered working. It is now time to review the toolkit.

The toolkit is a collection of ready-made scripts to run with DTrace to collect system information. There are scripts to check open files, memory, CPU usage and a lot more. Extract the scripts with the following command:

```
# gunzip -c DTraceToolkit* | tar xvf -
```

Change into that directory with the `cd` and change the execution permissions on all files, designated as those files with lower case names, to `755`.

All of these scripts will need modifications to their contents. The ones which refer to `/usr/bin/ksh` need that changed to `/usr/local/bin/ksh`, the others which use `/usr/bin/sh` need to be altered to use `/bin/sh`, and finally the ones which use `/usr/bin/perl` will need altered to use `/usr/local/bin/perl`.

> **Important:** At this point it is prudent to remind the reader that DTrace support in FreeBSD is *incomplete* and *experimental*. Many of these scripts will not work as they are either too Solaris-specific or use probes which are unsupported at this time.

At the time of this writing only two of the scripts of the DTrace Toolkit are fully supported in FreeBSD: the `hotkernel` and `procsystime` scripts. These are the two we will explore in the following parts of this section.

The `hotkernel` is designed to identify which function is using the most kernel time. Run normally, it will produce output similar to the following:

```
# ./hotkernel
Sampling... Hit Ctrl-C to end.
```

The system administrator must use the **Ctrl**+**C** key combination to stop the process. Upon termination, the script will display a list of kernel functions and timing information, sorting the output in increasing order of time:

```
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‘_isitmyx                                           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%
```

This script will also work with kernel modules. To use this feature, run the script with the `-m` flag:

```
# ./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%
```

The `procsystime` script captures and prints the system call time usage for a given PID or process name. In the following example, a new instance of `/bin/csh` was spawned. The `procsystime` was executed and remained waiting while a few commands were typed on the other incarnation of `csh`. These are the results of this test:

```
# ./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
```

As shown, the `read()` system call seems to use the most time in nanoseconds with the `getpid()` system call used the least amount of time.

25.5 The D Language

The DTrace Toolkit includes many scripts in the special language of DTrace. This language is called “the D language” by Sun documentation, and it is very similar to C++. An in depth discussion of the language is beyond the scope of this document. It is extensively discussed at http://wikis.sun.com/display/DTrace/Documentation.

IV. Network Communication

FreeBSD is one of the most widely deployed operating systems for high performance network servers. The chapters in this part cover:

- Serial communication
- PPP and PPP over Ethernet
- Electronic Mail
- Running Network Servers
- Firewalls
- Other Advanced Networking Topics

These chapters are designed to be read when you need the information. You do not have to read them in any particular order, nor do you need to read all of them before you can begin using FreeBSD in a network environment.

Chapter 26
Serial Communications

26.1 Synopsis

UNIX has always had support for serial communications. In fact, the very first UNIX machines relied on serial lines for user input and output. Things have changed a lot from the days when the average "terminal" consisted of a 10-character-per-second serial printer and a keyboard. This chapter will cover some of the ways in which FreeBSD uses serial communications.

After reading this chapter, you will know:

- How to connect terminals to your FreeBSD system.
- How to use a modem to dial out to remote hosts.
- How to allow remote users to login to your system with a modem.
- How to boot your system from a serial console.

Before reading this chapter, you should:

- Know how to configure and install a new kernel (Chapter 8).
- Understand UNIX permissions and processes (Chapter 3).
- Have access to the technical manual for the serial hardware (modem or multi-port card) that you would like to use with FreeBSD.

26.2 Introduction

> **Warning:** As of FreeBSD 8.0, device nodes for serial ports have been renamed from `/dev/cuadN` to `/dev/cuauN` and from `/dev/ttydN` to `/dev/ttyuN`. FreeBSD 7.X users will have to adapt the following documentation according to these changes.

26.2.1 Terminology

bps

Bits per Second — the rate at which data is transmitted

DTE

Data Terminal Equipment — for example, your computer

DCE

Data Communications Equipment — your modem

RS-232

EIA standard for hardware serial communications

When talking about communications data rates, this section does not use the term “baud”. Baud refers to the number of electrical state transitions that may be made in a period of time, while “bps” (bits per second) is the *correct* term to use (at least it does not seem to bother the curmudgeons quite as much).

26.2.2 Cables and Ports

To connect a modem or terminal to your FreeBSD system, you will need a serial port on your computer and the proper cable to connect to your serial device. If you are already familiar with your hardware and the cable it requires, you can safely skip this section.

26.2.2.1 Cables

There are several different kinds of serial cables. The two most common types for our purposes are null-modem cables and standard (“straight”) RS-232 cables. The documentation for your hardware should describe the type of cable required.

26.2.2.1.1 Null-modem Cables

A null-modem cable passes some signals, such as “Signal Ground”, straight through, but switches other signals. For example, the “Transmitted Data” pin on one end goes to the “Received Data” pin on the other end.

You can also construct your own null-modem cable for use with terminals (e.g., for quality purposes). This table shows the RS-232C signals and the pin numbers on a DB-25 connector. Note that the standard also calls for a straight-through pin 1 to pin 1 *Protective Ground* line, but it is often omitted. Some terminals work OK using only pins 2, 3 and 7, while others require different configurations than the examples shown below.

Table 26-1. DB-25 to DB-25 Null-Modem Cable

Signal	Pin #		Pin #	Signal
SG	7	connects to	7	SG
TD	2	connects to	3	RD
RD	3	connects to	2	TD
RTS	4	connects to	5	CTS
CTS	5	connects to	4	RTS
DTR	20	connects to	6	DSR
DTR	20	connects to	8	DCD
DSR	6	connects to	20	DTR
DCD	8	connects to	20	DTR

Here are two other schemes more common nowadays.

Table 26-2. DB-9 to DB-9 Null-Modem Cable

Signal	Pin #		Pin #	Signal
RD	2	connects to	3	TD
TD	3	connects to	2	RD
DTR	4	connects to	6	DSR
DTR	4	connects to	1	DCD
SG	5	connects to	5	SG
DSR	6	connects to	4	DTR
DCD	1	connects to	4	DTR
RTS	7	connects to	8	CTS
CTS	8	connects to	7	RTS

Table 26-3. DB-9 to DB-25 Null-Modem Cable

Signal	Pin #		Pin #	Signal
RD	2	connects to	2	TD
TD	3	connects to	3	RD
DTR	4	connects to	6	DSR
DTR	4	connects to	8	DCD
SG	5	connects to	7	SG
DSR	6	connects to	20	DTR
DCD	1	connects to	20	DTR
RTS	7	connects to	5	CTS
CTS	8	connects to	4	RTS

> **Note:** When one pin at one end connects to a pair of pins at the other end, it is usually implemented with one short wire between the pair of pins in their connector and a long wire to the other single pin.

The above designs seems to be the most popular. In another variation (explained in the book *RS-232 Made Easy*) SG connects to SG, TD connects to RD, RTS and CTS connect to DCD, DTR connects to DSR, and vice-versa.

26.2.2.1.2 Standard RS-232C Cables

A standard serial cable passes all of the RS-232C signals straight through. That is, the “Transmitted Data” pin on one end of the cable goes to the “Transmitted Data” pin on the other end. This is the type of cable to use to connect a modem to your FreeBSD system, and is also appropriate for some terminals.

26.2.2.2 Ports

Serial ports are the devices through which data is transferred between the FreeBSD host computer and the terminal. This section describes the kinds of ports that exist and how they are addressed in FreeBSD.

26.2.2.2.1 Kinds of Ports

Several kinds of serial ports exist. Before you purchase or construct a cable, you need to make sure it will fit the ports on your terminal and on the FreeBSD system.

Most terminals will have DB-25 ports. Personal computers, including PCs running FreeBSD, will have DB-25 or DB-9 ports. If you have a multiport serial card for your PC, you may have RJ-12 or RJ-45 ports.

See the documentation that accompanied the hardware for specifications on the kind of port in use. A visual inspection of the port often works too.

26.2.2.2.2 Port Names

In FreeBSD, you access each serial port through an entry in the `/dev` directory. There are two different kinds of entries:

- Call-in ports are named `/dev/ttyu`*`N`* where *`N`* is the port number, starting from zero. Generally, you use the call-in port for terminals. Call-in ports require that the serial line assert the data carrier detect (DCD) signal to work correctly.
- Call-out ports are named `/dev/cuau`*`N`*. You usually do not use the call-out port for terminals, just for modems. You may use the call-out port if the serial cable or the terminal does not support the carrier detect signal.

If you have connected a terminal to the first serial port (`COM1` in MS-DOS), then you will use `/dev/ttyu0` to refer to the terminal. If the terminal is on the second serial port (also known as `COM2`), use `/dev/ttyu1`, and so forth.

26.2.3 Kernel Configuration

FreeBSD supports four serial ports by default. In the MS-DOS world, these are known as `COM1`, `COM2`, `COM3`, and `COM4`. FreeBSD currently supports “dumb” multiport serial interface cards, such as the BocaBoard 1008 and 2016, as well as more intelligent multi-port cards such as those made by Digiboard and Stallion Technologies. However, the default kernel only looks for the standard COM ports.

To see if your kernel recognizes any of your serial ports, watch for messages while the kernel is booting, or use the `/sbin/dmesg` command to replay the kernel’s boot messages. In particular, look for messages that start with the characters `sio`.

> **Tip:** To view just the messages that have the word `sio`, use the command:
>
> ```
> # /sbin/dmesg | grep 'sio'
> ```

For example, on a system with four serial ports, these are the serial-port specific kernel boot messages:

```
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
```

If your kernel does not recognize all of your serial ports, you will probably need to configure your kernel in the `/boot/device.hints` file. You can also comment-out or completely remove lines for devices you do not have.

Please refer to the sio(4) manual page for more information on serial ports and multiport boards configuration. Be careful if you are using a configuration file that was previously used for a different version of FreeBSD because the device flags and the syntax have changed between versions.

> **Note:** `port IO_COM1` is a substitution for `port 0x3f8`, `IO_COM2` is `0x2f8`, `IO_COM3` is `0x3e8`, and `IO_COM4` is `0x2e8`, which are fairly common port addresses for their respective serial ports; interrupts 4, 3, 5, and 9 are fairly common interrupt request lines. Also note that regular serial ports *cannot* share interrupts on ISA-bus PCs (multiport boards have on-board electronics that allow all the 16550A's on the board to share one or two interrupt request lines).

26.2.4 Device Special Files

Most devices in the kernel are accessed through "device special files", which are located in the `/dev` directory. The `sio` devices are accessed through the `/dev/ttyu`*N* (dial-in) and `/dev/cuau`*N* (call-out) devices. FreeBSD also provides initialization devices (`/dev/ttyu`*N*`.init` and `/dev/cuau`*N*`.init`) and locking devices (`/dev/ttyu`*N*`.lock` and `/dev/cuau`*N*`.lock`). The initialization devices are used to initialize communications port parameters each time a port is opened, such as `crtscts` for modems which use `RTS/CTS` signaling for flow control. The locking devices are used to lock flags on ports to prevent users or programs changing certain parameters; see the manual pages termios(4), sio(4), and stty(1) for information on the terminal settings, locking and initializing devices, and setting terminal options, respectively.

26.2.5 Serial Port Configuration

The `ttyu`*N* (or `cuau`*N*) device is the regular device you will want to open for your applications. When a process opens the device, it will have a default set of terminal I/O settings. You can see these settings with the command

```
# stty -a -f /dev/ttyu1
```

When you change the settings to this device, the settings are in effect until the device is closed. When it is reopened, it goes back to the default set. To make changes to the default set, you can open and adjust the settings of the "initial state" device. For example, to turn on `CLOCAL` mode, 8 bit communication, and `XON/XOFF` flow control by default for `ttyu5`, type:

```
# stty -f /dev/ttyu5.init clocal cs8 ixon ixoff
```

System-wide initialization of the serial devices is controlled in `/etc/rc.d/serial`. This file affects the default settings of serial devices.

To prevent certain settings from being changed by an application, make adjustments to the “lock state” device. For example, to lock the speed of `ttyu5` to 57600 bps, type:

```
# stty -f /dev/ttyu5.lock 57600
```

Now, an application that opens `ttyu5` and tries to change the speed of the port will be stuck with 57600 bps.

Naturally, you should make the initial state and lock state devices writable only by the `root` account.

26.3 Terminals

> **Warning:** As of FreeBSD 8.0, device nodes for serial ports have been renamed from `/dev/cuad`*N* to `/dev/cuau`*N* and from `/dev/ttyd`*N* to `/dev/ttyu`*N*. FreeBSD 7.X users will have to adapt the following documentation according to these changes.

Terminals provide a convenient and low-cost way to access your FreeBSD system when you are not at the computer’s console or on a connected network. This section describes how to use terminals with FreeBSD.

26.3.1 Uses and Types of Terminals

The original UNIX systems did not have consoles. Instead, people logged in and ran programs through terminals that were connected to the computer’s serial ports. It is quite similar to using a modem and terminal software to dial into a remote system to do text-only work.

Today’s PCs have consoles capable of high quality graphics, but the ability to establish a login session on a serial port still exists in nearly every UNIX style operating system today; FreeBSD is no exception. By using a terminal attached to an unused serial port, you can log in and run any text program that you would normally run on the console or in an `xterm` window in the X Window System.

For the business user, you can attach many terminals to a FreeBSD system and place them on your employees’ desktops. For a home user, a spare computer such as an older IBM PC or a Macintosh can be a terminal wired into a more powerful computer running FreeBSD. You can turn what might otherwise be a single-user computer into a powerful multiple user system.

For FreeBSD, there are three kinds of terminals:

- Dumb terminals
- PCs acting as terminals
- X terminals

The remaining subsections describe each kind.

26.3.1.1 Dumb Terminals

Dumb terminals are specialized pieces of hardware that let you connect to computers over serial lines. They are called “dumb” because they have only enough computational power to display, send, and receive text. You cannot run any programs on them. It is the computer to which you connect them that has all the power to run text editors, compilers, email, games, and so forth.

There are hundreds of kinds of dumb terminals made by many manufacturers, including Digital Equipment Corporation’s VT-100 and Wyse’s WY-75. Just about any kind will work with FreeBSD. Some high-end terminals can even display graphics, but only certain software packages can take advantage of these advanced features.

Dumb terminals are popular in work environments where workers do not need access to graphical applications such as those provided by the X Window System.

26.3.1.2 PCs Acting as Terminals

If a dumb terminal has just enough ability to display, send, and receive text, then certainly any spare personal computer can be a dumb terminal. All you need is the proper cable and some *terminal emulation* software to run on the computer.

Such a configuration is popular in homes. For example, if your spouse is busy working on your FreeBSD system’s console, you can do some text-only work at the same time from a less powerful personal computer hooked up as a terminal to the FreeBSD system.

There are at least two utilities in the base-system of FreeBSD that can be used to work through a serial connection: cu(1) and tip(1).

To connect from a client system that runs FreeBSD to the serial connection of another system, you can use:

```
# cu -l serial-port-device
```

Where “serial-port-device” is the name of a special device file denoting a serial port of your system. These device files are called `/dev/cuau`*`N`*.

The “N”-part of a device name is the serial port number.

> **Note:** Note that device numbers in FreeBSD start from zero and not one (like they do, for instance in MS-DOS-derived systems). This means that what MS-DOS-based systems call `COM1` is usually `/dev/cuau0` in FreeBSD.

> **Note:** Some people prefer to use other programs, available through the Ports Collection. The Ports include quite a few utilities which can work in ways similar to cu(1) and tip(1), i.e. `comms/minicom`.

26.3.1.3 X Terminals

X terminals are the most sophisticated kind of terminal available. Instead of connecting to a serial port, they usually connect to a network like Ethernet. Instead of being relegated to text-only applications, they can display any X application.

We introduce X terminals just for the sake of completeness. However, this chapter does *not* cover setup, configuration, or use of X terminals.

26.3.2 Configuration

This section describes what you need to configure on your FreeBSD system to enable a login session on a terminal. It assumes you have already configured your kernel to support the serial port to which the terminal is connected—and that you have connected it.

Recall from Chapter 12 that the `init` process is responsible for all process control and initialization at system startup. One of the tasks performed by `init` is to read the `/etc/ttys` file and start a `getty` process on the available terminals. The `getty` process is responsible for reading a login name and starting the `login` program.

Thus, to configure terminals for your FreeBSD system the following steps should be taken as `root`:

1. Add a line to `/etc/ttys` for the entry in the `/dev` directory for the serial port if it is not already there.
2. Specify that `/usr/libexec/getty` be run on the port, and specify the appropriate *`getty`* type from the `/etc/gettytab` file.
3. Specify the default terminal type.
4. Set the port to "on."
5. Specify whether the port should be "secure."
6. Force `init` to reread the `/etc/ttys` file.

As an optional step, you may wish to create a custom *`getty`* type for use in step 2 by making an entry in `/etc/gettytab`. This chapter does not explain how to do so; you are encouraged to see the gettytab(5) and the getty(8) manual pages for more information.

26.3.2.1 Adding an Entry to `/etc/ttys`

The `/etc/ttys` file lists all of the ports on your FreeBSD system where you want to allow logins. For example, the first virtual console `ttyv0` has an entry in this file. You can log in on the console using this entry. This file also contains entries for the other virtual consoles, serial ports, and pseudo-ttys. For a hardwired terminal, just list the serial port's `/dev` entry without the `/dev` part (for example, `/dev/ttyv0` would be listed as `ttyv0`).

A default FreeBSD install includes an `/etc/ttys` file with support for the first four serial ports: `ttyu0` through `ttyu3`. If you are attaching a terminal to one of those ports, you do not need to add another entry.

Example 26-1. Adding Terminal Entries to `/etc/ttys`

Suppose we would like to connect two terminals to the system: a Wyse-50 and an old 286 IBM PC running **Procomm** terminal software emulating a VT-100 terminal. We connect the Wyse to the second serial port and the 286 to the sixth serial port (a port on a multiport serial card). The corresponding entries in the `/etc/ttys` file would look like this:

```
ttyu1❶  "/usr/libexec/getty std.38400"❷  wy50❸  on❹  insecure❺
ttyu5   "/usr/libexec/getty std.19200"  vt100  on  insecure
```

❶ The first field normally specifies the name of the terminal special file as it is found in `/dev`.

❷ The second field is the command to execute for this line, which is usually getty(8). `getty` initializes and opens the line, sets the speed, prompts for a user name and then executes the login(1) program.

The `getty` program accepts one (optional) parameter on its command line, the *`getty`* type. A *`getty`* type configures characteristics on the terminal line, like bps rate and parity. The `getty` program reads these characteristics from the file `/etc/gettytab`.

The file `/etc/gettytab` contains lots of entries for terminal lines both old and new. In almost all cases, the entries that start with the text `std` will work for hardwired terminals. These entries ignore parity. There is a `std` entry for each bps rate from 110 to 115200. Of course, you can add your own entries to this file. The gettytab(5) manual page provides more information.

When setting the *`getty`* type in the `/etc/ttys` file, make sure that the communications settings on the terminal match.

For our example, the Wyse-50 uses no parity and connects at 38400 bps. The 286 PC uses no parity and connects at 19200 bps.

❸ The third field is the type of terminal usually connected to that tty line. For dial-up ports, `unknown` or `dialup` is typically used in this field since users may dial up with practically any type of terminal or software. For hardwired terminals, the terminal type does not change, so you can put a real terminal type from the termcap(5) database file in this field.

For our example, the Wyse-50 uses the real terminal type while the 286 PC running **Procomm** will be set to emulate at VT-100.

❹ The fourth field specifies if the port should be enabled. Putting `on` here will have the `init` process start the program in the second field, `getty`. If you put `off` in this field, there will be no `getty`, and hence no logins on the port.

❺ The final field is used to specify whether the port is secure. Marking a port as secure means that you trust it enough to allow the `root` account (or any account with a user ID of 0) to login from that port. Insecure ports do not allow `root` logins. On an insecure port, users must login from unprivileged accounts and then use su(1) or a similar mechanism to gain superuser privileges.

It is highly recommended that you use “insecure” even for terminals that are behind locked doors. It is quite easy to login and use `su` if you need superuser privileges.

26.3.2.2 Force `init` to Reread `/etc/ttys`

After making the necessary changes to the `/etc/ttys` file you should send a SIGHUP (hangup) signal to the `init` process to force it to re-read its configuration file. For example:

```
# kill -HUP 1
```

Note: `init` is always the first process run on a system, therefore it will always have PID 1.

If everything is set up correctly, all cables are in place, and the terminals are powered up, then a `getty` process should be running on each terminal and you should see login prompts on your terminals at this point.

26.3.3 Troubleshooting Your Connection

Even with the most meticulous attention to detail, something could still go wrong while setting up a terminal. Here is a list of symptoms and some suggested fixes.

26.3.3.1 No Login Prompt Appears

Make sure the terminal is plugged in and powered up. If it is a personal computer acting as a terminal, make sure it is running terminal emulation software on the correct serial port.

Make sure the cable is connected firmly to both the terminal and the FreeBSD computer. Make sure it is the right kind of cable.

Make sure the terminal and FreeBSD agree on the bps rate and parity settings. If you have a video display terminal, make sure the contrast and brightness controls are turned up. If it is a printing terminal, make sure paper and ink are in good supply.

Make sure that a `getty` process is running and serving the terminal. For example, to get a list of running `getty` processes with `ps`, type:

```
# ps -axww|grep getty
```

You should see an entry for the terminal. For example, the following display shows that a `getty` is running on the second serial port `ttyu1` and is using the `std.38400` entry in `/etc/gettytab`:

```
22189  d1  Is+    0:00.03 /usr/libexec/getty std.38400 ttyu1
```

If no `getty` process is running, make sure you have enabled the port in `/etc/ttys`. Also remember to run `kill -HUP 1` after modifying the `ttys` file.

If the `getty` process is running but the terminal still does not display a login prompt, or if it displays a prompt but will not allow you to type, your terminal or cable may not support hardware handshaking. Try changing the entry in `/etc/ttys` from `std.38400` to `3wire.38400` (remember to run `kill -HUP 1` after modifying `/etc/ttys`). The `3wire` entry is similar to `std`, but ignores hardware handshaking. You may need to reduce the baud rate or enable software flow control when using `3wire` to prevent buffer overflows.

26.3.3.2 If Garbage Appears Instead of a Login Prompt

Make sure the terminal and FreeBSD agree on the bps rate and parity settings. Check the `getty` processes to make sure the correct *`getty`* type is in use. If not, edit `/etc/ttys` and run `kill -HUP 1`.

26.3.3.3 Characters Appear Doubled; the Password Appears When Typed

Switch the terminal (or the terminal emulation software) from “half duplex” or “local echo” to “full duplex.”

26.4 Dial-in Service

Warning: As of FreeBSD 8.0, device nodes for serial ports have been renamed from `/dev/cuadN` to `/dev/cuauN` and from `/dev/ttydN` to `/dev/ttyuN`. FreeBSD 7.X users will have to adapt the following documentation according to these changes.

Configuring your FreeBSD system for dial-in service is very similar to connecting terminals except that you are dealing with modems instead of terminals.

26.4.1 External vs. Internal Modems

External modems seem to be more convenient for dial-up, because external modems often can be semi-permanently configured via parameters stored in non-volatile RAM and they usually provide lighted indicators that display the state of important RS-232 signals. Blinking lights impress visitors, but lights are also very useful to see whether a modem is operating properly.

Internal modems usually lack non-volatile RAM, so their configuration may be limited only to setting DIP switches. If your internal modem has any signal indicator lights, it is probably difficult to view the lights when the system's cover is in place.

26.4.1.1 Modems and Cables

If you are using an external modem, then you will of course need the proper cable. A standard RS-232C serial cable should suffice as long as all of the normal signals are wired:

Table 26-4. Signal Names

Acronyms	Names
RD	Received Data
TD	Transmitted Data
DTR	Data Terminal Ready
DSR	Data Set Ready
DCD	Data Carrier Detect (RS-232's Received Line Signal Detector)
SG	Signal Ground
RTS	Request to Send
CTS	Clear to Send

FreeBSD needs the RTS and CTS signals for flow control at speeds above 2400 bps, the CD signal to detect when a call has been answered or the line has been hung up, and the DTR signal to reset the modem after a session is complete. Some cables are wired without all of the needed signals, so if you have problems, such as a login session not going away when the line hangs up, you may have a problem with your cable.

Like other UNIX like operating systems, FreeBSD uses the hardware signals to find out when a call has been answered or a line has been hung up and to hangup and reset the modem after a call. FreeBSD avoids sending commands to the modem or watching for status reports from the modem. If you are familiar with connecting modems to PC-based bulletin board systems, this may seem awkward.

26.4.2 Serial Interface Considerations

FreeBSD supports NS8250-, NS16450-, NS16550-, and NS16550A-based EIA RS-232C (CCITT V.24) communications interfaces. The 8250 and 16450 devices have single-character buffers. The 16550 device provides a 16-character buffer, which allows for better system performance. (Bugs in plain 16550's prevent the use of the 16-character buffer, so use 16550A's if possible). Because single-character-buffer devices require more work by the operating system than the 16-character-buffer devices, 16550A-based serial interface cards are much preferred. If the system has many active serial ports or will have a heavy load, 16550A-based cards are better for low-error-rate communications.

26.4.3 Quick Overview

As with terminals, `init` spawns a `getty` process for each configured serial port for dial-in connections. For example, if a modem is attached to `/dev/ttyu0`, the command `ps ax` might show this:

```
 4850 ??  I      0:00.09 /usr/libexec/getty V19200 ttyu0
```

When a user dials the modem's line and the modems connect, the CD (Carrier Detect) line is reported by the modem. The kernel notices that carrier has been detected and completes `getty`'s open of the port. `getty` sends a `login:` prompt at the specified initial line speed. `getty` watches to see if legitimate characters are received, and, in a typical configuration, if it finds junk (probably due to the modem's connection speed being different than `getty`'s speed), `getty` tries adjusting the line speeds until it receives reasonable characters.

After the user enters his/her login name, `getty` executes `/usr/bin/login`, which completes the login by asking for the user's password and then starting the user's shell.

26.4.4 Configuration Files

There are three system configuration files in the `/etc` directory that you will probably need to edit to allow dial-up access to your FreeBSD system. The first, `/etc/gettytab`, contains configuration information for the `/usr/libexec/getty` daemon. Second, `/etc/ttys` holds information that tells `/sbin/init` what `tty` devices should have `getty` processes running on them. Lastly, you can place port initialization commands in the `/etc/rc.d/serial` script.

There are two schools of thought regarding dial-up modems on UNIX. One group likes to configure their modems and systems so that no matter at what speed a remote user dials in, the local computer-to-modem RS-232 interface runs at a locked speed. The benefit of this configuration is that the remote user always sees a system login prompt immediately. The downside is that the system does not know what a user's true data rate is, so full-screen programs like **Emacs** will not adjust their screen-painting methods to make their response better for slower connections.

The other school configures their modems' RS-232 interface to vary its speed based on the remote user's connection speed. For example, V.32bis (14.4 Kbps) connections to the modem might make the modem run its RS-232 interface at 19.2 Kbps, while 2400 bps connections make the modem's RS-232 interface run at 2400 bps. Because `getty` does not understand any particular modem's connection speed reporting, `getty` gives a `login:` message at an initial speed and watches the characters that come back in response. If the user sees junk, it is assumed that they know they should press the **Enter** key until they see a recognizable prompt. If the data rates do not match, `getty` sees anything the user types as "junk", tries going to the next speed and gives the `login:` prompt again. This procedure can continue ad nauseam, but normally only takes a keystroke or two before the user sees a good prompt. Obviously, this login sequence does not look as clean as the former "locked-speed" method, but a user on a low-speed connection should receive better interactive response from full-screen programs.

This section will try to give balanced configuration information, but is biased towards having the modem's data rate follow the connection rate.

26.4.4.1 `/etc/gettytab`

`/etc/gettytab` is a termcap(5)-style file of configuration information for getty(8). Please see the gettytab(5) manual page for complete information on the format of the file and the list of capabilities.

26.4.4.1.1 Locked-speed Config

If you are locking your modem's data communications rate at a particular speed, you probably will not need to make any changes to `/etc/gettytab`.

26.4.4.1.2 Matching-speed Config

You will need to set up an entry in `/etc/gettytab` to give `getty` information about the speeds you wish to use for your modem. If you have a 2400 bps modem, you can probably use the existing `D2400` entry.

```
#
# Fast dialup terminals, 2400/1200/300 rotary (can start either way)
#
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:
```

If you have a higher speed modem, you will probably need to add an entry in `/etc/gettytab`; here is an entry you could use for a 14.4 Kbps modem with a top interface speed of 19.2 Kbps:

```
#
# Additions for a V.32bis Modem
#
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:
```

This will result in 8-bit, no parity connections.

The example above starts the communications rate at 19.2 Kbps (for a V.32bis connection), then cycles through 9600 bps (for V.32), 2400 bps, 1200 bps, 300 bps, and back to 19.2 Kbps. Communications rate cycling is implemented with the `nx=` ("next table") capability. Each of the lines uses a `tc=` ("table continuation") entry to pick up the rest of the "standard" settings for a particular data rate.

If you have a 28.8 Kbps modem and/or you want to take advantage of compression on a 14.4 Kbps modem, you need to use a higher communications rate than 19.2 Kbps. Here is an example of a `gettytab` entry starting a 57.6 Kbps:

```
#
# Additions for a V.32bis or V.34 Modem
# Starting at 57.6 Kbps
#
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:
```

If you have a slow CPU or a heavily loaded system and do not have 16550A-based serial ports, you may receive `sio` "silo" errors at 57.6 Kbps.

26.4.4.2 `/etc/ttys`

Configuration of the `/etc/ttys` file was covered in Example 26-1. Configuration for modems is similar but we must pass a different argument to `getty` and specify a different terminal type. The general format for both locked-speed and matching-speed configurations is:

```
ttyu0   "/usr/libexec/getty xxx"   dialup on
```

The first item in the above line is the device special file for this entry — `ttyu0` means `/dev/ttyu0` is the file that this `getty` will be watching. The second item, `"/usr/libexec/getty xxx"` (*xxx* will be replaced by the initial `gettytab` capability) is the process `init` will run on the device. The third item, `dialup`, is the default terminal type. The fourth parameter, `on`, indicates to `init` that the line is operational. There can be a fifth parameter, `secure`, but it should only be used for terminals which are physically secure (such as the system console).

The default terminal type (`dialup` in the example above) may depend on local preferences. `dialup` is the traditional default terminal type on dial-up lines so that users may customize their login scripts to notice when the terminal is `dialup` and automatically adjust their terminal type. However, the author finds it easier at his site to specify `vt102` as the default terminal type, since the users just use VT102 emulation on their remote systems.

After you have made changes to `/etc/ttys`, you may send the `init` process a HUP signal to re-read the file. You can use the command

```
# kill -HUP 1
```

to send the signal. If this is your first time setting up the system, you may want to wait until your modem(s) are properly configured and connected before signaling `init`.

26.4.4.2.1 Locked-speed Config

For a locked-speed configuration, your `ttys` entry needs to have a fixed-speed entry provided to `getty`. For a modem whose port speed is locked at 19.2 Kbps, the `ttys` entry might look like this:

```
ttyu0   "/usr/libexec/getty std.19200"   dialup on
```

If your modem is locked at a different data rate, substitute the appropriate value for `std.`*`speed`* instead of `std.19200`. Make sure that you use a valid type listed in `/etc/gettytab`.

26.4.4.2.2 Matching-speed Config

In a matching-speed configuration, your `ttys` entry needs to reference the appropriate beginning "auto-baud" (sic) entry in `/etc/gettytab`. For example, if you added the above suggested entry for a matching-speed modem that starts at 19.2 Kbps (the `gettytab` entry containing the `V19200` starting point), your `ttys` entry might look like this:

```
ttyu0   "/usr/libexec/getty V19200"   dialup on
```

26.4.4.3 `/etc/rc.d/serial`

High-speed modems, like V.32, V.32bis, and V.34 modems, need to use hardware (`RTS/CTS`) flow control. You can add `stty` commands to `/etc/rc.d/serial` to set the hardware flow control flag in the FreeBSD kernel for the modem ports.

For example to set the `termios` flag `crtscts` on serial port #1's (`COM2`) dial-in and dial-out initialization devices, the following lines could be added to `/etc/rc.d/serial`:

```
# Serial port initial configuration
stty -f /dev/ttyu1.init crtscts
stty -f /dev/cuau1.init crtscts
```

26.4.5 Modem Settings

If you have a modem whose parameters may be permanently set in non-volatile RAM, you will need to use a terminal program (such as **Telix** under MS-DOS or `tip` under FreeBSD) to set the parameters. Connect to the modem using the same communications speed as the initial speed `getty` will use and configure the modem's non-volatile RAM to match these requirements:

- CD asserted when connected
- DTR asserted for operation; dropping DTR hangs up line and resets modem
- CTS transmitted data flow control
- Disable XON/XOFF flow control
- RTS received data flow control
- Quiet mode (no result codes)

- No command echo

Please read the documentation for your modem to find out what commands and/or DIP switch settings you need to give it.

For example, to set the above parameters on a U.S. Robotics® Sportster® 14,400 external modem, one could give these commands to the modem:

```
ATZ
AT&C1&D2&H1&I0&R2&W
```

You might also want to take this opportunity to adjust other settings in the modem, such as whether it will use V.42bis and/or MNP5 compression.

The U.S. Robotics Sportster 14,400 external modem also has some DIP switches that need to be set; for other modems, perhaps you can use these settings as an example:

- Switch 1: UP — DTR Normal
- Switch 2: N/A (Verbal Result Codes/Numeric Result Codes)
- Switch 3: UP — Suppress Result Codes
- Switch 4: DOWN — No echo, offline commands
- Switch 5: UP — Auto Answer
- Switch 6: UP — Carrier Detect Normal
- Switch 7: UP — Load NVRAM Defaults
- Switch 8: N/A (Smart Mode/Dumb Mode)

Result codes should be disabled/suppressed for dial-up modems to avoid problems that can occur if `getty` mistakenly gives a `login:` prompt to a modem that is in command mode and the modem echoes the command or returns a result code. This sequence can result in a extended, silly conversation between `getty` and the modem.

26.4.5.1 Locked-speed Config

For a locked-speed configuration, you will need to configure the modem to maintain a constant modem-to-computer data rate independent of the communications rate. On a U.S. Robotics Sportster 14,400 external modem, these commands will lock the modem-to-computer data rate at the speed used to issue the commands:

```
ATZ
AT&B1&W
```

26.4.5.2 Matching-speed Config

For a variable-speed configuration, you will need to configure your modem to adjust its serial port data rate to match the incoming call rate. On a U.S. Robotics Sportster 14,400 external modem, these commands will lock the modem's error-corrected data rate to the speed used to issue the commands, but allow the serial port rate to vary for non-error-corrected connections:

```
ATZ
AT&B2&W
```

26.4.5.3 Checking the Modem's Configuration

Most high-speed modems provide commands to view the modem's current operating parameters in a somewhat human-readable fashion. On the U.S. Robotics Sportster 14,400 external modems, the command `ATI5` displays the settings that are stored in the non-volatile RAM. To see the true operating parameters of the modem (as influenced by the modem's DIP switch settings), use the commands `ATZ` and then `ATI4`.

If you have a different brand of modem, check your modem's manual to see how to double-check your modem's configuration parameters.

26.4.6 Troubleshooting

Here are a few steps you can follow to check out the dial-up modem on your system.

26.4.6.1 Checking Out the FreeBSD System

Hook up your modem to your FreeBSD system, boot the system, and, if your modem has status indication lights, watch to see whether the modem's DTR indicator lights when the `login:` prompt appears on the system's console — if it lights up, that should mean that FreeBSD has started a `getty` process on the appropriate communications port and is waiting for the modem to accept a call.

If the DTR indicator does not light, login to the FreeBSD system through the console and issue a `ps ax` to see if FreeBSD is trying to run a `getty` process on the correct port. You should see lines like these among the processes displayed:

```
114 ??  I      0:00.10 /usr/libexec/getty V19200 ttyu0
115 ??  I      0:00.10 /usr/libexec/getty V19200 ttyu1
```

If you see something different, like this:

```
114 d0  I      0:00.10 /usr/libexec/getty V19200 ttyu0
```

and the modem has not accepted a call yet, this means that `getty` has completed its open on the communications port. This could indicate a problem with the cabling or a mis-configured modem, because `getty` should not be able to open the communications port until CD (carrier detect) has been asserted by the modem.

If you do not see any `getty` processes waiting to open the desired `ttyuN` port, double-check your entries in `/etc/ttys` to see if there are any mistakes there. Also, check the log file `/var/log/messages` to see if there are any log messages from `init` or `getty` regarding any problems. If there are any messages, triple-check the configuration files `/etc/ttys` and `/etc/gettytab`, as well as the appropriate device special files `/dev/ttyuN`, for any mistakes, missing entries, or missing device special files.

26.4.6.2 Try Dialing In

Try dialing into the system; be sure to use 8 bits, no parity, and 1 stop bit on the remote system. If you do not get a prompt right away, or get garbage, try pressing **Enter** about once per second. If you still do not see a `login:` prompt after a while, try sending a `BREAK`. If you are using a high-speed modem to do the dialing, try dialing again after locking the dialing modem's interface speed (via `AT&B1` on a U.S. Robotics Sportster modem, for example).

If you still cannot get a `login:` prompt, check `/etc/gettytab` again and double-check that

- The initial capability name specified in `/etc/ttys` for the line matches a name of a capability in `/etc/gettytab`
- Each `nx=` entry matches another `gettytab` capability name
- Each `tc=` entry matches another `gettytab` capability name

If you dial but the modem on the FreeBSD system will not answer, make sure that the modem is configured to answer the phone when DTR is asserted. If the modem seems to be configured correctly, verify that the DTR line is asserted by checking the modem's indicator lights (if it has any).

If you have gone over everything several times and it still does not work, take a break and come back to it later. If it still does not work, perhaps you can send an electronic mail message to the FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) describing your modem and your problem, and the good folks on the list will try to help.

26.5 Dial-out Service

> **Warning:** As of FreeBSD 8.0, device nodes for serial ports have been renamed from `/dev/cuadN` to `/dev/cuauN`. FreeBSD 7.X users will have to adapt the following documentation according to these changes.

The following are tips for getting your host to be able to connect over the modem to another computer. This is appropriate for establishing a terminal session with a remote host.

This is useful to log onto a BBS.

This kind of connection can be extremely helpful to get a file on the Internet if you have problems with PPP. If you need to FTP something and PPP is broken, use the terminal session to FTP it. Then use zmodem to transfer it to your machine.

26.5.1 My Stock Hayes Modem Is Not Supported, What Can I Do?

Actually, the manual page for `tip` is out of date. There is a generic Hayes dialer already built in. Just use `at=hayes` in your `/etc/remote` file.

The Hayes driver is not smart enough to recognize some of the advanced features of newer modems—messages like `BUSY`, `NO DIALTONE`, or `CONNECT 115200` will just confuse it. You should turn those messages off when you use `tip` (using `ATX0&W`).

Also, the dial timeout for `tip` is 60 seconds. Your modem should use something less, or else tip will think there is a communication problem. Try `ATS7=45&W`.

26.5.2 How Am I Expected to Enter These AT Commands?

Make what is called a "direct" entry in your `/etc/remote` file. For example, if your modem is hooked up to the first serial port, `/dev/cuau0`, then put in the following line:

```
cuau0:dv=/dev/cuau0:br#19200:pa=none
```

Use the highest bps rate your modem supports in the br capability. Then, type `tip cuau0` and you will be connected to your modem.

Or use `cu` as `root` with the following command:

```
# cu -lline -sspeed
```

`line` is the serial port (e.g.`/dev/cuau0`) and *`speed`* is the speed (e.g.`57600`). When you are done entering the AT commands type `~.` to exit.

26.5.3 The @ Sign for the pn Capability Does Not Work!

The `@` sign in the phone number capability tells tip to look in `/etc/phones` for a phone number. But the `@` sign is also a special character in capability files like `/etc/remote`. Escape it with a backslash:

```
pn=\@
```

26.5.4 How Can I Dial a Phone Number on the Command Line?

Put what is called a "generic" entry in your `/etc/remote` file. For example:

```
tip115200|Dial any phone number at 115200 bps:\
        :dv=/dev/cuau0:br#115200:at=hayes:pa=none:du:
tip57600|Dial any phone number at 57600 bps:\
        :dv=/dev/cuau0:br#57600:at=hayes:pa=none:du:
```

Then you can do things like:

```
# tip -115200 5551234
```

If you prefer `cu` over `tip`, use a generic `cu` entry:

```
cu115200|Use cu to dial any number at 115200bps:\
        :dv=/dev/cuau1:br#57600:at=hayes:pa=none:du:
```

and type:

```
# cu 5551234 -s 115200
```

26.5.5 Do I Have to Type in the bps Rate Every Time I Do That?

Put in an entry for `tip1200` or `cu1200`, but go ahead and use whatever bps rate is appropriate with the br capability. `tip` thinks a good default is 1200 bps which is why it looks for a `tip1200` entry. You do not have to use 1200 bps, though.

26.5.6 I Access a Number of Hosts Through a Terminal Server

Rather than waiting until you are connected and typing `CONNECT` *`host`* each time, use tip's `cm` capability. For example, these entries in `/etc/remote`:

```
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/cuau2:br#38400:at=hayes:du:pa=none:pn=5551234:
```

will let you type `tip pain` or `tip muffin` to connect to the hosts `pain` or `muffin`, and `tip deep13` to get to the terminal server.

26.5.7 Can Tip Try More Than One Line for Each Site?

This is often a problem where a university has several modem lines and several thousand students trying to use them.

Make an entry for your university in `/etc/remote` and use `@` for the `pn` capability:

```
big-university:\
        :pn=\@:tc=dialout
dialout:\
        :dv=/dev/cuau3:br#9600:at=courier:du:pa=none:
```

Then, list the phone numbers for the university in `/etc/phones`:

```
big-university 5551111
big-university 5551112
big-university 5551113
big-university 5551114
```

`tip` will try each one in the listed order, then give up. If you want to keep retrying, run `tip` in a while loop.

26.5.8 Why Do I Have to Hit Ctrl+P Twice to Send Ctrl+P Once?

Ctrl+**P** is the default "force" character, used to tell `tip` that the next character is literal data. You can set the force character to any other character with the `~s` escape, which means "set a variable."

Type `~sforce=`*`single-char`* followed by a newline. *`single-char`* is any single character. If you leave out *`single-char`*, then the force character is the nul character, which you can get by typing **Ctrl**+**2** or **Ctrl**+**Space**. A pretty good value for *`single-char`* is **Shift**+**Ctrl**+**6**, which is only used on some terminal servers.

You can have the force character be whatever you want by specifying the following in your `$HOME/.tiprc` file:

```
force=single-char
```

26.5.9 Suddenly Everything I Type Is in Upper Case??

You must have pressed **Ctrl**+**A**, `tip`'s "raise character," specially designed for people with broken caps-lock keys. Use `~s` as above and set the variable `raisechar` to something reasonable. In fact, you can set it to the same as the force character, if you never expect to use either of these features.

Here is a sample `.tiprc` file perfect for **Emacs** users who need to type **Ctrl**+**2** and **Ctrl**+**A** a lot:

```
force=^^
raisechar=^^
```

The ^^ is **Shift**+**Ctrl**+**6**.

26.5.10 How Can I Do File Transfers with `tip`?

If you are talking to another UNIX system, you can send and receive files with `~p` (put) and `~t` (take). These commands run `cat` and `echo` on the remote system to accept and send files. The syntax is:

`~p` local-file [remote-file]

`~t` remote-file [local-file]

There is no error checking, so you probably should use another protocol, like zmodem.

26.5.11 How Can I Run zmodem with `tip`?

To receive files, start the sending program on the remote end. Then, type `~C rz` to begin receiving them locally.

To send files, start the receiving program on the remote end. Then, type `~C sz` *`files`* to send them to the remote system.

26.6 Setting Up the Serial Console

> **Warning:** As of FreeBSD 8.0, device nodes for serial ports have been renamed from `/dev/ttydN` to `/dev/ttyuN`. FreeBSD 7.X users will have to adapt the following documentation according to these changes.

26.6.1 Introduction

FreeBSD has the ability to boot on a system with only a dumb terminal on a serial port as a console. Such a configuration should be useful for two classes of people: system administrators who wish to install FreeBSD on machines that have no keyboard or monitor attached, and developers who want to debug the kernel or device drivers.

As described in Chapter 12, FreeBSD employs a three stage bootstrap. The first two stages are in the boot block code which is stored at the beginning of the FreeBSD slice on the boot disk. The boot block will then load and run the boot loader (`/boot/loader`) as the third stage code.

In order to set up the serial console you must configure the boot block code, the boot loader code and the kernel.

26.6.2 Serial Console Configuration, Terse Version

This section assumes that you are using the default setup and just want a fast overview of setting up the serial console.

1. Connect the serial cable to `COM1` and the controlling terminal.
2. To see all boot messages on the serial console, issue the following command while logged in as the superuser:

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

3. Edit `/etc/ttys` and change `off` to `on` and `dialup` to `vt100` for the `ttyu0` entry. Otherwise a password will not be required to connect via the serial console, resulting in a potential security hole.
4. Reboot the system to see if the changes took effect.

If a different configuration is required, a more in depth configuration explanation exists in Section 26.6.3.

26.6.3 Serial Console Configuration

1. Prepare a serial cable.

   You will need either a null-modem cable or a standard serial cable and a null-modem adapter. See Section 26.2.2 for a discussion on serial cables.

2. Unplug your keyboard.

   Most PC systems probe for the keyboard during the Power-On Self-Test (POST) and will generate an error if the keyboard is not detected. Some machines complain loudly about the lack of a keyboard and will not continue to boot until it is plugged in.

   If your computer complains about the error, but boots anyway, then you do not have to do anything special. (Some machines with Phoenix BIOS installed merely say `Keyboard failed` and continue to boot normally.)

   If your computer refuses to boot without a keyboard attached then you will have to configure the BIOS so that it ignores this error (if it can). Consult your motherboard's manual for details on how to do this.

   > **Tip:** Set the keyboard to "Not installed" in the BIOS setup. You will still be able to use your keyboard. All this does is tell the BIOS not to probe for a keyboard at power-on. Your BIOS should not complain if the keyboard is absent. You can leave the keyboard plugged in even with this flag set to "Not installed" and the keyboard will still work. If the above option is not present in the BIOS, look for an "Halt on Error" option instead. Setting this to "All but Keyboard" or even to "No Errors", will have the same effect.

   > **Note:** If your system has a PS/2® mouse, chances are very good that you may have to unplug your mouse as well as your keyboard. This is because PS/2 mice share some hardware with the keyboard and leaving the mouse plugged in can fool the keyboard probe into thinking the keyboard is still there. It is said that a Gateway 2000 Pentium 90 MHz system with an AMI BIOS that behaves this way. In general, this is not a problem since the mouse is not much good without the keyboard anyway.

3. Plug a dumb terminal into `COM1` (`sio0`).

   If you do not have a dumb terminal, you can use an old PC/XT with a modem program, or the serial port on another UNIX box. If you do not have a `COM1` (`sio0`), get one. At this time, there is no way to select a port other than `COM1` for the boot blocks without recompiling the boot blocks. If you are already using `COM1` for another device, you will have to temporarily remove that device and install a new boot block and kernel once you get FreeBSD up and running. (It is assumed that `COM1` will be available on a file/compute/terminal server anyway; if

you really need `COM1` for something else (and you cannot switch that something else to `COM2` (`sio1`)), then you probably should not even be bothering with all this in the first place.)

4. Make sure the configuration file of your kernel has appropriate flags set for `COM1` (`sio0`).

   Relevant flags are:

   `0x10`

   Enables console support for this unit. The other console flags are ignored unless this is set. Currently, at most one unit can have console support; the first one (in config file order) with this flag set is preferred. This option alone will not make the serial port the console. Set the following flag or use the `-h` option described below, together with this flag.

   `0x20`

   Forces this unit to be the console (unless there is another higher priority console), regardless of the `-h` option discussed below. The flag `0x20` must be used together with the `0x10` flag.

   `0x40`

   Reserves this unit (in conjunction with `0x10`) and makes the unit unavailable for normal access. You should not set this flag to the serial port unit which you want to use as the serial console. The only use of this flag is to designate the unit for kernel remote debugging. See The Developer's Handbook (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/developers-handbook/index.html) for more information on remote debugging.

   Example:

   ```
   device sio0 flags 0x10
   ```

   See the sio(4) manual page for more details.

   If the flags were not set, you need to run UserConfig (on a different console) or recompile the kernel.

5. Create `boot.config` in the root directory of the `a` partition on the boot drive.

   This file will instruct the boot block code how you would like to boot the system. In order to activate the serial console, you need one or more of the following options—if you want multiple options, include them all on the same line:

   `-h`

   Toggles internal and serial consoles. You can use this to switch console devices. For instance, if you boot from the internal (video) console, you can use `-h` to direct the boot loader and the kernel to use the serial port as its console device. Alternatively, if you boot from the serial port, you can use the `-h` to tell the boot loader and the kernel to use the video display as the console instead.

   `-D`

   Toggles single and dual console configurations. In the single configuration the console will be either the internal console (video display) or the serial port, depending on the state of the `-h` option above. In the dual console configuration, both the video display and the serial port will become the console at the same time, regardless of the state of the `-h` option. However, note that the dual console configuration takes effect only during the boot block is running. Once the boot loader gets control, the console specified by the `-h` option becomes the only console.

`-P`

Makes the boot block probe the keyboard. If no keyboard is found, the `-D` and `-h` options are automatically set.

> **Note:** Due to space constraints in the current version of the boot blocks, the `-P` option is capable of detecting extended keyboards only. Keyboards with less than 101 keys (and without F11 and F12 keys) may not be detected. Keyboards on some laptop computers may not be properly found because of this limitation. If this is the case with your system, you have to abandon using the `-P` option. Unfortunately there is no workaround for this problem.

Use either the `-P` option to select the console automatically, or the `-h` option to activate the serial console.

You may include other options described in boot(8) as well.

The options, except for `-P`, will be passed to the boot loader (`/boot/loader`). The boot loader will determine which of the internal video or the serial port should become the console by examining the state of the `-h` option alone. This means that if you specify the `-D` option but not the `-h` option in `/boot.config`, you can use the serial port as the console only during the boot block; the boot loader will use the internal video display as the console.

6. Boot the machine.

When you start your FreeBSD box, the boot blocks will echo the contents of `/boot.config` to the console. For example:

```
/boot.config: -P
Keyboard: no
```

The second line appears only if you put `-P` in `/boot.config` and indicates presence/absence of the keyboard. These messages go to either serial or internal console, or both, depending on the option in `/boot.config`.

Options	Message goes to
none	internal console
`-h`	serial console
`-D`	serial and internal consoles
`-Dh`	serial and internal consoles
`-P`, keyboard present	internal console
`-P`, keyboard absent	serial console

After the above messages, there will be a small pause before the boot blocks continue loading the boot loader and before any further messages printed to the console. Under normal circumstances, you do not need to interrupt the boot blocks, but you may want to do so in order to make sure things are set up correctly.

Hit any key, other than **Enter**, at the console to interrupt the boot process. The boot blocks will then prompt you for further action. You should now see something like:

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

Verify the above message appears on either the serial or internal console or both, according to the options you put in `/boot.config`. If the message appears in the correct console, hit **Enter** to continue the boot process.

If you want the serial console but you do not see the prompt on the serial terminal, something is wrong with your settings. In the meantime, you enter `-h` and hit **Enter** or **Return** (if possible) to tell the boot block (and then the boot loader and the kernel) to choose the serial port for the console. Once the system is up, go back and check what went wrong.

After the boot loader is loaded and you are in the third stage of the boot process you can still switch between the internal console and the serial console by setting appropriate environment variables in the boot loader. See Section 26.6.6.

26.6.4 Summary

Here is the summary of various settings discussed in this section and the console eventually selected.

26.6.4.1 Case 1: You Set the Flags to 0x10 for `sio0`

```
device sio0 flags 0x10
```

Options in /boot.config	Console during boot blocks	Console during boot loader	Console in kernel
nothing	internal	internal	internal
`-h`	serial	serial	serial
`-D`	serial and internal	internal	internal
`-Dh`	serial and internal	serial	serial
`-P`, keyboard present	internal	internal	internal
`-P`, keyboard absent	serial and internal	serial	serial

26.6.4.2 Case 2: You Set the Flags to 0x30 for `sio0`

```
device sio0 flags 0x30
```

Options in /boot.config	Console during boot blocks	Console during boot loader	Console in kernel
nothing	internal	internal	serial
`-h`	serial	serial	serial
`-D`	serial and internal	internal	serial
`-Dh`	serial and internal	serial	serial
`-P`, keyboard present	internal	internal	serial
`-P`, keyboard absent	serial and internal	serial	serial

26.6.5 Tips for the Serial Console

26.6.5.1 Setting a Faster Serial Port Speed

By default, the serial port settings are: 9600 baud, 8 bits, no parity, and 1 stop bit. If you wish to change the default console speed, you have the following options:

- Recompile the boot blocks with `BOOT_COMCONSOLE_SPEED` set to the new console speed. See Section 26.6.5.2 for detailed instructions about building and installing new boot blocks.

  If the serial console is configured in some other way than by booting with `-h`, or if the serial console used by the kernel is different from the one used by the boot blocks, then you must also add the following option to the kernel configuration file and compile a new kernel:

  ```
  options CONSPEED=19200
  ```

- Use the `-S` boot option of the kernel. The `-S` command line option can be added to `/boot.config`. See the boot(8) manual page for a description of how to add options to `/boot.config` and a list of the supported options.
- Enable the `comconsole_speed` option in your `/boot/loader.conf` file.

  This option depends on `console`, `boot_serial`, and `boot_multicons` being set in `/boot/loader.conf` too. An example of using `comconsole_speed` to change the serial console speed is:

  ```
  boot_multicons="YES"
  boot_serial="YES"
  comconsole_speed="115200"
  console="comconsole,vidconsole"
  ```

26.6.5.2 Using Serial Port Other Than `sio0` for the Console

Using a port other than `sio0` as the console requires some recompiling. If you want to use another serial port for whatever reasons, recompile the boot blocks, the boot loader and the kernel as follows.

1. Get the kernel source. (See Chapter 24)
2. Edit `/etc/make.conf` and set `BOOT_COMCONSOLE_PORT` to the address of the port you want to use (0x3F8, 0x2F8, 0x3E8 or 0x2E8). Only `sio0` through `sio3` (`COM1` through `COM4`) can be used; multiport serial cards will not work. No interrupt setting is needed.
3. Create a custom kernel configuration file and add appropriate flags for the serial port you want to use. For example, if you want to make `sio1` (`COM2`) the console:

   ```
   device sio1 flags 0x10
   ```

   or

   ```
   device sio1 flags 0x30
   ```

   The console flags for the other serial ports should not be set.
4. Recompile and install the boot blocks and the boot loader:

   ```
   # cd /sys/boot
   # make clean
   # make
   # make install
   ```

5. Rebuild and install the kernel.
6. Write the boot blocks to the boot disk with bsdlabel(8) and boot from the new kernel.

26.6.5.3 Entering the DDB Debugger from the Serial Line

If you wish to drop into the kernel debugger from the serial console (useful for remote diagnostics, but also dangerous if you generate a spurious BREAK on the serial port!) then you should compile your kernel with the following options:

```
options BREAK_TO_DEBUGGER
options DDB
```

26.6.5.4 Getting a Login Prompt on the Serial Console

While this is not required, you may wish to get a *login* prompt over the serial line, now that you can see boot messages and can enter the kernel debugging session through the serial console. Here is how to do it.

Open the file `/etc/ttys` with an editor and locate the lines:

```
ttyu0 "/usr/libexec/getty std.9600" unknown off secure
ttyu1 "/usr/libexec/getty std.9600" unknown off secure
ttyu2 "/usr/libexec/getty std.9600" unknown off secure
ttyu3 "/usr/libexec/getty std.9600" unknown off secure
```

`ttyu0` through `ttyu3` corresponds to `COM1` through `COM4`. Change `off` to `on` for the desired port. If you have changed the speed of the serial port, you need to change `std.9600` to match the current setting, e.g. `std.19200`.

You may also want to change the terminal type from `unknown` to the actual type of your serial terminal.

After editing the file, you must `kill -HUP 1` to make this change take effect.

26.6.6 Changing Console from the Boot Loader

Previous sections described how to set up the serial console by tweaking the boot block. This section shows that you can specify the console by entering some commands and environment variables in the boot loader. As the boot loader is invoked at the third stage of the boot process, after the boot block, the settings in the boot loader will override the settings in the boot block.

26.6.6.1 Setting Up the Serial Console

You can easily specify the boot loader and the kernel to use the serial console by writing just one line in `/boot/loader.conf`:

```
set console="comconsole"
```

This will take effect regardless of the settings in the boot block discussed in the previous section.

You had better put the above line as the first line of `/boot/loader.conf` so as to see boot messages on the serial console as early as possible.

Likewise, you can specify the internal console as:

```
set console="vidconsole"
```

If you do not set the boot loader environment variable `console`, the boot loader, and subsequently the kernel, will use whichever console indicated by the `-h` option in the boot block.

The console can be specified in `/boot/loader.conf.local` or in `/boot/loader.conf`.

See loader.conf(5) for more information.

> **Note:** At the moment, the boot loader has no option equivalent to the `-P` option in the boot block, and there is no provision to automatically select the internal console and the serial console based on the presence of the keyboard.

26.6.6.2 Using a Serial Port Other Than `sio0` for the Console

You need to recompile the boot loader to use a serial port other than `sio0` for the serial console. Follow the procedure described in Section 26.6.5.2.

26.6.7 Caveats

The idea here is to allow people to set up dedicated servers that require no graphics hardware or attached keyboards. Unfortunately, while most systems will let you boot without a keyboard, there are quite a few that will not let you boot without a graphics adapter. Machines with AMI BIOSes can be configured to boot with no graphics adapter installed simply by changing the “graphics adapter” setting in the CMOS configuration to “Not installed.”

However, many machines do not support this option and will refuse to boot if you have no display hardware in the system. With these machines, you will have to leave some kind of graphics card plugged in, (even if it is just a junky mono board) although you will not have to attach a monitor. You might also try installing an AMI BIOS.

Chapter 27
PPP and SLIP

27.1 Synopsis

FreeBSD has a number of ways to link one computer to another. To establish a network or Internet connection through a dial-up modem, or to allow others to do so through you, requires the use of PPP or SLIP. This chapter describes setting up these modem-based communication services in detail.

After reading this chapter, you will know:

- How to set up user PPP.
- How to set up kernel PPP (FreeBSD 7.X only).
- How to set up PPPoE (PPP over Ethernet).
- How to set up PPPoA (PPP over ATM).
- How to configure and set up a SLIP client and server (FreeBSD 7.X only).

Before reading this chapter, you should:

- Be familiar with basic network terminology.
- Understand the basics and purpose of a dialup connection and PPP and/or SLIP.

You may be wondering what the main difference is between user PPP and kernel PPP. The answer is simple: user PPP processes the inbound and outbound data in userland rather than in the kernel. This is expensive in terms of copying the data between the kernel and userland, but allows a far more feature-rich PPP implementation. User PPP uses the `tun` device to communicate with the outside world whereas kernel PPP uses the `ppp` device.

> **Note:** Throughout in this chapter, user PPP will simply be referred to as **ppp** unless a distinction needs to be made between it and any other PPP software such as **pppd** (FreeBSD 7.X only). Unless otherwise stated, all of the commands explained in this chapter should be executed as `root`.

27.2 Using User PPP

> **Warning:** As of FreeBSD 8.0, device nodes for serial ports have been renamed from `/dev/cuadN` to `/dev/cuauN` and from `/dev/ttydN` to `/dev/ttyuN`. FreeBSD 7.X users will have to adapt the following documentation according to these changes.

27.2.1 User PPP

27.2.1.1 Assumptions

This document assumes you have the following:

- An account with an Internet Service Provider (ISP) which you connect to using PPP.
- A modem or other device connected to your system and properly configured to allow you to connect to your ISP.
- The dial-up number(s) of your ISP.
- Your login name and password. (Either a regular UNIX style login and password pair, or a PAP or CHAP login and password pair).
- The IP address of one or more name servers. Normally, you will be given two IP addresses by your ISP to use for this. If they have not given you at least one, then you can use the `enable dns` command in `ppp.conf` and **ppp** will set the name servers for you. This feature depends on your ISPs PPP implementation supporting DNS negotiation.

The following information may be supplied by your ISP, but is not completely necessary:

- The IP address of your ISP's gateway. The gateway is the machine to which you will connect and will be set up as your *default route*. If you do not have this information, we can make one up and your ISP's PPP server will tell us the correct value when we connect.

  This IP number is referred to as `HISADDR` by **ppp**.
- The netmask you should use. If your ISP has not provided you with one, you can safely use `255.255.255.255`.
- If your ISP provides you with a static IP address and hostname, you can enter it. Otherwise, we simply let the peer assign whatever IP address it sees fit.

If you do not have any of the required information, contact your ISP.

> **Note:** Throughout this section, many of the examples showing the contents of configuration files are numbered by line. These numbers serve to aid in the presentation and discussion only and are not meant to be placed in the actual file. Proper indentation with tab and space characters is also important.

27.2.1.2 Automatic PPP Configuration

Both `ppp` and `pppd` (the kernel level implementation of PPP, FreeBSD 7.X only) use the configuration files located in the `/etc/ppp` directory. Examples for user ppp can be found in `/usr/share/examples/ppp/`.

Configuring `ppp` requires that you edit a number of files, depending on your requirements. What you put in them depends to some extent on whether your ISP allocates IP addresses statically (i.e., you get given one IP address, and always use that one) or dynamically (i.e., your IP address changes each time you connect to your ISP).

27.2.1.2.1 PPP and Static IP Addresses

You will need to edit the `/etc/ppp/ppp.conf` configuration file. It should look similar to the example below.

Note: Lines that end in a : start in the first column (beginning of the line)— all other lines should be indented as shown using spaces or tabs.

```
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    provider:
12      set phone "(123) 456 7890"
13      set authname foo
14      set authkey bar
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
```

Line 1:

Identifies the default entry. Commands in this entry are executed automatically when ppp is run.

Line 2:

Enables logging parameters. When the configuration is working satisfactorily, this line should be reduced to saying:

```
set log phase tun
```

in order to avoid excessive log file sizes.

Line 3:

Tells PPP how to identify itself to the peer. PPP identifies itself to the peer if it has any trouble negotiating and setting up the link, providing information that the peers administrator may find useful when investigating such problems.

Line 4:

Identifies the device to which the modem is connected. `COM1` is `/dev/cuau0` and `COM2` is `/dev/cuau1`.

Line 5:

Sets the speed you want to connect at. If 115200 does not work (it should with any reasonably new modem), try 38400 instead.

Line 6 & 7:

The dial string. User PPP uses an expect-send syntax similar to the chat(8) program. Refer to the manual page for information on the features of this language.

Note that this command continues onto the next line for readability. Any command in `ppp.conf` may do this if the last character on the line is a `\` character.

Line 8:

Sets the idle timeout for the link. 180 seconds is the default, so this line is purely cosmetic.

Line 9:

Tells PPP to ask the peer to confirm the local resolver settings. If you run a local name server, this line should be commented out or removed.

Line 10:

A blank line for readability. Blank lines are ignored by PPP.

Line 11:

Identifies an entry for a provider called “provider”. This could be changed to the name of your ISP so that later you can use the `load` *`ISP`* to start the connection.

Line 12:

Sets the phone number for this provider. Multiple phone numbers may be specified using the colon (`:`) or pipe character (`|`) as a separator. The difference between the two separators is described in ppp(8). To summarize, if you want to rotate through the numbers, use a colon. If you want to always attempt to dial the first number first and only use the other numbers if the first number fails, use the pipe character. Always quote the entire set of phone numbers as shown.

You must enclose the phone number in quotation marks (") if there is any intention on using spaces in the phone number. This can cause a simple, yet subtle error.

Line 13 & 14:

Identifies the user name and password. When connecting using a UNIX style login prompt, these values are referred to by the `set login` command using the \U and \P variables. When connecting using PAP or CHAP, these values are used at authentication time.

Line 15:

If you are using PAP or CHAP, there will be no login at this point, and this line should be commented out or removed. See PAP and CHAP authentication for further details.

The login string is of the same chat-like syntax as the dial string. In this example, the string works for a service whose login session looks like this:

```
J. Random Provider
login: foo
password: bar
protocol: ppp
```

You will need to alter this script to suit your own needs. When you write this script for the first time, you should ensure that you have enabled “chat” logging so you can determine if the conversation is going as expected.

Line 16:

Sets the default idle timeout (in seconds) for the connection. Here, the connection will be closed automatically after 300 seconds of inactivity. If you never want to timeout, set this value to zero or use the `-ddial` command line switch.

Line 17:

Sets the interface addresses. The string *`x.x.x.x`* should be replaced by the IP address that your provider has allocated to you. The string *`y.y.y.y`* should be replaced by the IP address that your ISP indicated for their gateway (the machine to which you connect). If your ISP has not given you a gateway address, use `10.0.0.2/0`. If you need to use a "guessed" address, make sure that you create an entry in `/etc/ppp/ppp.linkup` as per the instructions for PPP and Dynamic IP addresses. If this line is omitted, `ppp` cannot run in `-auto` mode.

Line 18:

Adds a default route to your ISP's gateway. The special word `HISADDR` is replaced with the gateway address specified on line 17. It is important that this line appears after line 17, otherwise `HISADDR` will not yet be initialized.

If you do not wish to run ppp in `-auto`, this line should be moved to the `ppp.linkup` file.

It is not necessary to add an entry to `ppp.linkup` when you have a static IP address and are running ppp in `-auto` mode as your routing table entries are already correct before you connect. You may however wish to create an entry to invoke programs after connection. This is explained later with the sendmail example.

Example configuration files can be found in the `/usr/share/examples/ppp/` directory.

27.2.1.2.2 PPP and Dynamic IP Addresses

If your service provider does not assign static IP addresses, `ppp` can be configured to negotiate the local and remote addresses. This is done by "guessing" an IP address and allowing `ppp` to set it up correctly using the IP Configuration Protocol (IPCP) after connecting. The `ppp.conf` configuration is the same as PPP and Static IP Addresses, with the following change:

```
17      set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.255
```

Again, do not include the line number, it is just for reference. Indentation of at least one space is required.

Line 17:

The number after the `/` character is the number of bits of the address that ppp will insist on. You may wish to use IP numbers more appropriate to your circumstances, but the above example will always work.

The last argument (`0.0.0.0`) tells PPP to start negotiations using address `0.0.0.0` rather than `10.0.0.1` and is necessary for some ISPs. Do not use `0.0.0.0` as the first argument to `set ifaddr` as it prevents PPP from setting up an initial route in `-auto` mode.

If you are not running in `-auto` mode, you will need to create an entry in `/etc/ppp/ppp.linkup`. `ppp.linkup` is used after a connection has been established. At this point, `ppp` will have assigned the interface addresses and it will now be possible to add the routing table entries:

```
1     provider:
```

```
2       add default HISADDR
```

Line 1:

On establishing a connection, ppp will look for an entry in ppp.linkup according to the following rules: First, try to match the same label as we used in ppp.conf. If that fails, look for an entry for the IP address of our gateway. This entry is a four-octet IP style label. If we still have not found an entry, look for the MYADDR entry.

Line 2:

This line tells ppp to add a default route that points to HISADDR. HISADDR will be replaced with the IP number of the gateway as negotiated by the IPCP.

See the pmdemand entry in the files /usr/share/examples/ppp/ppp.conf.sample and /usr/share/examples/ppp/ppp.linkup.sample for a detailed example.

27.2.1.2.3 Receiving Incoming Calls

When you configure **ppp** to receive incoming calls on a machine connected to a LAN, you must decide if you wish to forward packets to the LAN. If you do, you should allocate the peer an IP number from your LAN's subnet, and use the command enable proxy in your /etc/ppp/ppp.conf file. You should also confirm that the /etc/rc.conf file contains the following:

```
gateway_enable="YES"
```

27.2.1.2.4 Which getty?

Configuring FreeBSD for Dial-up Services provides a good description on enabling dial-up services using getty(8).

An alternative to getty is mgetty (http://mgetty.greenie.net/) (from comms/mgetty+sendfax port), a smarter version of getty designed with dial-up lines in mind.

The advantages of using mgetty is that it actively *talks* to modems, meaning if port is turned off in /etc/ttys then your modem will not answer the phone.

Later versions of mgetty (from 0.99beta onwards) also support the automatic detection of PPP streams, allowing your clients script-less access to your server.

Refer to Mgetty and AutoPPP for more information on mgetty.

*27.2.1.2.5 **PPP** Permissions*

The ppp command must normally be run as the root user. If however, you wish to allow ppp to run in server mode as a normal user by executing ppp as described below, that user must be given permission to run ppp by adding them to the network group in /etc/group.

You will also need to give them access to one or more sections of the configuration file using the allow command:

```
allow users fred mary
```

If this command is used in the default section, it gives the specified users access to everything.

27.2.1.2.6 PPP Shells for Dynamic-IP Users

Create a file called `/etc/ppp/ppp-shell` containing the following:

```
#!/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
```

This script should be executable. Now make a symbolic link called `ppp-dialup` to this script using the following commands:

```
# ln -s ppp-shell /etc/ppp/ppp-dialup
```

You should use this script as the *shell* for all of your dialup users. This is an example from `/etc/passwd` for a dialup PPP user with username `pchilds` (remember do not directly edit the password file, use vipw(8)).

```
pchilds:*:1011:300:Peter Childs PPP:/home/ppp:/etc/ppp/ppp-dialup
```

Create a `/home/ppp` directory that is world readable containing the following 0 byte files:

```
-r--r--r--   1 root     wheel           0 May 27 02:23 .hushlogin
-r--r--r--   1 root     wheel           0 May 27 02:22 .rhosts
```

which prevents `/etc/motd` from being displayed.

27.2.1.2.7 PPP Shells for Static-IP Users

Create the `ppp-shell` file as above, and for each account with statically assigned IPs create a symbolic link to `ppp-shell`.

For example, if you have three dialup customers, `fred`, `sam`, and `mary`, that you route /24 CIDR networks for, you would type the following:

```
# 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
```

Each of these users dialup accounts should have their shell set to the symbolic link created above (for example, `mary`’s shell should be `/etc/ppp/ppp-mary`).

27.2.1.2.8 Setting Up ppp.conf for Dynamic-IP Users

The /etc/ppp/ppp.conf file should contain something along the lines of:

```
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
```

Note: The indenting is important.

The default: section is loaded for each session. For each dialup line enabled in /etc/ttys create an entry similar to the one for ttyu0: above. Each line should get a unique IP address from your pool of IP addresses for dynamic users.

27.2.1.2.9 Setting Up ppp.conf for Static-IP Users

Along with the contents of the sample /usr/share/examples/ppp/ppp.conf above you should add a section for each of the statically assigned dialup users. We will continue with our fred, sam, and mary example.

```
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
```

The file /etc/ppp/ppp.linkup should also contain routing information for each static IP user if required. The line below would add a route for the 203.14.101.0/24 network via the client's ppp link.

```
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 mgetty and AutoPPP

By default the comms/mgetty+sendfax port comes with the AUTO_PPP option enabled allowing mgetty to detect the LCP phase of PPP connections and automatically spawn off a ppp shell. However, since the default login/password sequence does not occur it is necessary to authenticate users using either PAP or CHAP.

This section assumes the user has successfully compiled, and installed the comms/mgetty+sendfax port on his system.

Make sure your /usr/local/etc/mgetty+sendfax/login.config file has the following in it:

```
/AutoPPP/ -     -        /etc/ppp/ppp-pap-dialup
```

This will tell mgetty to run the ppp-pap-dialup script for detected PPP connections.

Create a file called /etc/ppp/ppp-pap-dialup containing the following (the file should be executable):

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

For each dialup line enabled in /etc/ttys, create a corresponding entry in /etc/ppp/ppp.conf. This will happily co-exist with the definitions we created above.

```
pap:
  enable pap
  set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
  enable proxy
```

Each user logging in with this method will need to have a username/password in /etc/ppp/ppp.secret file, or alternatively add the following option to authenticate users via PAP from the /etc/passwd file.

```
enable passwdauth
```

If you wish to assign some users a static IP number, you can specify the number as the third argument in /etc/ppp/ppp.secret. See /usr/share/examples/ppp/ppp.secret.sample for examples.

27.2.1.2.11 MS Extensions

It is possible to configure PPP to supply DNS and NetBIOS nameserver addresses on demand.

To enable these extensions with PPP version 1.x, the following lines might be added to the relevant section of /etc/ppp/ppp.conf.

```
enable msext
set ns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

And for PPP version 2 and above:

```
accept dns
set dns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

This will tell the clients the primary and secondary name server addresses, and a NetBIOS nameserver host.

In version 2 and above, if the `set dns` line is omitted, PPP will use the values found in `/etc/resolv.conf`.

27.2.1.2.12 PAP and CHAP Authentication

Some ISPs set their system up so that the authentication part of your connection is done using either of the PAP or CHAP authentication mechanisms. If this is the case, your ISP will not give a `login:` prompt when you connect, but will start talking PPP immediately.

PAP is less secure than CHAP, but security is not normally an issue here as passwords, although being sent as plain text with PAP, are being transmitted down a serial line only. There is not much room for crackers to "eavesdrop".

Referring back to the PPP and Static IP addresses or PPP and Dynamic IP addresses sections, the following alterations must be made:

```
13      set authname MyUserName
14      set authkey MyPassword
15      set login
```

Line 13:

This line specifies your PAP/CHAP user name. You will need to insert the correct value for *`MyUserName`*.

Line 14:

This line specifies your PAP/CHAP password. You will need to insert the correct value for *`MyPassword`*. You may want to add an additional line, such as:

```
16      accept PAP
```

or

```
16      accept CHAP
```

to make it obvious that this is the intention, but PAP and CHAP are both accepted by default.

Line 15:

Your ISP will not normally require that you log into the server if you are using PAP or CHAP. You must therefore disable your "set login" string.

27.2.1.2.13 Changing Your *ppp* Configuration on the Fly

It is possible to talk to the `ppp` program while it is running in the background, but only if a suitable diagnostic port has been set up. To do this, add the following line to your configuration:

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

This will tell PPP to listen to the specified UNIX domain socket, asking clients for the specified password before allowing access. The `%d` in the name is replaced with the `tun` device number that is in use.

Once a socket has been set up, the pppctl(8) program may be used in scripts that wish to manipulate the running program.

27.2.1.3 Using PPP Network Address Translation Capability

PPP has ability to use internal NAT without kernel diverting capabilities. This functionality may be enabled by the following line in `/etc/ppp/ppp.conf`:

```
nat enable yes
```

Alternatively, PPP NAT may be enabled by command-line option `-nat`. There is also `/etc/rc.conf` knob named `ppp_nat`, which is enabled by default.

If you use this feature, you may also find useful the following `/etc/ppp/ppp.conf` options to enable incoming connections forwarding:

```
nat port tcp 10.0.0.2:ftp ftp
nat port tcp 10.0.0.2:http http
```

or do not trust the outside at all

```
nat deny_incoming yes
```

27.2.1.4 Final System Configuration

You now have `ppp` configured, but there are a few more things to do before it is ready to work. They all involve editing the `/etc/rc.conf` file.

Working from the top down in this file, make sure the `hostname=` line is set, e.g.:

```
hostname="foo.example.com"
```

If your ISP has supplied you with a static IP address and name, it is probably best that you use this name as your host name.

Look for the `network_interfaces` variable. If you want to configure your system to dial your ISP on demand, make sure the `tun0` device is added to the list, otherwise remove it.

```
network_interfaces="lo0 tun0"
ifconfig_tun0=
```

> **Note:** The `ifconfig_tun0` variable should be empty, and a file called `/etc/start_if.tun0` should be created. This file should contain the line:
>
> ```
> ppp -auto mysystem
> ```
>
> This script is executed at network configuration time, starting your ppp daemon in automatic mode. If you have a LAN for which this machine is a gateway, you may also wish to use the `-alias` switch. Refer to the manual page for further details.

Make sure that the router program is set to `NO` with the following line in your `/etc/rc.conf`:

```
router_enable="NO"
```

It is important that the `routed` daemon is not started, as `routed` tends to delete the default routing table entries created by `ppp`.

It is probably a good idea to ensure that the `sendmail_flags` line does not include the `-q` option, otherwise `sendmail` will attempt to do a network lookup every now and then, possibly causing your machine to dial out. You may try:

```
sendmail_flags="-bd"
```

The downside of this is that you must force `sendmail` to re-examine the mail queue whenever the ppp link is up by typing:

```
# /usr/sbin/sendmail -q
```

You may wish to use the `!bg` command in `ppp.linkup` to do this automatically:

```
1     provider:
2       delete ALL
3       add 0 0 HISADDR
4       !bg sendmail -bd -q30m
```

If you do not like this, it is possible to set up a “dfilter” to block SMTP traffic. Refer to the sample files for further details.

All that is left is to reboot the machine. After rebooting, you can now either type:

```
# ppp
```

and then `dial provider` to start the PPP session, or, if you want `ppp` to establish sessions automatically when there is outbound traffic (and you have not created the `start_if.tun0` script), type:

```
# ppp -auto provider
```

27.2.1.5 Summary

To recap, the following steps are necessary when setting up ppp for the first time:

Client side:

1. Ensure that the `tun` device is built into your kernel.
2. Ensure that the `tun`*N* device file is available in the `/dev` directory.
3. Create an entry in `/etc/ppp/ppp.conf`. The `pmdemand` example should suffice for most ISPs.
4. If you have a dynamic IP address, create an entry in `/etc/ppp/ppp.linkup`.
5. Update your `/etc/rc.conf` file.
6. Create a `start_if.tun0` script if you require demand dialing.

Server side:

1. Ensure that the `tun` device is built into your kernel.

2. Ensure that the `tunN` device file is available in the `/dev` directory.
3. Create an entry in `/etc/passwd` (using the vipw(8) program).
4. Create a profile in this users home directory that runs `ppp -direct direct-server` or similar.
5. Create an entry in `/etc/ppp/ppp.conf`. The `direct-server` example should suffice.
6. Create an entry in `/etc/ppp/ppp.linkup`.
7. Update your `/etc/rc.conf` file.

27.3 Using Kernel PPP

Warning: This section applies and is valid only for FreeBSD 7.X.

27.3.1 Setting Up Kernel PPP

Before you start setting up PPP on your machine, make sure that `pppd` is located in `/usr/sbin` and the directory `/etc/ppp` exists.

`pppd` can work in two modes:

1. As a “client” — you want to connect your machine to the outside world via a PPP serial connection or modem line.
2. As a “server” — your machine is located on the network, and is used to connect other computers using PPP.

In both cases you will need to set up an options file (`/etc/ppp/options` or `~/.ppprc` if you have more than one user on your machine that uses PPP).

You will also need some modem/serial software (preferably `comms/kermit`), so you can dial and establish a connection with the remote host.

27.3.2 Using `pppd` as a Client

The following `/etc/ppp/options` might be used to connect to a Cisco terminal server PPP line.

```
crtscts         # enable hardware flow control
modem           # modem control line
noipdefault     # remote PPP server must supply your IP address
                # if the remote host does not send your IP during IPCP
                # negotiation, remove this option
passive         # wait for LCP packets
domain ppp.foo.com      # put your domain name here

:remote_ip    # put the IP of remote PPP host here
                # it will be used to route packets via PPP link
                # if you didn't specified the noipdefault option
```

```
                # change this line to local_ip:remote_ip

defaultroute    # put this if you want that PPP server will be your
                # default router
```

To connect:

1. Dial to the remote host using **Kermit** (or some other modem program), and enter your user name and password (or whatever is needed to enable PPP on the remote host).
2. Exit **Kermit** (without hanging up the line).
3. Enter the following:

   ```
   # /usr/sbin/pppd /dev/tty01 19200
   ```

   Be sure to use the appropriate speed and device name.

Now your computer is connected with PPP. If the connection fails, you can add the `debug` option to the `/etc/ppp/options` file, and check console messages to track the problem.

Following `/etc/ppp/pppup` script will make all 3 stages automatic:

```
#!/bin/sh
pgrep -l pppd
pid=‘pgrep pppd‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing pppd, PID=’ ${pid}
        kill ${pid}
fi
pgrep -l kermit
pid=‘pgrep kermit‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing kermit, PID=’ ${pid}
        kill -9 ${pid}
fi

ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.dial
pppd /dev/tty01 19200
```

`/etc/ppp/kermit.dial` is a **Kermit** script that dials and makes all necessary authorization on the remote host (an example of such a script is attached to the end of this document).

Use the following `/etc/ppp/pppdown` script to disconnect the PPP line:

```
#!/bin/sh
pid=‘pgrep pppd‘
if [ X${pid} != "X" ] ; then
        echo ’killing pppd, PID=’ ${pid}
        kill -TERM ${pid}
fi

pgrep -l kermit
```

```
pid=‘pgrep kermit‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing kermit, PID=’ ${pid}
        kill -9 ${pid}
fi

/sbin/ifconfig ppp0 down
/sbin/ifconfig ppp0 delete
kermit -y /etc/ppp/kermit.hup
/etc/ppp/ppptest
```

Check to see if `pppd` is still running by executing `/usr/etc/ppp/ppptest`, which should look like this:

```
#!/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
```

To hang up the modem, execute `/etc/ppp/kermit.hup`, which should contain:

```
set line /dev/tty01 ; put your modem device here
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
```

Here is an alternate method using `chat` instead of `kermit`:

The following two files are sufficient to accomplish a `pppd` connection.

`/etc/ppp/options`:

```
/dev/cuad1 115200

crtscts  # enable hardware flow control
```

```
modem  # modem control line
connect "/usr/bin/chat -f /etc/ppp/login.chat.script"
noipdefault # remote PPP serve must supply your IP address
         # if the remote host doesn't send your IP during
                # IPCP negotiation, remove this option
passive         # wait for LCP packets
domain your.domain # put your domain name here

:  # put the IP of remote PPP host here
         # it will be used to route packets via PPP link
                # if you didn't specified the noipdefault option
                # change this line to local_ip:remote_ip

defaultroute # put this if you want that PPP server will be
         # your default router
```

`/etc/ppp/login.chat.script`:

Note: The following should go on a single line.

```
ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDTphone.number
  CONNECT "" TIMEOUT 10 ogin:-\\r-ogin: login-id
  TIMEOUT 5 sword: password
```

Once these are installed and modified correctly, all you need to do is run `pppd`, like so:

```
# pppd
```

27.3.3 Using `pppd` as a Server

`/etc/ppp/options` should contain something similar to the following:

```
crtscts                         # Hardware flow control
netmask 255.255.255.0           # netmask (not required)
192.114.208.20:192.114.208.165  # IP's of local and remote hosts
                                # local ip must be different from one
                                # you assigned to the Ethernet (or other)
                                # interface on your machine.
                                # remote IP is IP address that will be
                                # assigned to the remote machine
domain ppp.foo.com              # your domain
passive                         # wait for LCP
modem                           # modem line
```

The following `/etc/ppp/pppserv` script will tell **pppd** to behave as a server:

```
#!/bin/sh
pgrep -l pppd
pid=‘pgrep pppd‘
if [ "X${pid}" != "X" ] ; then
```

```
        echo 'killing pppd, PID=' ${pid}
        kill ${pid}
fi
pgrep -l kermit
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi

# reset ppp interface
ifconfig ppp0 down
ifconfig ppp0 delete

# enable autoanswer mode
kermit -y /etc/ppp/kermit.ans

# run ppp
pppd /dev/tty01 19200
```

Use this `/etc/ppp/pppservdown` script to stop the server:

```
#!/bin/sh
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill ${pid}
fi
pgrep -l kermit
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi
ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.noans
```

The following **Kermit** script (`/etc/ppp/kermit.ans`) will enable/disable autoanswer mode on your modem. It should look like this:

```
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 ATS0=1\13   ; change this to out ATS0=0\13 if you want to disable
                ; autoanswer mode
inp 5 OK
echo \13
exit
```

A script named `/etc/ppp/kermit.dial` is used for dialing and authenticating on the remote host. You will need to customize it for your needs. Put your login and password in this script; you will also need to change the input statement depending on responses from your modem and remote host.

```
;
; put the com line attached to the modem here:
;
set line /dev/tty01
;
; put the modem speed here:
;
set speed 19200
set file type binary            ; full 8 bit file xfer
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                ; Then SET CARRIER if necessary,
set dial display on             ; Then SET DIAL if necessary,
set input echo on
set input timeout proceed
set input case ignore
def \%x 0                       ; login prompt counter
goto slhup

:slcmd                          ; put the modem in command mode
echo Put the modem in command mode.
clear                           ; Clear unread characters from input buffer
pause 1
output +++                      ; hayes escape sequence
input 1 OK\13\10                ; wait for OK
if success goto slhup
output \13
```

```
pause 1
output at\13
input 1 OK\13\10
if fail goto slcmd              ; if modem doesn't answer OK, try again

:slhup                          ; hang up the phone
clear                           ; Clear unread characters from input buffer
pause 1
echo Hanging up the phone.
output ath0\13                  ; hayes command for on hook
input 2 OK\13\10
if fail goto slcmd              ; if no OK answer, put modem in command mode

:sldial                         ; dial the number
pause 1
echo Dialing.
output atdt9,550311\13\10               ; put phone number here
assign \%x 0                    ; zero the time counter

:look
clear                           ; Clear unread characters from input buffer
increment \%x                   ; Count the seconds
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                        ; login
assign \%x 0                    ; zero the time counter
pause 1
echo Looking for login prompt.

:slloop
increment \%x                   ; Count the seconds
clear                           ; Clear unread characters from input buffer
output \13
;
; put your expected login prompt here:
;
input 1 {Username: }
if success goto sluid
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
```

```
if < \%x 10 goto slloop         ; try 10 times to get a login prompt
else goto slhup                 ; hang up and start again if 10 failures

:sluid
;
; put your userid here:
;
output ppp-login\13
input 1 {Password: }
;
; put your password here:
;
output ppp-password\13
input 1 {Entering SLIP mode.}
echo
quit

:slnodial
echo \7No dialtone.  Check the telephone line!\7
exit 1

; local variables:
; mode: csh
; comment-start: "; "
; comment-start-skip: "; "
; end:
```

27.4 Troubleshooting PPP Connections

> **Warning:** As of FreeBSD 8.0, the uart(4) driver replaces the sio(4) driver. Device nodes for serial ports have been renamed from `/dev/cuadN` to `/dev/cuauN` and from `/dev/ttydN` to `/dev/ttyuN`. FreeBSD 7.X users will have to adapt the following documentation according to these changes.

This section covers a few issues which may arise when using PPP over a modem connection. For instance, perhaps you need to know exactly what prompts the system you are dialing into will present. Some ISPs present the `ssword` prompt, and others will present `password`; if the `ppp` script is not written accordingly, the login attempt will fail. The most common way to debug `ppp` connections is by connecting manually. The following information will walk you through a manual connection step by step.

27.4.1 Check the Device Nodes

When using a custom kernel, make sure to include the following line in your kernel configuration file:

```
device   uart
```

The `uart` device is already included in the `GENERIC` kernel, so no additional steps are necessary in this case. Just check the `dmesg` output for the modem device with:

```
# dmesg | grep uart
```

You should get some pertinent output about the `uart` devices. These are the COM ports we need. If your modem acts like a standard serial port then you should see it listed on `uart1`, or `COM2`. If so, you are not required to rebuild the kernel. When matching up sio modem is on `uart1` or `COM2` if you are in DOS, then your modem device would be `/dev/cuau1`.

27.4.2 Connecting Manually

Connecting to the Internet by manually controlling `ppp` is quick, easy, and a great way to debug a connection or just get information on how your ISP treats `ppp` client connections. Lets start **PPP** from the command line. Note that in all of our examples we will use *example* as the hostname of the machine running **PPP**. You start `ppp` by just typing `ppp`:

```
# ppp
```

We have now started `ppp`.

```
ppp ON example> set device /dev/cuau1
```

We set our modem device, in this case it is `cuau1`.

```
ppp ON example> set speed 115200
```

Set the connection speed, in this case we are using 115,200 kbps.

```
ppp ON example> enable dns
```

Tell `ppp` to configure our resolver and add the nameserver lines to `/etc/resolv.conf`. If `ppp` cannot determine our hostname, we can set one manually later.

```
ppp ON example> term
```

Switch to “terminal” mode so that we can manually control the modem.

```
deflink: Entering terminal mode on /dev/cuau1
type '~h' for help

at
OK
atdt123456789
```

Use `at` to initialize the modem, then use `atdt` and the number for your ISP to begin the dial in process.

```
CONNECT
```

Confirmation of the connection, if we are going to have any connection problems, unrelated to hardware, here is where we will attempt to resolve them.

```
ISP Login:myusername
```

Here you are prompted for a username, return the prompt with the username that was provided by the ISP.

```
ISP Pass:mypassword
```

This time we are prompted for a password, just reply with the password that was provided by the ISP. Just like logging into FreeBSD, the password will not echo.

```
Shell or PPP:ppp
```

Depending on your ISP this prompt may never appear. Here we are being asked if we wish to use a shell on the provider, or to start `ppp`. In this example, we have chosen to use `ppp` as we want an Internet connection.

```
Ppp ON example>
```

Notice that in this example the first `p` has been capitalized. This shows that we have successfully connected to the ISP.

```
PPp ON example>
```

We have successfully authenticated with our ISP and are waiting for the assigned IP address.

```
PPP ON example>
```

We have made an agreement on an IP address and successfully completed our connection.

```
PPP ON example>add default HISADDR
```

Here we add our default route, we need to do this before we can talk to the outside world as currently the only established connection is with the peer. If this fails due to existing routes you can put a bang character `!` in front of the `add`. Alternatively, you can set this before making the actual connection and it will negotiate a new route accordingly.

If everything went good we should now have an active connection to the Internet, which could be thrown into the background using **CTRL**+**z** If you notice the `PPP` return to `ppp` then we have lost our connection. This is good to know because it shows our connection status. Capital P's show that we have a connection to the ISP and lowercase p's show that the connection has been lost for whatever reason. `ppp` only has these 2 states.

27.4.2.1 Debugging

If you have a direct line and cannot seem to make a connection, then turn hardware flow CTS/RTS to off with the `set ctsrts off`. This is mainly the case if you are connected to some **PPP** capable terminal servers, where **PPP** hangs when it tries to write data to your communication link, so it would be waiting for a CTS, or Clear To Send signal which may never come. If you use this option however, you should also use the `set accmap` option, which may be required to defeat hardware dependent on passing certain characters from end to end, most of the time XON/XOFF. See the ppp(8) manual page for more information on this option, and how it is used.

If you have an older modem, you may need to use the `set parity even`. Parity is set at none be default, but is used for error checking (with a large increase in traffic) on older modems and some ISPs. You may need this option for the Compuserve ISP.

PPP may not return to the command mode, which is usually a negotiation error where the ISP is waiting for your side to start negotiating. At this point, using the `~p` command will force ppp to start sending the configuration information.

If you never obtain a login prompt, then most likely you need to use PAP or CHAP authentication instead of the UNIX style in the example above. To use PAP or CHAP just add the following options to **PPP** before going into terminal mode:

```
ppp ON example> set authname myusername
```

Where *myusername* should be replaced with the username that was assigned by the ISP.

```
ppp ON example> set authkey mypassword
```

Where *mypassword* should be replaced with the password that was assigned by the ISP.

If you connect fine, but cannot seem to find any domain name, try to use ping(8) with an IP address and see if you can get any return information. If you experience 100 percent (100%) packet loss, then it is most likely that you were not assigned a default route. Double check that the option `add default HISADDR` was set during the connection. If you can connect to a remote IP address then it is possible that a resolver address has not been added to the `/etc/resolv.conf`. This file should look like:

```
domain example.com
nameserver x.x.x.x
nameserver y.y.y.y
```

Where *x.x.x.x* and *y.y.y.y* should be replaced with the IP address of your ISP's DNS servers. This information may or may not have been provided when you signed up, but a quick call to your ISP should remedy that.

You could also have syslog(3) provide a logging function for your **PPP** connection. Just add:

```
!ppp
*.*     /var/log/ppp.log
```

to `/etc/syslog.conf`. In most cases, this functionality already exists.

27.5 Using PPP over Ethernet (PPPoE)

This section describes how to set up PPP over Ethernet (PPPoE).

27.5.1 Configuring the Kernel

No kernel configuration is necessary for PPPoE any longer. If the necessary netgraph support is not built into the kernel, it will be dynamically loaded by **ppp**.

27.5.2 Setting Up `ppp.conf`

Here is an example of a working `ppp.conf`:

```
default:
  set log Phase tun command # you can add more detailed logging if you wish
  set ifaddr 10.0.0.1/0 10.0.0.2/0
```

```
name_of_service_provider:
  set device PPPoE:xl1 # replace xl1 with your Ethernet device
  set authname YOURLOGINNAME
  set authkey YOURPASSWORD
  set dial
  set login
  add default HISADDR
```

27.5.3 Running ppp

As `root`, you can run:

```
# ppp -ddial name_of_service_provider
```

27.5.4 Starting ppp at Boot

Add the following to your `/etc/rc.conf` file:

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_nat="YES" # if you want to enable nat for your local network, otherwise NO
ppp_profile="name_of_service_provider"
```

27.5.5 Using a PPPoE Service Tag

Sometimes it will be necessary to use a service tag to establish your connection. Service tags are used to distinguish between different PPPoE servers attached to a given network.

You should have been given any required service tag information in the documentation provided by your ISP. If you cannot locate it there, ask your ISP's tech support personnel.

As a last resort, you could try the method suggested by the Roaring Penguin PPPoE (http://www.roaringpenguin.com/pppoe/) program which can be found in the Ports Collection. Bear in mind however, this may de-program your modem and render it useless, so think twice before doing it. Simply install the program shipped with the modem by your provider. Then, access the System menu from the program. The name of your profile should be listed there. It is usually *ISP*.

The profile name (service tag) will be used in the PPPoE configuration entry in `ppp.conf` as the provider part of the `set device` command (see the ppp(8) manual page for full details). It should look like this:

```
set device PPPoE:xl1:ISP
```

Do not forget to change *`xl1`* to the proper device for your Ethernet card.

Do not forget to change *`ISP`* to the profile you have just found above.

For additional information, see:

- Cheaper Broadband with FreeBSD on DSL (http://renaud.waldura.com/doc/freebsd/pppoe/) by Renaud Waldura.

- Nutzung von T-DSL und T-Online mit FreeBSD (http://www.ruhr.de/home/nathan/FreeBSD/tdsl-freebsd.html) by Udo Erdelhoff (in German).

27.5.6 PPPoE with a 3Com® HomeConnect® ADSL Modem Dual Link

This modem does not follow RFC 2516 (http://www.faqs.org/rfcs/rfc2516.html) (*A Method for transmitting PPP over Ethernet (PPPoE)*, written by L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone, and R. Wheeler). Instead, different packet type codes have been used for the Ethernet frames. Please complain to 3Com (http://www.3com.com/) if you think it should comply with the PPPoE specification.

In order to make FreeBSD capable of communicating with this device, a sysctl must be set. This can be done automatically at boot time by updating `/etc/sysctl.conf`:

```
net.graph.nonstandard_pppoe=1
```

or can be done immediately with the command:

```
# sysctl net.graph.nonstandard_pppoe=1
```

Unfortunately, because this is a system-wide setting, it is not possible to talk to a normal PPPoE client or server and a 3Com HomeConnect® ADSL Modem at the same time.

27.6 Using PPP over ATM (PPPoA)

The following describes how to set up PPP over ATM (PPPoA). PPPoA is a popular choice among European DSL providers.

27.6.1 Using PPPoA with the Alcatel SpeedTouch™ USB

PPPoA support for this device is supplied as a port in FreeBSD because the firmware is distributed under Alcatel's license agreement (http://www.speedtouchdsl.com/disclaimer_lx.htm) and can not be redistributed freely with the base system of FreeBSD.

To install the software, simply use the Ports Collection. Install the `net/pppoa` port and follow the instructions provided with it.

Like many USB devices, the Alcatel SpeedTouch™ USB needs to download firmware from the host computer to operate properly. It is possible to automate this process in FreeBSD so that this transfer takes place whenever the device is plugged into a USB port. The following information can be added to the `/etc/usbd.conf` file to enable this automatic firmware transfer. This file must be edited as the `root` user.

```
device "Alcatel SpeedTouch USB"
    devname "ugen[0-9]+"
    vendor 0x06b9
    product 0x4061
    attach "/usr/local/sbin/modem_run -f /usr/local/libdata/mgmt.o"
```

To enable the USB daemon, **usbd**, put the following the line into `/etc/rc.conf`:

```
usbd_enable="YES"
```

It is also possible to set up **ppp** to dial up at startup. To do this add the following lines to `/etc/rc.conf`. Again, for this procedure you will need to be logged in as the `root` user.

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_profile="adsl"
```

For this to work correctly you will need to have used the sample `ppp.conf` which is supplied with the `net/pppoa` port.

27.6.2 Using mpd

You can use **mpd** to connect to a variety of services, in particular PPTP services. You can find **mpd** in the Ports Collection, `net/mpd`. Many ADSL modems require that a PPTP tunnel is created between the modem and computer, one such modem is the Alcatel SpeedTouch Home.

First you must install the port, and then you can configure **mpd** to suit your requirements and provider settings. The port places a set of sample configuration files which are well documented in *`PREFIX`*`/etc/mpd/`. Note here that *`PREFIX`* means the directory into which your ports are installed, this defaults to `/usr/local/`. A complete guide to configure **mpd** is available in HTML format once the port has been installed. It is placed in *`PREFIX`*`/share/doc/mpd/`. Here is a sample configuration for connecting to an ADSL service with **mpd**. The configuration is spread over two files, first the `mpd.conf`:

```
default:
    load adsl

adsl:
    new -i ng0 adsl adsl
    set bundle authname username ❶
    set bundle password password 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 The username used to authenticate with your ISP.

❷ The password used to authenticate with your ISP.

The `mpd.links` file contains information about the link, or links, you wish to establish. An example `mpd.links` to accompany the above example is given beneath:

```
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 The IP address of your FreeBSD computer which you will be using **mpd** from.

❷ The IP address of your ADSL modem. For the Alcatel SpeedTouch Home this address defaults to `10.0.0.138`.

It is possible to initialize the connection easily by issuing the following command as `root`:

```
# mpd -b adsl
```

You can see the status of the connection with the following command:

```
% ifconfig ng0
ng0: flags=88d1<UP,POINTOPOINT,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
     inet 216.136.204.117 --> 204.152.186.171 netmask 0xffffffff
```

Using **mpd** is the recommended way to connect to an ADSL service with FreeBSD.

27.6.3 Using pptpclient

It is also possible to use FreeBSD to connect to other PPPoA services using `net/pptpclient`.

To use `net/pptpclient` to connect to a DSL service, install the port or package and edit your `/etc/ppp/ppp.conf`. You will need to be `root` to perform both of these operations. An example section of `ppp.conf` is given below. For further information on `ppp.conf` options consult the **ppp** manual page, ppp(8).

```
adsl:
 set log phase chat lcp ipcp ccp tun command
 set timeout 0
 enable dns
 set authname username ❶
 set authkey password ❷
 set ifaddr 0 0
 add default HISADDR
```

❶ The username of your account with the DSL provider.

❷ The password for your account.

> **Warning:** Because you must put your account's password in the `ppp.conf` file in plain text form you should make sure than nobody can read the contents of this file. The following series of commands will make sure the file is only readable by the `root` account. Refer to the manual pages for chmod(1) and chown(8) for further information.
>
> ```
> # chown root:wheel /etc/ppp/ppp.conf
> ```

```
# chmod 600 /etc/ppp/ppp.conf
```

This will open a tunnel for a PPP session to your DSL router. Ethernet DSL modems have a preconfigured LAN IP address which you connect to. In the case of the Alcatel SpeedTouch Home this address is `10.0.0.138`. Your router documentation should tell you which address your device uses. To open the tunnel and start a PPP session execute the following command:

```
# pptp address adsl
```

> **Tip:** You may wish to add an ampersand ("&") to the end of the previous command because **pptp** will not return your prompt to you otherwise.

A `tun` virtual tunnel device will be created for interaction between the **pptp** and **ppp** processes. Once you have been returned to your prompt, or the **pptp** process has confirmed a connection you can examine the tunnel like so:

```
% ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
        inet 216.136.204.21 --> 204.152.186.171 netmask 0xffffff00
        Opened by PID 918
```

If you are unable to connect, check the configuration of your router, which is usually accessible via **telnet** or with a web browser. If you still cannot connect you should examine the output of the `pptp` command and the contents of the **ppp** log file, `/var/log/ppp.log` for clues.

27.7 Using SLIP

> **Warning:** This section applies and is valid only for FreeBSD 7.X.

27.7.1 Setting Up a SLIP Client

The following is one way to set up a FreeBSD machine for SLIP on a static host network. For dynamic hostname assignments (your address changes each time you dial up), you probably need to have a more complex setup.

First, determine which serial port your modem is connected to. Many people set up a symbolic link, such as `/dev/modem`, to point to the real device name, `/dev/cuadN`. This allows you to abstract the actual device name should you ever need to move the modem to a different port. It can become quite cumbersome when you need to fix a bunch of files in `/etc` and `.kermrc` files all over the system!

> **Note:** `/dev/cuad0` is `COM1`, `/dev/cuad1` is `COM2`, etc.

Make sure you have the following in your kernel configuration file:

```
device   sl
```

It is included in the `GENERIC` kernel, so this should not be a problem unless you have deleted it.

27.7.1.1 Things You Have to Do Only Once

1. Add your home machine, the gateway and nameservers to your `/etc/hosts` file. Ours looks like this:

   ```
   127.0.0.1               localhost loghost
   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. Make sure you have `files` before `dns` in the `hosts:` section of your `/etc/nsswitch.conf` file. Without these parameters funny things may happen.
3. Edit the `/etc/rc.conf` file.
   1. Set your hostname by editing the line that says:

      ```
      hostname="myname.my.domain"
      ```

      Your machine's full Internet hostname should be placed here.
   2. Designate the default router by changing the line:

      ```
      defaultrouter="NO"
      ```

      to:

      ```
      defaultrouter="slip-gateway"
      ```
4. Make a file `/etc/resolv.conf` which contains:

   ```
   domain CS.Example.EDU
   nameserver 128.32.136.9
   nameserver 128.32.136.12
   ```

   As you can see, these set up the nameserver hosts. Of course, the actual domain names and addresses depend on your environment.
5. Set the password for `root` and `toor` (and any other accounts that do not have a password).
6. Reboot your machine and make sure it comes up with the correct hostname.

27.7.1.2 Making a SLIP Connection

1. Dial up, type `slip` at the prompt, enter your machine name and password. What is required to be entered depends on your environment. If you use **Kermit**, you can try a script like this:

   ```
   # kermit setup
   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
# The next macro will dial up and login
define slip dial 643-9600, input 10 =>, if failure stop, -
output slip\x0d, input 10 Username:, if failure stop, -
output silvia\x0d, input 10 Password:, if failure stop, -
output ***\x0d, echo \x0aCONNECTED\x0a
```

Of course, you have to change the username and password to fit yours. After doing so, you can just type `slip` from the **Kermit** prompt to connect.

> **Note:** Leaving your password in plain text anywhere in the filesystem is generally a *bad* idea. Do it at your own risk.

2. Leave the **Kermit** there (you can suspend it by **Ctrl-z**) and as `root`, type:

   ```
   # slattach -h -c -s 115200 /dev/modem
   ```

   If you are able to `ping` hosts on the other side of the router, you are connected! If it does not work, you might want to try `-a` instead of `-c` as an argument to `slattach`.

27.7.1.3 How to Shutdown the Connection

Do the following:

```
# kill -INT `cat /var/run/slattach.modem.pid`
```

to kill `slattach`. Keep in mind you must be `root` to do the above. Then go back to `kermit` (by running `fg` if you suspended it) and exit from it (**q**).

The slattach(8) manual page says you have to use `ifconfig sl0 down` to mark the interface down, but this does not seem to make any difference. (`ifconfig sl0` reports the same thing.)

Some times, your modem might refuse to drop the carrier. In that case, simply start `kermit` and quit it again. It usually goes out on the second try.

27.7.1.4 Troubleshooting

If it does not work, feel free to ask on freebsd-net (http://lists.FreeBSD.org/mailman/listinfo/freebsd-net) mailing list. The things that people tripped over so far:

- Not using `-c` or `-a` in `slattach` (This should not be fatal, but some users have reported that this solves their problems.)
- Using `s10` instead of `sl0` (might be hard to see the difference on some fonts).
- Try `ifconfig sl0` to see your interface status. For example, you might get:

  ```
  # ifconfig sl0
  sl0: flags=10<POINTOPOINT>
          inet 136.152.64.181 --> 136.152.64.1 netmask ffffff00
  ```

- If you get `no route to host` messages from ping(8), there may be a problem with your routing table. You can use the `netstat -r` command to display the current routes :

```
# 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)
```

  The preceding examples are from a relatively busy system. The numbers on your system will vary depending on network activity.

27.7.2 Setting Up a SLIP Server

This document provides suggestions for setting up SLIP Server services on a FreeBSD system, which typically means configuring your system to automatically start up connections upon login for remote SLIP clients.

27.7.2.1 Prerequisites

This section is very technical in nature, so background knowledge is required. It is assumed that you are familiar with the TCP/IP network protocol, and in particular, network and node addressing, network address masks, subnetting, routing, and routing protocols, such as RIP. Configuring SLIP services on a dial-up server requires a knowledge of these concepts, and if you are not familiar with them, please read a copy of either Craig Hunt's *TCP/IP Network Administration* published by O'Reilly & Associates, Inc. (ISBN Number 0-937175-82-X), or Douglas Comer's books on the TCP/IP protocol.

It is further assumed that you have already set up your modem(s) and configured the appropriate system files to allow logins through your modems. If you have not prepared your system for this yet, please see Section 26.4 for details on dialup services configuration. You may also want to check the manual pages for sio(4) for information on the serial port device driver and ttys(5), gettytab(5), getty(8), & init(8) for information relevant to configuring the system to accept logins on modems, and perhaps stty(1) for information on setting serial port parameters (such as `clocal` for directly-connected serial interfaces).

27.7.2.2 Quick Overview

In its typical configuration, using FreeBSD as a SLIP server works as follows: a SLIP user dials up your FreeBSD SLIP Server system and logs in with a special SLIP login ID that uses `/usr/sbin/sliplogin` as the special user's shell. The `sliplogin` program browses the file `/etc/sliphome/slip.hosts` to find a matching line for the special user, and if it finds a match, connects the serial line to an available SLIP interface and then runs the shell script `/etc/sliphome/slip.login` to configure the SLIP interface.

27.7.2.2.1 An Example of a SLIP Server Login

For example, if a SLIP user ID were `Shelmerg`, `Shelmerg`'s entry in `/etc/master.passwd` would look something like this:

```
Shelmerg:password:1964:89::0:0:Guy Helmer - SLIP:/usr/users/Shelmerg:/usr/sbin/sliplogin
```

When `Shelmerg` logs in, `sliplogin` will search `/etc/sliphome/slip.hosts` for a line that had a matching user ID; for example, there may be a line in `/etc/sliphome/slip.hosts` that reads:

```
Shelmerg        dc-slip sl-helmer       0xfffffc00    autocomp
```

`sliplogin` will find that matching line, hook the serial line into the next available SLIP interface, and then execute `/etc/sliphome/slip.login` like this:

```
/etc/sliphome/slip.login 0 19200 Shelmerg dc-slip sl-helmer 0xfffffc00 autocomp
```

If all goes well, `/etc/sliphome/slip.login` will issue an `ifconfig` for the SLIP interface to which `sliplogin` attached itself (SLIP interface 0, in the above example, which was the first parameter in the list given to `slip.login`) to set the local IP address (`dc-slip`), remote IP address (`sl-helmer`), network mask for the SLIP interface (`0xfffffc00`), and any additional flags (`autocomp`). If something goes wrong, `sliplogin` usually logs good informational messages via the **syslogd** daemon facility, which usually logs to `/var/log/messages` (see the manual pages for syslogd(8) and syslog.conf(5) and perhaps check `/etc/syslog.conf` to see to what **syslogd** is logging and where it is logging to).

27.7.2.3 Kernel Configuration

FreeBSD's default kernel (`GENERIC`) comes with SLIP (sl(4)) support; in case of a custom kernel, you have to add the following line to your kernel configuration file:

```
device   sl
```

By default, your FreeBSD machine will not forward packets. If you want your FreeBSD SLIP Server to act as a router, you will have to edit the `/etc/rc.conf` file and change the setting of the `gateway_enable` variable to `YES`. This will make sure that setting the routing option will be persistent after a reboot.

To apply the settings immediately you can execute the following command as `root`:

```
# /etc/rc.d/routing start
```

Please refer to Chapter 8 on Configuring the FreeBSD Kernel for help in reconfiguring your kernel.

27.7.2.4 Sliplogin Configuration

As mentioned earlier, there are three files in the `/etc/sliphome` directory that are part of the configuration for `/usr/sbin/sliplogin` (see sliplogin(8) for the actual manual page for `sliplogin`): `slip.hosts`, which defines the SLIP users and their associated IP addresses; `slip.login`, which usually just configures the SLIP interface; and (optionally) `slip.logout`, which undoes `slip.login`'s effects when the serial connection is terminated.

27.7.2.4.1 `slip.hosts` Configuration

`/etc/sliphome/slip.hosts` contains lines which have at least four items separated by whitespace:

- SLIP user's login ID
- Local address (local to the SLIP server) of the SLIP link
- Remote address of the SLIP link
- Network mask

The local and remote addresses may be host names (resolved to IP addresses by `/etc/hosts` or by the domain name service, depending on your specifications in the file `/etc/nsswitch.conf`), and the network mask may be a name that can be resolved by a lookup into `/etc/networks`. On a sample system, `/etc/sliphome/slip.hosts` looks like this:

```
#
# login local-addr      remote-addr     mask            opt1    opt2
#                                               (normal,compress,noicmp)
#
Shelmerg  dc-slip       sl-helmerg      0xfffffc00      autocomp
```

At the end of the line is one or more of the options:

- `normal` — no header compression
- `compress` — compress headers
- `autocomp` — compress headers if the remote end allows it
- `noicmp` — disable ICMP packets (so any "ping" packets will be dropped instead of using up your bandwidth)

Your choice of local and remote addresses for your SLIP links depends on whether you are going to dedicate a TCP/IP subnet or if you are going to use "proxy ARP" on your SLIP server (it is not "true" proxy ARP, but that is the terminology used in this section to describe it). If you are not sure which method to select or how to assign IP addresses, please refer to the TCP/IP books referenced in the SLIP Prerequisites (Section 27.7.2.1) and/or consult your IP network manager.

If you are going to use a separate subnet for your SLIP clients, you will need to allocate the subnet number out of your assigned IP network number and assign each of your SLIP client's IP numbers out of that subnet. Then, you will probably need to configure a static route to the SLIP subnet via your SLIP server on your nearest IP router.

Otherwise, if you will use the "proxy ARP" method, you will need to assign your SLIP client's IP addresses out of your SLIP server's Ethernet subnet, and you will also need to adjust your `/etc/sliphome/slip.login` and `/etc/sliphome/slip.logout` scripts to use arp(8) to manage the "proxy ARP" entries in the SLIP server's ARP table.

27.7.2.4.2 `slip.login` Configuration

The typical `/etc/sliphome/slip.login` file looks like this:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90
```

```
#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
```

This `slip.login` file merely runs `ifconfig` for the appropriate SLIP interface with the local and remote addresses and network mask of the SLIP interface.

If you have decided to use the "proxy ARP" method (instead of using a separate subnet for your SLIP clients), your `/etc/sliphome/slip.login` file will need to look something like this:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90

#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
# Answer ARP requests for the SLIP client with our Ethernet addr
/usr/sbin/arp -s $5 00:11:22:33:44:55 pub
```

The additional line in this `slip.login`, `arp -s $5 00:11:22:33:44:55 pub`, creates an ARP entry in the SLIP server's ARP table. This ARP entry causes the SLIP server to respond with the SLIP server's Ethernet MAC address whenever another IP node on the Ethernet asks to speak to the SLIP client's IP address.

When using the example above, be sure to replace the Ethernet MAC address (`00:11:22:33:44:55`) with the MAC address of your system's Ethernet card, or your "proxy ARP" will definitely not work! You can discover your SLIP server's Ethernet MAC address by looking at the results of running `netstat -i`; the second line of the output should look something like:

```
ed0   1500  <Link>0.2.c1.28.5f.4a         191923 0   129457     0   116
```

This indicates that this particular system's Ethernet MAC address is `00:02:c1:28:5f:4a` — the periods in the Ethernet MAC address given by `netstat -i` must be changed to colons and leading zeros should be added to each single-digit hexadecimal number to convert the address into the form that arp(8) desires; see the manual page on arp(8) for complete information on usage.

> **Note:** When you create `/etc/sliphome/slip.login` and `/etc/sliphome/slip.logout`, the "execute" bit (i.e., `chmod 755 /etc/sliphome/slip.login /etc/sliphome/slip.logout`) must be set, or `sliplogin` will be unable to execute it.

27.7.2.4.3 `slip.logout` Configuration

`/etc/sliphome/slip.logout` is not strictly needed (unless you are implementing "proxy ARP"), but if you decide to create it, this is an example of a basic `slip.logout` script:

```
#!/bin/sh -
#
#       slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
```

If you are using "proxy ARP", you will want to have `/etc/sliphome/slip.logout` remove the ARP entry for the SLIP client:

```
#!/bin/sh -
#
#       @(#)slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
# Quit answering ARP requests for the SLIP client
/usr/sbin/arp -d $5
```

The `arp -d $5` removes the ARP entry that the "proxy ARP" `slip.login` added when the SLIP client logged in.

It bears repeating: make sure `/etc/sliphome/slip.logout` has the execute bit set after you create it (i.e., `chmod 755 /etc/sliphome/slip.logout`).

27.7.2.5 Routing Considerations

If you are not using the "proxy ARP" method for routing packets between your SLIP clients and the rest of your network (and perhaps the Internet), you will probably have to add static routes to your closest default router(s) to route your SLIP clients subnet via your SLIP server.

27.7.2.5.1 Static Routes

Adding static routes to your nearest default routers can be troublesome (or impossible if you do not have authority to do so...). If you have a multiple-router network in your organization, some routers, such as those made by Cisco and Proteon, may not only need to be configured with the static route to the SLIP subnet, but also need to be told which

static routes to tell other routers about, so some expertise and troubleshooting/tweaking may be necessary to get static-route-based routing to work.

Chapter 28
Electronic Mail

28.1 Synopsis

"Electronic Mail", better known as email, is one of the most widely used forms of communication today. This chapter provides a basic introduction to running a mail server on FreeBSD, as well as an introduction to sending and receiving email using FreeBSD; however, it is not a complete reference and in fact many important considerations are omitted. For more complete coverage of the subject, the reader is referred to the many excellent books listed in Appendix B.

After reading this chapter, you will know:

- What software components are involved in sending and receiving electronic mail.
- Where basic **sendmail** configuration files are located in FreeBSD.
- The difference between remote and local mailboxes.
- How to block spammers from illegally using your mail server as a relay.
- How to install and configure an alternate Mail Transfer Agent on your system, replacing **sendmail**.
- How to troubleshoot common mail server problems.
- How to use SMTP with UUCP.
- How to set up the system to send mail only.
- How to use mail with a dialup connection.
- How to configure SMTP Authentication for added security.
- How to install and use a Mail User Agent, such as **mutt** to send and receive email.
- How to download your mail from a remote POP or IMAP server.
- How to automatically apply filters and rules to incoming email.

Before reading this chapter, you should:

- Properly set up your network connection (Chapter 31).
- Properly set up the DNS information for your mail host (Chapter 29).
- Know how to install additional third-party software (Chapter 4).

28.2 Using Electronic Mail

There are five major parts involved in an email exchange. They are: the user program, the server daemon, DNS, a remote or local mailbox, and of course, the mailhost itself.

28.2.1 The User Program

This includes command line programs such as **mutt**, **alpine**, **elm**, and `mail`, and GUI programs such as **balsa**, **xfmail** to name a few, and something more "sophisticated" like a WWW browser. These programs simply pass off the email transactions to the local "mailhost", either by calling one of the server daemons available, or delivering it over TCP.

28.2.2 Mailhost Server Daemon

FreeBSD ships with **sendmail** by default, but also support numerous other mail server daemons, just some of which include:

- **exim**;
- **postfix**;
- **qmail**.

The server daemon usually has two functions—it is responsible for receiving incoming mail as well as delivering outgoing mail. It is *not* responsible for the collection of mail using protocols such as POP or IMAP to read your email, nor does it allow connecting to local `mbox` or Maildir mailboxes. You may require an additional daemon for that.

> **Warning:** Older versions of **sendmail** have some serious security issues which may result in an attacker gaining local and/or remote access to your machine. Make sure that you are running a current version to avoid these problems. Optionally, install an alternative MTA from the FreeBSD Ports Collection.

28.2.3 Email and DNS

The Domain Name System (DNS) and its daemon `named` play a large role in the delivery of email. In order to deliver mail from your site to another, the server daemon will look up the remote site in the DNS to determine the host that will receive mail for the destination. This process also occurs when mail is sent from a remote host to your mail server.

DNS is responsible for mapping hostnames to IP addresses, as well as for storing information specific to mail delivery, known as MX records. The MX (Mail eXchanger) record specifies which host, or hosts, will receive mail for a particular domain. If you do not have an MX record for your hostname or domain, the mail will be delivered directly to your host provided you have an A record pointing your hostname to your IP address.

You may view the MX records for any domain by using the host(1) command, as seen in the example below:

```
% host -t mx FreeBSD.org
FreeBSD.org mail is handled (pri=10) by mx1.FreeBSD.org
```

28.2.4 Receiving Mail

Receiving mail for your domain is done by the mail host. It will collect all mail sent to your domain and store it either in `mbox` (the default method for storing mail) or Maildir format, depending on your configuration. Once mail has been stored, it may either be read locally using applications such as mail(1) or **mutt**, or remotely accessed and collected using protocols such as POP or IMAP. This means that should you only wish to read mail locally, you are not required to install a POP or IMAP server.

28.2.4.1 Accessing remote mailboxes using POP and IMAP

In order to access mailboxes remotely, you are required to have access to a POP or IMAP server. These protocols allow users to connect to their mailboxes from remote locations with ease. Though both POP and IMAP allow users to remotely access mailboxes, IMAP offers many advantages, some of which are:

- IMAP can store messages on a remote server as well as fetch them.
- IMAP supports concurrent updates.
- IMAP can be extremely useful over low-speed links as it allows users to fetch the structure of messages without downloading them; it can also perform tasks such as searching on the server in order to minimize data transfer between clients and servers.

In order to install a POP or IMAP server, the following steps should be performed:

1. Choose an IMAP or POP server that best suits your needs. The following POP and IMAP servers are well known and serve as some good examples:
   - **qpopper**;
   - **teapop**;
   - **imap-uw**;
   - **courier-imap**;
2. Install the POP or IMAP daemon of your choosing from the ports collection.
3. Where required, modify `/etc/inetd.conf` to load the POP or IMAP server.

> **Warning:** It should be noted that both POP and IMAP transmit information, including username and password credentials in clear-text. This means that if you wish to secure the transmission of information across these protocols, you should consider tunneling sessions over ssh(1). Tunneling sessions is described in Section 14.10.8.

28.2.4.2 Accessing local mailboxes

Mailboxes may be accessed locally by directly utilizing MUAs on the server on which the mailbox resides. This can be done using applications such as **mutt** or mail(1).

28.2.5 The Mail Host

The mail host is the name given to a server that is responsible for delivering and receiving mail for your host, and possibly your network.

28.3 sendmail Configuration

sendmail(8) is the default Mail Transfer Agent (MTA) in FreeBSD. **sendmail**'s job is to accept mail from Mail User Agents (MUA) and deliver it to the appropriate mailer as defined by its configuration file. **sendmail** can also accept network connections and deliver mail to local mailboxes or deliver it to another program.

sendmail uses the following configuration files:

Filename	Function
`/etc/mail/access`	**sendmail** access database file
`/etc/mail/aliases`	Mailbox aliases
`/etc/mail/local-host-names`	Lists of hosts **sendmail** accepts mail for
`/etc/mail/mailer.conf`	Mailer program configuration
`/etc/mail/mailertable`	Mailer delivery table
`/etc/mail/sendmail.cf`	**sendmail** master configuration file
`/etc/mail/virtusertable`	Virtual users and domain tables

28.3.1 `/etc/mail/access`

The access database defines what host(s) or IP addresses have access to the local mail server and what kind of access they have. Hosts can be listed as `OK`, `REJECT`, `RELAY` or simply passed to **sendmail**'s error handling routine with a given mailer error. Hosts that are listed as `OK`, which is the default, are allowed to send mail to this host as long as the mail's final destination is the local machine. Hosts that are listed as `REJECT` are rejected for all mail connections. Hosts that have the `RELAY` option for their hostname are allowed to send mail for any destination through this mail server.

Example 28-1. Configuring the sendmail Access Database

```
cyberspammer.com                550 We do not accept mail from spammers
FREE.STEALTH.MAILER@            550 We do not accept mail from spammers
another.source.of.spam          REJECT
okay.cyberspammer.com           OK
128.32                          RELAY
```

In this example we have five entries. Mail senders that match the left hand side of the table are affected by the action on the right side of the table. The first two examples give an error code to **sendmail**'s error handling routine. The message is printed to the remote host when a mail matches the left hand side of the table. The next entry rejects mail from a specific host on the Internet, `another.source.of.spam`. The next entry accepts mail connections from a host `okay.cyberspammer.com`, which is more exact than the `cyberspammer.com` line above. More specific matches override less exact matches. The last entry allows relaying of electronic mail from hosts with an IP address

that begins with `128.32`. These hosts would be able to send mail through this mail server that are destined for other mail servers.

When this file is updated, you need to run `make` in `/etc/mail/` to update the database.

28.3.2 `/etc/mail/aliases`

The aliases database contains a list of virtual mailboxes that are expanded to other user(s), files, programs or other aliases. Here are a few examples that can be used in `/etc/mail/aliases`:

Example 28-2. Mail Aliases

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket:  /dev/null
procmail: "|/usr/local/bin/procmail"
```

The file format is simple; the mailbox name on the left side of the colon is expanded to the target(s) on the right. The first example simply expands the mailbox `root` to the mailbox `localuser`, which is then looked up again in the aliases database. If no match is found, then the message is delivered to the local user `localuser`. The next example shows a mail list. Mail to the mailbox `ftp-bugs` is expanded to the three local mailboxes `joe`, `eric`, and `paul`. Note that a remote mailbox could be specified as <`user@example.com`>. The next example shows writing mail to a file, in this case `/dev/null`. The last example shows sending mail to a program, in this case the mail message is written to the standard input of `/usr/local/bin/procmail` through a UNIX pipe.

When this file is updated, you need to run `make` in `/etc/mail/` to update the database.

28.3.3 `/etc/mail/local-host-names`

This is a list of hostnames sendmail(8) is to accept as the local host name. Place any domains or hosts that **sendmail** is to be receiving mail for. For example, if this mail server was to accept mail for the domain `example.com` and the host `mail.example.com`, its `local-host-names` might look something like this:

```
example.com
mail.example.com
```

When this file is updated, sendmail(8) needs to be restarted to read the changes.

28.3.4 `/etc/mail/sendmail.cf`

sendmail's master configuration file, `sendmail.cf` controls the overall behavior of **sendmail**, including everything from rewriting e-mail addresses to printing rejection messages to remote mail servers. Naturally, with such a diverse role, this configuration file is quite complex and its details are a bit out of the scope of this section. Fortunately, this file rarely needs to be changed for standard mail servers.

The master **sendmail** configuration file can be built from m4(1) macros that define the features and behavior of **sendmail**. Please see `/usr/src/contrib/sendmail/cf/README` for some of the details.

When changes to this file are made, **sendmail** needs to be restarted for the changes to take effect.

28.3.5 `/etc/mail/virtusertable`

The `virtusertable` maps mail addresses for virtual domains and mailboxes to real mailboxes. These mailboxes can be local, remote, aliases defined in `/etc/mail/aliases` or files.

Example 28-3. Example Virtual Domain Mail Map

```
root@example.com                root
postmaster@example.com          postmaster@noc.example.net
@example.com                    joe
```

In the above example, we have a mapping for a domain `example.com`. This file is processed in a first match order down the file. The first item maps <`root@example.com`> to the local mailbox `root`. The next entry maps <`postmaster@example.com`> to the mailbox `postmaster` on the host `noc.example.net`. Finally, if nothing from `example.com` has matched so far, it will match the last mapping, which matches every other mail message addressed to someone at `example.com`. This will be mapped to the local mailbox `joe`.

28.4 Changing Your Mail Transfer Agent

As already mentioned, FreeBSD comes with **sendmail** already installed as your MTA (Mail Transfer Agent). Therefore by default it is in charge of your outgoing and incoming mail.

However, for a variety of reasons, some system administrators want to change their system's MTA. These reasons range from simply wanting to try out another MTA to needing a specific feature or package which relies on another mailer. Fortunately, whatever the reason, FreeBSD makes it easy to make the change.

28.4.1 Install a New MTA

You have a wide choice of MTAs available. A good starting point is the FreeBSD Ports Collection where you will be able to find many. Of course you are free to use any MTA you want from any location, as long as you can make it run under FreeBSD.

Start by installing your new MTA. Once it is installed it gives you a chance to decide if it really fulfills your needs, and also gives you the opportunity to configure your new software before getting it to take over from **sendmail**. When doing this, you should be sure that installing the new software will not attempt to overwrite system binaries such as `/usr/bin/sendmail`. Otherwise, your new mail software has essentially been put into service before you have configured it.

Please refer to your chosen MTA's documentation for information on how to configure the software you have chosen.

28.4.2 Disable sendmail

> **Warning:** If you disable **sendmail**'s outgoing mail service, it is important that you replace it with an alternative mail delivery system. If you choose not to, system functions such as periodic(8) will be unable to deliver their results by e-mail as they would normally expect to. Many parts of your system may expect to have a functional **sendmail**-compatible system. If applications continue to use **sendmail**'s binaries to try to send e-mail after you have disabled them, mail could go into an inactive **sendmail** queue, and never be delivered.

In order to completely disable **sendmail**, including the outgoing mail service, you must use

```
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
```

in `/etc/rc.conf`.

If you only want to disable **sendmail**'s incoming mail service, you should set

```
sendmail_enable="NO"
```

in `/etc/rc.conf`. More information on **sendmail**'s startup options is available from the rc.sendmail(8) manual page.

28.4.3 Running Your New MTA on Boot

The new MTA can be started during boot by adding a configuration line to `/etc/rc.conf` like the following example for postfix:

```
# echo 'postfix_enable="YES"' >> /etc/rc.conf
```

The MTA will now be automatically started during boot.

28.4.4 Replacing sendmail as the System's Default Mailer

The program **sendmail** is so ubiquitous as standard software on UNIX systems that some software just assumes it is already installed and configured. For this reason, many alternative MTA's provide their own compatible implementations of the **sendmail** command-line interface; this facilitates using them as "drop-in" replacements for **sendmail**.

Therefore, if you are using an alternative mailer, you will need to make sure that software trying to execute standard **sendmail** binaries such as `/usr/bin/sendmail` actually executes your chosen mailer instead. Fortunately, FreeBSD provides a system called mailwrapper(8) that does this job for you.

When **sendmail** is operating as installed, you will find something like the following in `/etc/mail/mailer.conf`:

```
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
```

This means that when any of these common commands (such as `sendmail` itself) are run, the system actually invokes a copy of mailwrapper named `sendmail`, which checks `mailer.conf` and executes `/usr/libexec/sendmail/sendmail` instead. This system makes it easy to change what binaries are actually executed when these default `sendmail` functions are invoked.

Therefore if you wanted `/usr/local/supermailer/bin/sendmail-compat` to be run instead of **sendmail**, you could change `/etc/mail/mailer.conf` to read:

```
sendmail  /usr/local/supermailer/bin/sendmail-compat
send-mail /usr/local/supermailer/bin/sendmail-compat
mailq  /usr/local/supermailer/bin/mailq-compat
newaliases /usr/local/supermailer/bin/newaliases-compat
hoststat /usr/local/supermailer/bin/hoststat-compat
purgestat /usr/local/supermailer/bin/purgestat-compat
```

28.4.5 Finishing

Once you have everything configured the way you want it, you should either kill the **sendmail** processes that you no longer need and start the processes belonging to your new software, or simply reboot. Rebooting will also give you the opportunity to ensure that you have correctly configured your system to start your new MTA automatically on boot.

28.5 Troubleshooting

1. Why do I have to use the FQDN for hosts on my site?

You will probably find that the host is actually in a different domain; for example, if you are in `foo.bar.edu` and you wish to reach a host called `mumble` in the `bar.edu` domain, you will have to refer to it by the fully-qualified domain name, `mumble.bar.edu`, instead of just `mumble`.

Traditionally, this was allowed by BSD BIND resolvers. However the current version of **BIND** that ships with FreeBSD no longer provides default abbreviations for non-fully qualified domain names other than the domain you are in. So an unqualified host `mumble` must either be found as `mumble.foo.bar.edu`, or it will be searched for in the root domain.

This is different from the previous behavior, where the search continued across `mumble.bar.edu`, and `mumble.edu`. Have a look at RFC 1535 for why this was considered bad practice, or even a security hole.

As a good workaround, you can place the line:

```
search foo.bar.edu bar.edu
```

instead of the previous:

```
domain foo.bar.edu
```

into your `/etc/resolv.conf`. However, make sure that the search order does not go beyond the “boundary between local and public administration”, as RFC 1535 calls it.

2. sendmail says `mail loops back to myself`

This is answered in the **sendmail** FAQ as follows:

```
I'm getting these error messages:

553 MX list for domain.net points back to relay.domain.net
554 <user@domain.net>... Local configuration error
```

```
How can I solve this problem?

You have asked mail to the domain (e.g., domain.net) to be
forwarded to a specific host (in this case, relay.domain.net)
by using an MX record, but the relay machine does not recognize
itself as domain.net. Add domain.net to /etc/mail/local-host-names
[known as /etc/sendmail.cw prior to version 8.10]
(if you are using FEATURE(use_cw_file)) or add "Cw domain.net"
to /etc/mail/sendmail.cf.
```

The **sendmail** FAQ can be found at http://www.sendmail.org/faq/ and is recommended reading if you want to do any "tweaking" of your mail setup.

3. How can I run a mail server on a dial-up PPP host?

You want to connect a FreeBSD box on a LAN to the Internet. The FreeBSD box will be a mail gateway for the LAN. The PPP connection is non-dedicated.

There are at least two ways to do this. One way is to use UUCP.

Another way is to get a full-time Internet server to provide secondary MX services for your domain. For example, if your company's domain is `example.com` and your Internet service provider has set `example.net` up to provide secondary MX services to your domain:

```
example.com.          MX        10      example.com.
                      MX        20      example.net.
```

Only one host should be specified as the final recipient (add `Cw example.com` in `/etc/mail/sendmail.cf` on `example.com`).

When the sending `sendmail` is trying to deliver the mail it will try to connect to you (`example.com`) over the modem link. It will most likely time out because you are not online. The program **sendmail** will automatically deliver it to the secondary MX site, i.e. your Internet provider (`example.net`). The secondary MX site will then periodically try to connect to your host and deliver the mail to the primary MX host (`example.com`).

You might want to use something like this as a login script:

```
#!/bin/sh
# Put me in /usr/local/bin/pppmyisp
( sleep 60 ; /usr/sbin/sendmail -q ) &
/usr/sbin/ppp -direct pppmyisp
```

If you are going to create a separate login script for a user you could use `sendmail -qRexample.com` instead in the script above. This will force all mail in your queue for `example.com` to be processed immediately.

A further refinement of the situation is as follows:

Message stolen from the FreeBSD Internet service provider's mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp).

```
> 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,restrictqrun

Remove restrictqrun 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”.
```

4. Why do I keep getting `Relaying Denied` errors when sending mail from other hosts?

In default FreeBSD installations, **sendmail** is configured to only send mail from the host it is running on. For example, if a POP server is available, then users will be able to check mail from school, work, or other remote locations but they still will not be able to send outgoing emails from outside locations. Typically, a few moments after the attempt, an email will be sent from **MAILER-DAEMON** with a `5.7 Relaying Denied` error message.

There are several ways to get around this. The most straightforward solution is to put your ISP’s address in a relay-domains file at `/etc/mail/relay-domains`. A quick way to do this would be:

```
# echo "your.isp.example.com" > /etc/mail/relay-domains
```

After creating or editing this file you must restart **sendmail**. This works great if you are a server administrator and do not wish to send mail locally, or would like to use a point and click client/system on another machine or even another ISP. It is also very useful if you only have one or two email accounts set up. If there is a large number of addresses to add, you can simply open this file in your favorite text editor and then add the domains, one per line:

```
your.isp.example.com
other.isp.example.net
users-isp.example.org
www.example.org
```

Now any mail sent through your system, by any host in this list (provided the user has an account on your system), will succeed. This is a very nice way to allow users to send mail from your system remotely without allowing people to send SPAM through your system.

28.6 Advanced Topics

The following section covers more involved topics such as mail configuration and setting up mail for your entire domain.

28.6.1 Basic Configuration

Out of the box, you should be able to send email to external hosts as long as you have set up `/etc/resolv.conf` or are running your own name server. If you would like to have mail for your host delivered to the MTA (e.g., **sendmail**) on your own FreeBSD host, there are two methods:

- Run your own name server and have your own domain. For example, `FreeBSD.org`
- Get mail delivered directly to your host. This is done by delivering mail directly to the current DNS name for your machine. For example, `example.FreeBSD.org`.

Regardless of which of the above you choose, in order to have mail delivered directly to your host, it must have a permanent static IP address (not a dynamic address, as with most PPP dial-up configurations). If you are behind a firewall, it must pass SMTP traffic on to you. If you want to receive mail directly at your host, you need to be sure of either of two things:

- Make sure that the (lowest-numbered) MX record in your DNS points to your host's IP address.
- Make sure there is no MX entry in your DNS for your host.

Either of the above will allow you to receive mail directly at your host.

Try this:

```
# hostname
example.FreeBSD.org
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
```

If that is what you see, mail directly to <`yourlogin@example.FreeBSD.org`> should work without problems (assuming **sendmail** is running correctly on `example.FreeBSD.org`).

If instead you see something like this:

```
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
example.FreeBSD.org mail is handled (pri=10) by hub.FreeBSD.org
```

All mail sent to your host (`example.FreeBSD.org`) will end up being collected on `hub` under the same username instead of being sent directly to your host.

The above information is handled by your DNS server. The DNS record that carries mail routing information is the *M*ail e*X*change entry. If no MX record exists, mail will be delivered directly to the host by way of its IP address.

The MX entry for `freefall.FreeBSD.org` at one time looked like this:

```
freefall  MX 30 mail.crl.net
freefall  MX 40 agora.rdrop.com
freefall  MX 10 freefall.FreeBSD.org
freefall  MX 20 who.cdrom.com
```

As you can see, `freefall` had many MX entries. The lowest MX number is the host that receives mail directly if available; if it is not accessible for some reason, the others (sometimes called "backup MXes") accept messages temporarily, and pass it along when a lower-numbered host becomes available, eventually to the lowest-numbered host.

Alternate MX sites should have separate Internet connections from your own in order to be most useful. Your ISP or another friendly site should have no problem providing this service for you.

28.6.2 Mail for Your Domain

In order to set up a "mailhost" (a.k.a. mail server) you need to have any mail sent to various workstations directed to it. Basically, you want to "claim" any mail for any hostname in your domain (in this case `*.FreeBSD.org`) and divert it to your mail server so your users can receive their mail on the master mail server.

To make life easiest, a user account with the same *username* should exist on both machines. Use adduser(8) to do this.

The mailhost you will be using must be the designated mail exchanger for each workstation on the network. This is done in your DNS configuration like so:

```
example.FreeBSD.org A 204.216.27.XX  ; Workstation
   MX 10 hub.FreeBSD.org ; Mailhost
```

This will redirect mail for the workstation to the mailhost no matter where the A record points. The mail is sent to the MX host.

You cannot do this yourself unless you are running a DNS server. If you are not, or cannot run your own DNS server, talk to your ISP or whoever provides your DNS.

If you are doing virtual email hosting, the following information will come in handy. For this example, we will assume you have a customer with his own domain, in this case `customer1.org`, and you want all the mail for `customer1.org` sent to your mailhost, `mail.myhost.com`. The entry in your DNS should look like this:

```
customer1.org  MX 10 mail.myhost.com
```

You do *not* need an A record for `customer1.org` if you only want to handle email for that domain.

> **Note:** Be aware that pinging `customer1.org` will not work unless an A record exists for it.

The last thing that you must do is tell **sendmail** on your mailhost what domains and/or hostnames it should be accepting mail for. There are a few different ways this can be done. Either of the following will work:

- Add the hosts to your `/etc/mail/local-host-names` file if you are using the `FEATURE(use_cw_file)`. If you are using a version of **sendmail** earlier than 8.10, the file is `/etc/sendmail.cw`.
- Add a `Cwyour.host.com` line to your `/etc/sendmail.cf` or `/etc/mail/sendmail.cf` if you are using **sendmail** 8.10 or higher.

28.7 SMTP with UUCP

The **sendmail** configuration that ships with FreeBSD is designed for sites that connect directly to the Internet. Sites that wish to exchange their mail via UUCP must install another **sendmail** configuration file.

Tweaking `/etc/mail/sendmail.cf` manually is an advanced topic. **sendmail** version 8 generates config files via m4(1) preprocessing, where the actual configuration occurs on a higher abstraction level. The m4(1) configuration files can be found under `/usr/share/sendmail/cf`. The file `README` in the `cf` directory can serve as a basic introduction to m4(1) configuration.

The best way to support UUCP delivery is to use the `mailertable` feature. This creates a database that **sendmail** can use to make routing decisions.

First, you have to create your `.mc` file. The directory `/usr/share/sendmail/cf/cf` contains a few examples. Assuming you have named your file `foo.mc`, all you need to do in order to convert it into a valid `sendmail.cf` is:

```
# cd /etc/mail
# make foo.cf
# cp foo.cf /etc/mail/sendmail.cf
```

A typical `.mc` file might look like:

```
VERSIONID('Your version number') OSTYPE(bsd4.4)

FEATURE(accept_unresolvable_domains)
FEATURE(nocanonify)
FEATURE(mailertable, 'hash -o /etc/mail/mailertable')

define('UUCP_RELAY', your.uucp.relay)
define('UUCP_MAX_SIZE', 200000)
define('confDONT_PROBE_INTERFACES')

MAILER(local)
MAILER(smtp)
MAILER(uucp)

Cw    your.alias.host.name
Cw    youruucpnodename.UUCP
```

The lines containing `accept_unresolvable_domains`, `nocanonify`, and `confDONT_PROBE_INTERFACES` features will prevent any usage of the DNS during mail delivery. The `UUCP_RELAY` clause is needed to support UUCP delivery. Simply put an Internet hostname there that is able to handle .UUCP pseudo-domain addresses; most likely, you will enter the mail relay of your ISP there.

Once you have this, you need an `/etc/mail/mailertable` file. If you have only one link to the outside that is used for all your mails, the following file will suffice:

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
.                           uucp-dom:your.uucp.relay
```

A more complex example might look like this:

```
#
# 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:
```

The first three lines handle special cases where domain-addressed mail should not be sent out to the default route, but instead to some UUCP neighbor in order to “shortcut” the delivery path. The next line handles mail to the local Ethernet domain that can be delivered using SMTP. Finally, the UUCP neighbors are mentioned in the .UUCP pseudo-domain notation, to allow for a *`uucp-neighbor !recipient`* override of the default rules. The last line is always a single dot, matching everything else, with UUCP delivery to a UUCP neighbor that serves as your universal mail gateway to the world. All of the node names behind the `uucp-dom:` keyword must be valid UUCP neighbors, as you can verify using the command `uuname`.

As a reminder that this file needs to be converted into a DBM database file before use. The command line to accomplish this is best placed as a comment at the top of the `mailertable` file. You always have to execute this command each time you change your `mailertable` file.

Final hint: if you are uncertain whether some particular mail routing would work, remember the `-bt` option to **sendmail**. It starts **sendmail** in *address test mode*; simply enter `3,0`, followed by the address you wish to test for the mail routing. The last line tells you the used internal mail agent, the destination host this agent will be called with, and the (possibly translated) address. Leave this mode by typing **Ctrl**+**D**.

```
% sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 foo@example.com
canonify           input: foo @ example . com
...
parse            returns: $# uucp-dom $@ your.uucp.relay $: foo < @ example . com . >
> ^D
```

28.8 Setting Up to Send Only

There are many instances where you may only want to send mail through a relay. Some examples are:

- Your computer is a desktop machine, but you want to use programs such as send-pr(1). To do so, you should use your ISP’s mail relay.
- The computer is a server that does not handle mail locally, but needs to pass off all mail to a relay for processing.

Just about any MTA is capable of filling this particular niche. Unfortunately, it can be very difficult to properly configure a full-featured MTA just to handle offloading mail. Programs such as **sendmail** and **postfix** are largely overkill for this use.

Additionally, if you are using a typical Internet access service, your agreement may forbid you from running a “mail server”.

The easiest way to fulfill those needs is to install the `mail/ssmtp` port. Execute the following commands as `root`:

```
# cd /usr/ports/mail/ssmtp
# make install replace clean
```

Once installed, `mail/ssmtp` can be configured with a four-line file located at `/usr/local/etc/ssmtp/ssmtp.conf`:

```
root=yourrealemail@example.com
mailhub=mail.example.com
rewriteDomain=example.com
hostname=_HOSTNAME_
```

Make sure you use your real email address for `root`. Enter your ISP's outgoing mail relay in place of `mail.example.com` (some ISPs call this the "outgoing mail server" or "SMTP server").

Make sure you disable **sendmail**, including the outgoing mail service. See Section 28.4.2 for details.

`mail/ssmtp` has some other options available. See the example configuration file in `/usr/local/etc/ssmtp` or the manual page of **ssmtp** for some examples and more information.

Setting up **ssmtp** in this manner will allow any software on your computer that needs to send mail to function properly, while not violating your ISP's usage policy or allowing your computer to be hijacked for spamming.

28.9 Using Mail with a Dialup Connection

If you have a static IP address, you should not need to adjust anything from the defaults. Set your host name to your assigned Internet name and **sendmail** will do the rest.

If you have a dynamically assigned IP number and use a dialup PPP connection to the Internet, you will probably have a mailbox on your ISPs mail server. Let's assume your ISP's domain is `example.net`, and that your user name is `user`, you have called your machine `bsd.home`, and your ISP has told you that you may use `relay.example.net` as a mail relay.

In order to retrieve mail from your mailbox, you must install a retrieval agent. The **fetchmail** utility is a good choice as it supports many different protocols. This program is available as a package or from the Ports Collection (`mail/fetchmail`). Usually, your ISP will provide POP. If you are using user PPP, you can automatically fetch your mail when an Internet connection is established with the following entry in `/etc/ppp/ppp.linkup`:

```
MYADDR:
!bg su user -c fetchmail
```

If you are using **sendmail** (as shown below) to deliver mail to non-local accounts, you probably want to have **sendmail** process your mailqueue as soon as your Internet connection is established. To do this, put this command after the `fetchmail` command in `/etc/ppp/ppp.linkup`:

```
  !bg su user -c "sendmail -q"
```

Assume that you have an account for `user` on `bsd.home`. In the home directory of `user` on `bsd.home`, create a `.fetchmailrc` file:

```
poll example.net protocol pop3 fetchall pass MySecret
```

This file should not be readable by anyone except `user` as it contains the password `MySecret`.

In order to send mail with the correct `from:` header, you must tell **sendmail** to use <`user@example.net`> rather than <`user@bsd.home`>. You may also wish to tell **sendmail** to send all mail via `relay.example.net`, allowing quicker mail transmission.

The following `.mc` file should suffice:

```
VERSIONID('bsd.home.mc version 1.0')
OSTYPE(bsd4.4)dnl
FEATURE(nouucp)dnl
MAILER(local)dnl
MAILER(smtp)dnl
Cwlocalhost
Cwbsd.home
MASQUERADE_AS('example.net')dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(nocanonify)dnl
FEATURE(nodns)dnl
define('SMART_HOST', 'relay.example.net')
Dmbsd.home
define('confDOMAIN_NAME','bsd.home')dnl
define('confDELIVERY_MODE','deferred')dnl
```

Refer to the previous section for details of how to turn this `.mc` file into a `sendmail.cf` file. Also, do not forget to restart **sendmail** after updating `sendmail.cf`.

28.10 SMTP Authentication

Having SMTP Authentication in place on your mail server has a number of benefits. SMTP Authentication can add another layer of security to **sendmail**, and has the benefit of giving mobile users who switch hosts the ability to use the same mail server without the need to reconfigure their mail client settings each time.

1. Install `security/cyrus-sasl2` from the ports. You can find this port in `security/cyrus-sasl2`. The `security/cyrus-sasl2` port supports a number of compile-time options. For the SMTP Authentication method we will be using here, make sure that the `LOGIN` option is not disabled.
2. After installing `security/cyrus-sasl2`, edit `/usr/local/lib/sasl2/Sendmail.conf` (or create it if it does not exist) and add the following line:

   ```
   pwcheck_method: saslauthd
   ```
3. Next, install `security/cyrus-sasl2-saslauthd`, edit `/etc/rc.conf` to add the following line:

   ```
   saslauthd_enable="YES"
   ```

   and finally start the saslauthd daemon:

   ```
   # /usr/local/etc/rc.d/saslauthd start
   ```

   This daemon serves as a broker for **sendmail** to authenticate against your FreeBSD `passwd` database. This saves the trouble of creating a new set of usernames and passwords for each user that needs to use SMTP authentication, and keeps the login and mail password the same.
4. Now edit `/etc/make.conf` and add the following lines:

```
SENDMAIL_CFLAGS=-I/usr/local/include/sasl -DSASL
SENDMAIL_LDFLAGS=-L/usr/local/lib
SENDMAIL_LDADD=-lsasl2
```

These lines will give **sendmail** the proper configuration options for linking to `cyrus-sasl2` at compile time. Make sure that `cyrus-sasl2` has been installed before recompiling **sendmail**.

5. Recompile **sendmail** by executing the following commands:

```
# 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
```

The compile of **sendmail** should not have any problems if `/usr/src` has not been changed extensively and the shared libraries it needs are available.

6. After **sendmail** has been compiled and reinstalled, edit your `/etc/mail/freebsd.mc` file (or whichever file you use as your `.mc` file. Many administrators choose to use the output from hostname(1) as the `.mc` file for uniqueness). Add these lines to it:

```
dnl set SASL options
TRUST_AUTH_MECH(‘GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN’)dnl
define(‘confAUTH_MECHANISMS’, ‘GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN’)dnl
```

These options configure the different methods available to **sendmail** for authenticating users. If you would like to use a method other than **pwcheck**, please see the included documentation.

7. Finally, run make(1) while in `/etc/mail`. That will run your new `.mc` file and create a `.cf` file named `freebsd.cf` (or whatever name you have used for your `.mc` file). Then use the command `make install restart`, which will copy the file to `sendmail.cf`, and will properly restart **sendmail**. For more information about this process, you should refer to `/etc/mail/Makefile`.

If all has gone correctly, you should be able to enter your login information into the mail client and send a test message. For further investigation, set the `LogLevel` of **sendmail** to 13 and watch `/var/log/maillog` for any errors.

For more information, please see the **sendmail** page regarding SMTP authentication (http://www.sendmail.org/~ca/email/auth.html).

28.11 Mail User Agents

A Mail User Agent (MUA) is an application that is used to send and receive email. Furthermore, as email “evolves” and becomes more complex, MUA’s are becoming increasingly powerful in the way they interact with email; this gives users increased functionality and flexibility. FreeBSD contains support for numerous mail user agents, all of which can be easily installed using the FreeBSD Ports Collection. Users may choose between graphical email clients such as **evolution** or **balsa**, console based clients such as **mutt**, **alpine** or `mail`, or the web interfaces used by some large organizations.

28.11.1 mail

mail(1) is the default Mail User Agent (MUA) in FreeBSD. It is a console based MUA that offers all the basic functionality required to send and receive text-based email, though it is limited in interaction abilities with attachments and can only support local mailboxes.

Although `mail` does not natively support interaction with POP or IMAP servers, these mailboxes may be downloaded to a local `mbox` file using an application such as **fetchmail**, which will be discussed later in this chapter (Section 28.12).

In order to send and receive email, simply invoke the `mail` command as per the following example:

```
% mail
```

The contents of the user mailbox in `/var/mail` are automatically read by the `mail` utility. Should the mailbox be empty, the utility exits with a message indicating that no mails could be found. Once the mailbox has been read, the application interface is started, and a list of messages will be displayed. Messages are automatically numbered, as can be seen in the following example:

```
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   "test"
 N  2 root@localhost        Mon Mar  8 14:05  14/509   "user account"
 N  3 root@localhost        Mon Mar  8 14:05  14/509   "sample"
```

Messages can now be read by using the **t** `mail` command, suffixed by the message number that should be displayed. In this example, we will read the first email:

```
& 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: test
Date: Mon,  8 Mar 2004 14:05:52 +0200 (SAST)
From: root@localhost (Charlie Root)

This is a test message, please reply if you receive it.
```

As can be seen in the example above, the **t** key will cause the message to be displayed with full headers. To display the list of messages again, the **h** key should be used.

If the email requires a response, you may use `mail` to reply, by using either the **R** or **r** `mail` keys. The **R** key instructs `mail` to reply only to the sender of the email, while **r** replies not only to the sender, but also to other recipients of the message. You may also suffix these commands with the mail number which you would like make a reply to. Once this has been done, the response should be entered, and the end of the message should be marked by a single **.** on a new line. An example can be seen below:

```
& R 1
To: root@localhost
Subject: Re: test

Thank you, I did get your email.
```

```
.
EOT
```

In order to send new email, the **m** key should be used, followed by the recipient email address. Multiple recipients may also be specified by separating each address with the **,** delimiter. The subject of the message may then be entered, followed by the message contents. The end of the message should be specified by putting a single **.** on a new line.

```
& mail root@localhost
Subject: I mastered mail

Now I can send and receive email using mail ... :)
.
EOT
```

While inside the `mail` utility, the **?** command may be used to display help at any time, the mail(1) manual page should also be consulted for more help with `mail`.

> **Note:** As previously mentioned, the mail(1) command was not originally designed to handle attachments, and thus deals with them very poorly. Newer MUAs such as **mutt** handle attachments in a much more intelligent way. But should you still wish to use the `mail` command, the `converters/mpack` port may be of considerable use.

28.11.2 mutt

mutt is a small yet very powerful Mail User Agent, with excellent features, just some of which include:

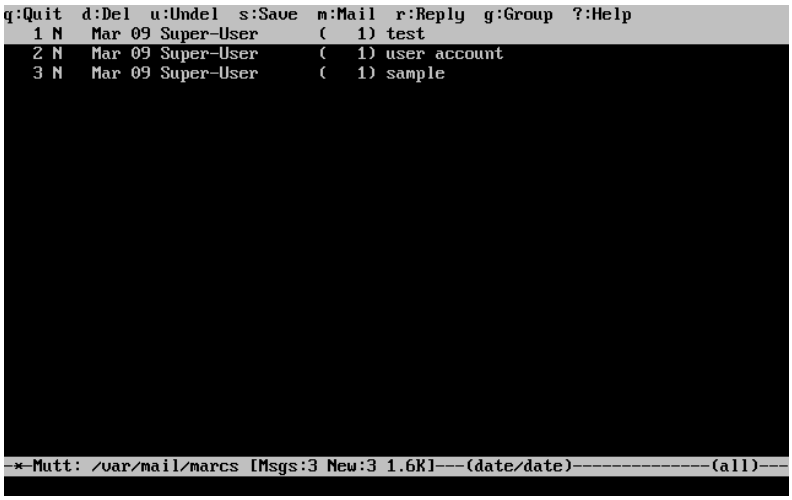
- The ability to thread messages;
- PGP support for digital signing and encryption of email;
- MIME Support;
- Maildir Support;
- Highly customizable.

All of these features help to make **mutt** one of the most advanced mail user agents available. See http://www.mutt.org for more information on **mutt**.

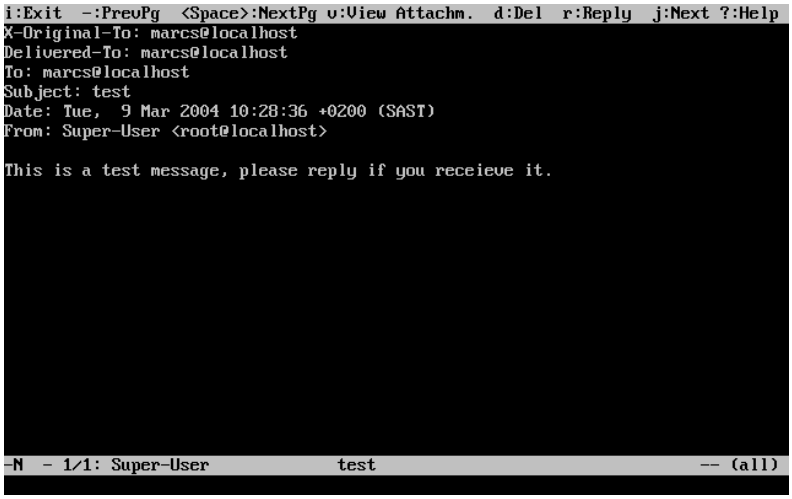
The stable version of **mutt** may be installed using the `mail/mutt` port, while the current development version may be installed via the `mail/mutt-devel` port. After the port has been installed, **mutt** can be started by issuing the following command:

```
% mutt
```

mutt will automatically read the contents of the user mailbox in `/var/mail` and display the contents if applicable. If no mails are found in the user mailbox, then **mutt** will wait for commands from the user. The example below shows **mutt** displaying a list of messages:



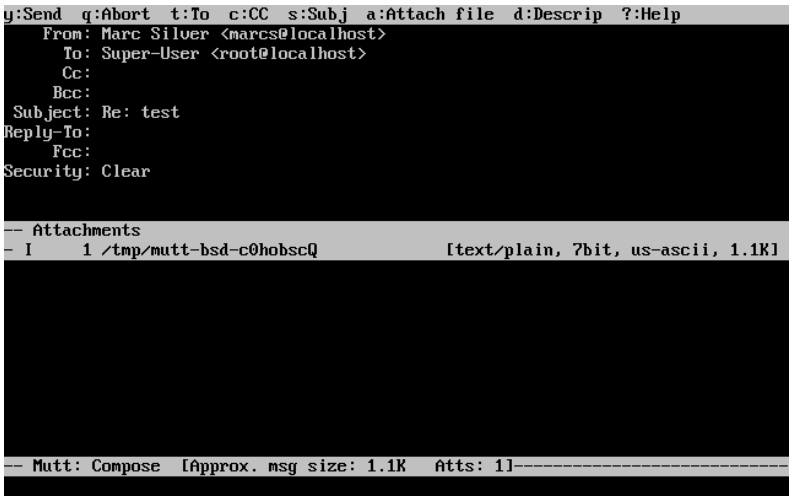
In order to read an email, simply select it using the cursor keys, and press the **Enter** key. An example of **mutt** displaying email can be seen below:



As with the mail(1) command, **mutt** allows users to reply only to the sender of the message as well as to all recipients. To reply only to the sender of the email, use the **r** keyboard shortcut. To send a group reply, which will be sent to the original sender as well as all the message recipients, use the **g** shortcut.

> **Note: mutt** makes use of the vi(1) command as an editor for creating and replying to emails. This may be customized by the user by creating or editing their own `.muttrc` file in their home directory and setting the `editor` variable or by setting the `EDITOR` environment variable. See http://www.mutt.org/ for more information about configuring **mutt**.

In order to compose a new mail message, press **m**. After a valid subject has been given, **mutt** will start vi(1) and the mail can be written. Once the contents of the mail are complete, save and quit from `vi` and **mutt** will resume, displaying a summary screen of the mail that is to be delivered. In order to send the mail, press **y**. An example of the summary screen can be seen below:



mutt also contains extensive help, which can be accessed from most of the menus by pressing the **?** key. The top line also displays the keyboard shortcuts where appropriate.

28.11.3 alpine

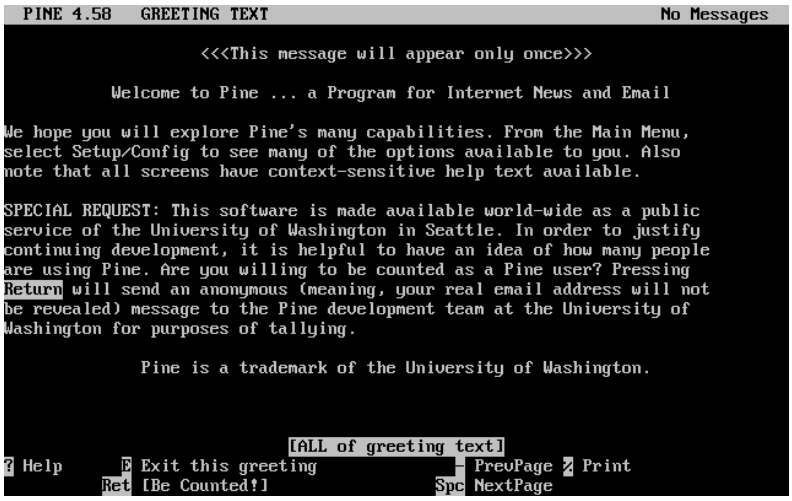
alpine is aimed at a beginner user, but also includes some advanced features.

> **Warning:** The **alpine** software has had several remote vulnerabilities discovered in the past, which allowed remote attackers to execute arbitrary code as users on the local system, by the action of sending a specially-prepared email. All such *known* problems have been fixed, but the **alpine** code is written in a very insecure style and the FreeBSD Security Officer believes there are likely to be other undiscovered vulnerabilities. You install **alpine** at your own risk.

The current version of **alpine** may be installed using the `mail/alpine` port. Once the port has installed, **alpine** can be started by issuing the following command:

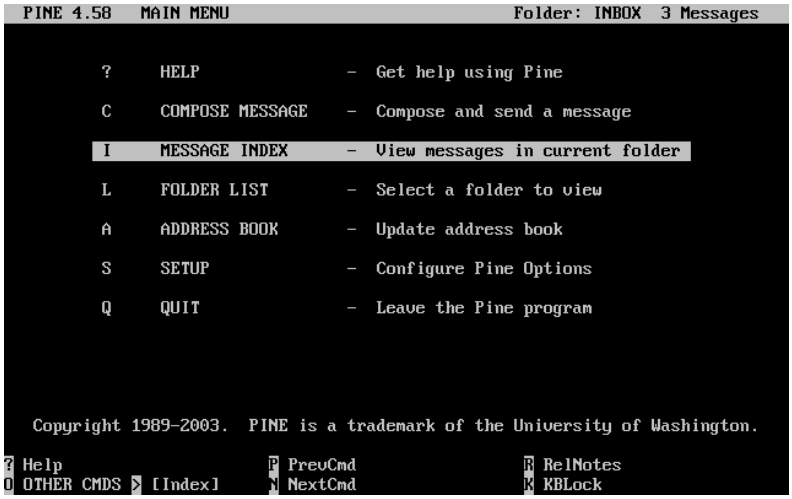
```
% alpine
```

The first time that **alpine** is run it displays a greeting page with a brief introduction, as well as a request from the **alpine** development team to send an anonymous email message allowing them to judge how many users are using their client. To send this anonymous message, press **Enter**, or alternatively press **E** to exit the greeting without sending an anonymous message. An example of the greeting page can be seen below:

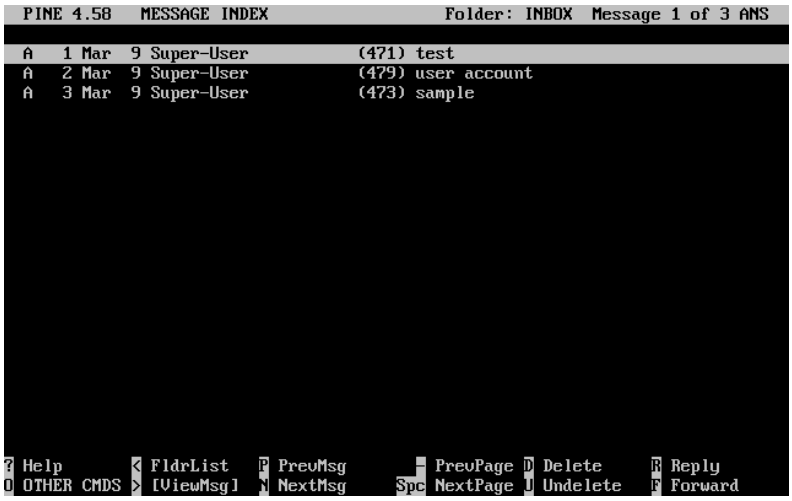


Users are then presented with the main menu, which can be easily navigated using the cursor keys. This main menu provides shortcuts for the composing new mails, browsing of mail directories, and even the administration of address book entries. Below the main menu, relevant keyboard shortcuts to perform functions specific to the task at hand are shown.

The default directory opened by **alpine** is the `inbox`. To view the message index, press **I**, or select the MESSAGE INDEX option as seen below:

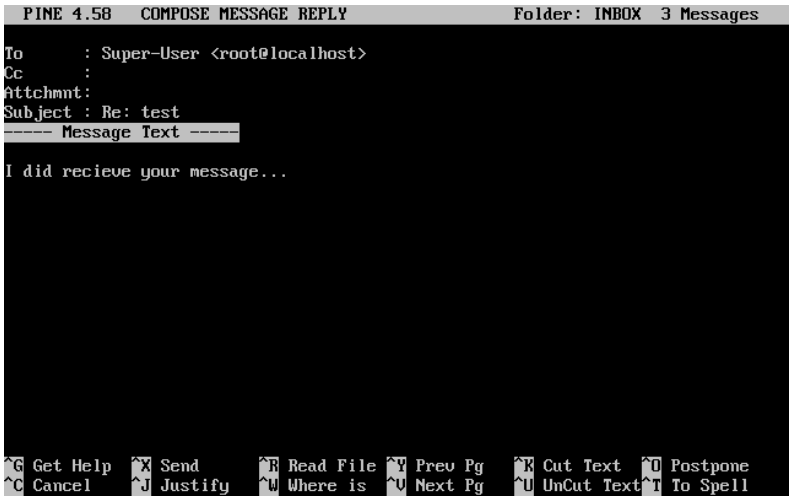


The message index shows messages in the current directory, and can be navigated by using the cursor keys. Highlighted messages can be read by pressing the **Enter** key.



In the screenshot below, a sample message is displayed by **alpine**. Keyboard shortcuts are displayed as a reference at the bottom of the screen. An example of one of these shortcuts is the **r** key, which tells the MUA to reply to the current message being displayed.

Replying to an email in **alpine** is done using the **pico** editor, which is installed by default with **alpine**. The **pico** utility makes it easy to navigate around the message and is slightly more forgiving on novice users than vi(1) or mail(1). Once the reply is complete, the message can be sent by pressing **Ctrl**+**X**. The **alpine** application will ask for confirmation.



The **alpine** application can be customized using the SETUP option from the main menu. Consult http://www.washington.edu/alpine/ for more information.

28.12 Using fetchmail

fetchmail is a full-featured IMAP and POP client which allows users to automatically download mail from remote IMAP and POP servers and save it into local mailboxes; there it can be accessed more easily. **fetchmail** can be installed using the `mail/fetchmail` port, and offers various features, some of which include:

- Support of POP3, APOP, KPOP, IMAP, ETRN and ODMR protocols.
- Ability to forward mail using SMTP, which allows filtering, forwarding, and aliasing to function normally.
- May be run in daemon mode to check periodically for new messages.
- Can retrieve multiple mailboxes and forward them based on configuration, to different local users.

While it is outside the scope of this document to explain all of **fetchmail**'s features, some basic features will be explained. The **fetchmail** utility requires a configuration file known as `.fetchmailrc`, in order to run correctly. This file includes server information as well as login credentials. Due to the sensitive nature of the contents of this file, it is advisable to make it readable only by the owner, with the following command:

```
% chmod 600 .fetchmailrc
```

The following `.fetchmailrc` serves as an example for downloading a single user mailbox using POP. It tells **fetchmail** to connect to `example.com` using a username of `joesoap` and a password of `XXX`. This example assumes that the user `joesoap` is also a user on the local system.

```
poll example.com protocol pop3 username "joesoap" password "XXX"
```

The next example connects to multiple POP and IMAP servers and redirects to different local usernames where applicable:

```
poll example.com proto pop3:
user "joesoap", with password "XXX", is "jsoap" here;
```

```
user "andrea", with password "XXXX";
poll example2.net proto imap:
user "john", with password "XXXXX", is "myth" here;
```

The **fetchmail** utility can be run in daemon mode by running it with the `-d` flag, followed by the interval (in seconds) that **fetchmail** should poll servers listed in the `.fetchmailrc` file. The following example would cause **fetchmail** to poll every 600 seconds:

```
% fetchmail -d 600
```

More information on **fetchmail** can be found at http://fetchmail.berlios.de/.

28.13 Using procmail

The **procmail** utility is an incredibly powerful application used to filter incoming mail. It allows users to define "rules" which can be matched to incoming mails to perform specific functions or to reroute mail to alternative mailboxes and/or email addresses. **procmail** can be installed using the `mail/procmail` port. Once installed, it can be directly integrated into most MTAs; consult your MTA documentation for more information. Alternatively, **procmail** can be integrated by adding the following line to a `.forward` in the home directory of the user utilizing **procmail** features:

```
"|exec /usr/local/bin/procmail || exit 75"
```

The following section will display some basic **procmail** rules, as well as brief descriptions on what they do. These rules, and others must be inserted into a `.procmailrc` file, which must reside in the user's home directory.

The majority of these rules can also be found in the procmailex(5) manual page.

Forward all mail from <`user@example.com`> to an external address of <`goodmail@example2.com`>:

```
:0
* ^From.*user@example.com
! goodmail@example2.com
```

Forward all mails shorter than 1000 bytes to an external address of <`goodmail@example2.com`>:

```
:0
* < 1000
! goodmail@example2.com
```

Send all mail sent to <`alternate@example.com`> into a mailbox called `alternate`:

```
:0
* ^TOalternate@example.com
alternate
```

Send all mail with a subject of "Spam" to `/dev/null`:

```
:0
^Subject:.*Spam
/dev/null
```

A useful recipe that parses incoming `FreeBSD.org` mailing lists and places each list in its own mailbox:

```
:0
* ^Sender:.owner-freebsd-\/[^@]+@FreeBSD.ORG
{
 LISTNAME=${MATCH}
 :0
 * LISTNAME??^\/[^@]+
 FreeBSD-${MATCH}
}
```

Chapter 29
Network Servers

29.1 Synopsis

This chapter will cover some of the more frequently used network services on UNIX systems. We will cover how to install, configure, test, and maintain many different types of network services. Example configuration files are included throughout this chapter for you to benefit from.

After reading this chapter, you will know:

- How to manage the **inetd** daemon.
- How to set up a network file system.
- How to set up a network information server for sharing user accounts.
- How to set up automatic network settings using DHCP.
- How to set up a domain name server.
- How to set up the **Apache** HTTP Server.
- How to set up a File Transfer Protocol (FTP) Server.
- How to set up a file and print server for Windows clients using **Samba**.
- How to synchronize the time and date, and set up a time server, with the NTP protocol.
- How to configure the standard logging daemon, `syslogd`, to accept logs from remote hosts.

Before reading this chapter, you should:

- Understand the basics of the `/etc/rc` scripts.
- Be familiar with basic network terminology.
- Know how to install additional third-party software (Chapter 4).

29.2 The inetd "Super-Server"

29.2.1 Overview

inetd(8) is sometimes referred to as the "Internet Super-Server" because it manages connections for several services. When a connection is received by **inetd**, it determines which program the connection is destined for, spawns the particular process and delegates the socket to it (the program is invoked with the service socket as its standard input, output and error descriptors). Running **inetd** for servers that are not heavily used can reduce the overall system load, when compared to running each daemon individually in stand-alone mode.

Primarily, **inetd** is used to spawn other daemons, but several trivial protocols are handled directly, such as **chargen**, **auth**, and **daytime**.

This section will cover the basics in configuring **inetd** through its command-line options and its configuration file, `/etc/inetd.conf`.

29.2.2 Settings

inetd is initialized through the rc(8) system. The `inetd_enable` option is set to `NO` by default, but may be turned on by **sysinstall** during installation, depending on the configuration chosen by the user. Placing:

```
inetd_enable="YES"
```

or

```
inetd_enable="NO"
```

into `/etc/rc.conf` will enable or disable **inetd** starting at boot time. The command:

```
# /etc/rc.d/inetd rcvar
```

can be run to display the current effective setting.

Additionally, different command-line options can be passed to **inetd** via the `inetd_flags` option.

29.2.3 Command-Line Options

Like most server daemons, **inetd** has a number of options that it can be passed in order to modify its behaviour. The full list of options reads:

```
inetd [-d] [-l] [-w] [-W] [-c maximum] [-C rate] [-a address | hostname] [-p filename]
[-R rate] [-s maximum] [configuration file]
```

Options can be passed to **inetd** using the `inetd_flags` option in `/etc/rc.conf`. By default, `inetd_flags` is set to `-wW -C 60`, which turns on TCP wrapping for **inetd**'s services, and prevents any single IP address from requesting any service more than 60 times in any given minute.

Although we mention rate-limiting options below, novice users may be pleased to note that these parameters usually do not need to be modified. These options may be useful should you find that you are receiving an excessive amount of connections. A full list of options can be found in the inetd(8) manual.

-c maximum

> Specify the default maximum number of simultaneous invocations of each service; the default is unlimited. May be overridden on a per-service basis with the `max-child` parameter.

-C rate

> Specify the default maximum number of times a service can be invoked from a single IP address in one minute; the default is unlimited. May be overridden on a per-service basis with the `max-connections-per-ip-per-minute` parameter.

-R rate

Specify the maximum number of times a service can be invoked in one minute; the default is 256. A rate of 0 allows an unlimited number of invocations.

-s maximum

Specify the maximum number of times a service can be invoked from a single IP address at any one time; the default is unlimited. May be overridden on a per-service basis with the `max-child-per-ip` parameter.

29.2.4 `inetd.conf`

Configuration of **inetd** is done via the file `/etc/inetd.conf`.

When a modification is made to `/etc/inetd.conf`, **inetd** can be forced to re-read its configuration file by running the command:

Example 29-1. Reloading the inetd configuration file

```
# /etc/rc.d/inetd reload
```

Each line of the configuration file specifies an individual daemon. Comments in the file are preceded by a "#". The format of each entry in `/etc/inetd.conf` is as follows:

```
service-name
socket-type
protocol
{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]
user[:group][/login-class]
server-program
server-program-arguments
```

An example entry for the ftpd(8) daemon using IPv4 might read:

```
ftp     stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -l
```

service-name

This is the service name of the particular daemon. It must correspond to a service listed in `/etc/services`. This determines which port **inetd** must listen to. If a new service is being created, it must be placed in `/etc/services` first.

socket-type

Either `stream`, `dgram`, `raw`, or `seqpacket`. `stream` must be used for connection-based, TCP daemons, while `dgram` is used for daemons utilizing the UDP transport protocol.

protocol

One of the following:

Protocol	Explanation
tcp, tcp4	TCP IPv4

Protocol	Explanation
udp, udp4	UDP IPv4
tcp6	TCP IPv6
udp6	UDP IPv6
tcp46	Both TCP IPv4 and v6
udp46	Both UDP IPv4 and v6

{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]

`wait|nowait` indicates whether the daemon invoked from **inetd** is able to handle its own socket or not. `dgram` socket types must use the `wait` option, while stream socket daemons, which are usually multi-threaded, should use `nowait`. `wait` usually hands off multiple sockets to a single daemon, while `nowait` spawns a child daemon for each new socket.

The maximum number of child daemons **inetd** may spawn can be set using the `max-child` option. If a limit of ten instances of a particular daemon is needed, a `/10` would be placed after `nowait`. Specifying `/0` allows an unlimited number of children

In addition to `max-child`, two other options which limit the maximum connections from a single place to a particular daemon can be enabled. `max-connections-per-ip-per-minute` limits the number of connections from any particular IP address per minutes, e.g. a value of ten would limit any particular IP address connecting to a particular service to ten attempts per minute. `max-child-per-ip` limits the number of children that can be started on behalf on any single IP address at any moment. These options are useful to prevent intentional or unintentional excessive resource consumption and Denial of Service (DoS) attacks to a machine.

In this field, either of `wait` or `nowait` is mandatory. `max-child`, `max-connections-per-ip-per-minute` and `max-child-per-ip` are optional.

A stream-type multi-threaded daemon without any `max-child`, `max-connections-per-ip-per-minute` or `max-child-per-ip` limits would simply be: `nowait`.

The same daemon with a maximum limit of ten daemons would read: `nowait/10`.

The same setup with a limit of twenty connections per IP address per minute and a maximum total limit of ten child daemons would read: `nowait/10/20`.

These options are utilized by the default settings of the fingerd(8) daemon, as seen here:

```
finger stream  tcp     nowait/3/10 nobody /usr/libexec/fingerd fingerd -s
```

Finally, an example of this field with a maximum of 100 children in total, with a maximum of 5 for any one IP address would read: `nowait/100/0/5`.

user

This is the username that the particular daemon should run as. Most commonly, daemons run as the `root` user. For security purposes, it is common to find some servers running as the `daemon` user, or the least privileged `nobody` user.

server-program

The full path of the daemon to be executed when a connection is received. If the daemon is a service provided by **inetd** internally, then `internal` should be used.

server-program-arguments

This works in conjunction with `server-program` by specifying the arguments, starting with `argv[0]`, passed to the daemon on invocation. If `mydaemon -d` is the command line, `mydaemon -d` would be the value of `server-program-arguments`. Again, if the daemon is an internal service, use `internal` here.

29.2.5 Security

Depending on the choices made at install time, many of **inetd**'s services may be enabled by default. If there is no apparent need for a particular daemon, consider disabling it. Place a "#" in front of the daemon in question in `/etc/inetd.conf`, and then reload the inetd configuration. Some daemons, such as **fingerd**, may not be desired at all because they provide information that may be useful to an attacker.

Some daemons are not security-conscious and have long, or non-existent, timeouts for connection attempts. This allows an attacker to slowly send connections to a particular daemon, thus saturating available resources. It may be a good idea to place `max-connections-per-ip-per-minute`, `max-child` or `max-child-per-ip` limitations on certain daemons if you find that you have too many connections.

By default, TCP wrapping is turned on. Consult the hosts_access(5) manual page for more information on placing TCP restrictions on various **inetd** invoked daemons.

29.2.6 Miscellaneous

daytime, **time**, **echo**, **discard**, **chargen**, and **auth** are all internally provided services of **inetd**.

The **auth** service provides identity network services, and is configurable to a certain degree, whilst the others are simply on or off.

Consult the inetd(8) manual page for more in-depth information.

29.3 Network File System (NFS)

Among the many different file systems that FreeBSD supports is the Network File System, also known as NFS. NFS allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files.

Some of the most notable benefits that NFS can provide are:

- Local workstations use less disk space because commonly used data can be stored on a single machine and still remain accessible to others over the network.
- There is no need for users to have separate home directories on every network machine. Home directories could be set up on the NFS server and made available throughout the network.
- Storage devices such as floppy disks, CDROM drives, and Zip® drives can be used by other machines on the network. This may reduce the number of removable media drives throughout the network.

29.3.1 How NFS Works

NFS consists of at least two main parts: a server and one or more clients. The client remotely accesses the data that is stored on the server machine. In order for this to function properly a few processes have to be configured and running.

The server has to be running the following daemons:

Daemon	Description
nfsd	The NFS daemon which services requests from the NFS clients.
mountd	The NFS mount daemon which carries out the requests that nfsd(8) passes on to it.
rpcbind	This daemon allows NFS clients to discover which port the NFS server is using.

The client can also run a daemon, known as **nfsiod**. The **nfsiod** daemon services the requests from the NFS server. This is optional, and improves performance, but is not required for normal and correct operation. See the nfsiod(8) manual page for more information.

29.3.2 Configuring NFS

NFS configuration is a relatively straightforward process. The processes that need to be running can all start at boot time with a few modifications to your `/etc/rc.conf` file.

On the NFS server, make sure that the following options are configured in the `/etc/rc.conf` file:

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

mountd runs automatically whenever the NFS server is enabled.

On the client, make sure this option is present in `/etc/rc.conf`:

```
nfs_client_enable="YES"
```

The `/etc/exports` file specifies which file systems NFS should export (sometimes referred to as “share”). Each line in `/etc/exports` specifies a file system to be exported and which machines have access to that file system. Along with what machines have access to that file system, access options may also be specified. There are many such options that can be used in this file but only a few will be mentioned here. You can easily discover other options by reading over the exports(5) manual page.

Here are a few example `/etc/exports` entries:

The following examples give an idea of how to export file systems, although the settings may be different depending on your environment and network configuration. For instance, to export the `/cdrom` directory to three example machines that have the same domain name as the server (hence the lack of a domain name for each) or have entries in your `/etc/hosts` file. The `-ro` flag makes the exported file system read-only. With this flag, the remote system will not be able to write any changes to the exported file system.

```
/cdrom -ro host1 host2 host3
```

The following line exports `/home` to three hosts by IP address. This is a useful setup if you have a private network without a DNS server configured. Optionally the `/etc/hosts` file could be configured for internal hostnames; please review hosts(5) for more information. The `-alldirs` flag allows the subdirectories to be mount points. In

other words, it will not mount the subdirectories but permit the client to mount only the directories that are required or needed.

```
/home  -alldirs  10.0.0.2 10.0.0.3 10.0.0.4
```

The following line exports `/a` so that two clients from different domains may access the file system. The `-maproot=root` flag allows the `root` user on the remote system to write data on the exported file system as `root`. If the `-maproot=root` flag is not specified, then even if a user has `root` access on the remote system, he will not be able to modify files on the exported file system.

```
/a  -maproot=root  host.example.com box.example.org
```

In order for a client to access an exported file system, the client must have permission to do so. Make sure the client is listed in your `/etc/exports` file.

In `/etc/exports`, each line represents the export information for one file system to one host. A remote host can only be specified once per file system, and may only have one default entry. For example, assume that `/usr` is a single file system. The following `/etc/exports` would be invalid:

```
# Invalid when /usr is one file system
/usr/src   client
/usr/ports client
```

One file system, `/usr`, has two lines specifying exports to the same host, `client`. The correct format for this situation is:

```
/usr/src /usr/ports  client
```

The properties of one file system exported to a given host must all occur on one line. Lines without a client specified are treated as a single host. This limits how you can export file systems, but for most people this is not an issue.

The following is an example of a valid export list, where `/usr` and `/exports` are local file systems:

```
# Export src and ports to client01 and client02, but only
# client01 has root privileges on it
/usr/src /usr/ports -maproot=root    client01
/usr/src /usr/ports               client02
# The client machines have root and can mount anywhere
# on /exports. Anyone in the world can mount /exports/obj read-only
/exports -alldirs -maproot=root      client01 client02
/exports/obj -ro
```

The **mountd** daemon must be forced to recheck the `/etc/exports` file whenever it has been modified, so the changes can take effect. This can be accomplished either by sending a HUP signal to the running daemon:

```
# kill -HUP ‘cat /var/run/mountd.pid‘
```

or by invoking the `mountd` rc(8) script with the appropriate parameter:

```
# /etc/rc.d/mountd onereload
```

Please refer to Section 11.7 for more information about using rc scripts.

Alternatively, a reboot will make FreeBSD set everything up properly. A reboot is not necessary though. Executing the following commands as `root` should start everything up.

On the NFS server:

```
# rpcbind
# nfsd -u -t -n 4
# mountd -r
```

On the NFS client:

```
# nfsiod -n 4
```

Now everything should be ready to actually mount a remote file system. In these examples the server's name will be `server` and the client's name will be `client`. If you only want to temporarily mount a remote file system or would rather test the configuration, just execute a command like this as `root` on the client:

```
# mount server:/home /mnt
```

This will mount the `/home` directory on the server at `/mnt` on the client. If everything is set up correctly you should be able to enter `/mnt` on the client and see all the files that are on the server.

If you want to automatically mount a remote file system each time the computer boots, add the file system to the `/etc/fstab` file. Here is an example:

```
server:/home /mnt nfs rw 0 0
```

The fstab(5) manual page lists all the available options.

29.3.3 Locking

Some applications (e.g. **mutt**) require file locking to operate correctly. In the case of NFS, **rpc.lockd** can be used for file locking. To enable it, add the following to the `/etc/rc.conf` file on both client and server (it is assumed that the NFS client and server are configured already):

```
rpc_lockd_enable="YES"
rpc_statd_enable="YES"
```

Start the application by using:

```
# /etc/rc.d/lockd start
# /etc/rc.d/statd start
```

If real locking between the NFS clients and NFS server is not required, it is possible to let the NFS client do locking locally by passing `-L` to mount_nfs(8). Refer to the mount_nfs(8) manual page for further details.

29.3.4 Practical Uses

NFS has many practical uses. Some of the more common ones are listed below:

- Set several machines to share a CDROM or other media among them. This is cheaper and often a more convenient method to install software on multiple machines.
- On large networks, it might be more convenient to configure a central NFS server in which to store all the user home directories. These home directories can then be exported to the network so that users would always have the same home directory, regardless of which workstation they log in to.
- Several machines could have a common `/usr/ports/distfiles` directory. That way, when you need to install a port on several machines, you can quickly access the source without downloading it on each machine.

29.3.5 Automatic Mounts with amd

amd(8) (the automatic mounter daemon) automatically mounts a remote file system whenever a file or directory within that file system is accessed. Filesystems that are inactive for a period of time will also be automatically unmounted by **amd**. Using **amd** provides a simple alternative to permanent mounts, as permanent mounts are usually listed in `/etc/fstab`.

amd operates by attaching itself as an NFS server to the `/host` and `/net` directories. When a file is accessed within one of these directories, **amd** looks up the corresponding remote mount and automatically mounts it. `/net` is used to mount an exported file system from an IP address, while `/host` is used to mount an export from a remote hostname.

An access to a file within `/host/foobar/usr` would tell **amd** to attempt to mount the `/usr` export on the host `foobar`.

Example 29-2. Mounting an Export with amd

You can view the available mounts of a remote host with the `showmount` command. For example, to view the mounts of a host named `foobar`, you can use:

```
% showmount -e foobar
Exports list on foobar:
/usr                                10.10.10.0
/a                                  10.10.10.0
% cd /host/foobar/usr
```

As seen in the example, the `showmount` shows `/usr` as an export. When changing directories to `/host/foobar/usr`, **amd** attempts to resolve the hostname `foobar` and automatically mount the desired export.

amd can be started by the startup scripts by placing the following lines in `/etc/rc.conf`:

```
amd_enable="YES"
```

Additionally, custom flags can be passed to **amd** from the `amd_flags` option. By default, `amd_flags` is set to:

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

The `/etc/amd.map` file defines the default options that exports are mounted with. The `/etc/amd.conf` file defines some of the more advanced features of **amd**.

Consult the amd(8) and amd.conf(5) manual pages for more information.

29.3.6 Problems Integrating with Other Systems

Certain Ethernet adapters for ISA PC systems have limitations which can lead to serious network problems, particularly with NFS. This difficulty is not specific to FreeBSD, but FreeBSD systems are affected by it.

The problem nearly always occurs when (FreeBSD) PC systems are networked with high-performance workstations, such as those made by Silicon Graphics, Inc., and Sun Microsystems, Inc. The NFS mount will work fine, and some operations may succeed, but suddenly the server will seem to become unresponsive to the client, even though requests to and from other systems continue to be processed. This happens to the client system, whether the client is the FreeBSD system or the workstation. On many systems, there is no way to shut down the client gracefully once this problem has manifested itself. The only solution is often to reset the client, because the NFS situation cannot be resolved.

Though the "correct" solution is to get a higher performance and capacity Ethernet adapter for the FreeBSD system, there is a simple workaround that will allow satisfactory operation. If the FreeBSD system is the *server*, include the option `-w=1024` on the mount from the client. If the FreeBSD system is the *client*, then mount the NFS file system with the option `-r=1024`. These options may be specified using the fourth field of the `fstab` entry on the client for automatic mounts, or by using the `-o` parameter of the mount(8) command for manual mounts.

It should be noted that there is a different problem, sometimes mistaken for this one, when the NFS servers and clients are on different networks. If that is the case, make *certain* that your routers are routing the necessary UDP information, or you will not get anywhere, no matter what else you are doing.

In the following examples, `fastws` is the host (interface) name of a high-performance workstation, and `freebox` is the host (interface) name of a FreeBSD system with a lower-performance Ethernet adapter. Also, `/sharedfs` will be the exported NFS file system (see exports(5)), and `/project` will be the mount point on the client for the exported file system. In all cases, note that additional options, such as `hard` or `soft` and `bg` may be desirable in your application.

Examples for the FreeBSD system (`freebox`) as the client in `/etc/fstab` on `freebox`:

```
fastws:/sharedfs /project nfs rw,-r=1024 0 0
```

As a manual mount command on `freebox`:

```
# mount -t nfs -o -r=1024 fastws:/sharedfs /project
```

Examples for the FreeBSD system as the server in `/etc/fstab` on `fastws`:

```
freebox:/sharedfs /project nfs rw,-w=1024 0 0
```

As a manual mount command on `fastws`:

```
# mount -t nfs -o -w=1024 freebox:/sharedfs /project
```

Nearly any 16-bit Ethernet adapter will allow operation without the above restrictions on the read or write size.

For anyone who cares, here is what happens when the failure occurs, which also explains why it is unrecoverable. NFS typically works with a "block" size of 8 K (though it may do fragments of smaller sizes). Since the maximum Ethernet packet is around 1500 bytes, the NFS "block" gets split into multiple Ethernet packets, even though it is still a single unit to the upper-level code, and must be received, assembled, and *acknowledged* as a unit. The high-performance workstations can pump out the packets which comprise the NFS unit one right after the other, just as close together as the standard allows. On the smaller, lower capacity cards, the later packets overrun the earlier packets of the same unit before they can be transferred to the host and the unit as a whole cannot be reconstructed or

acknowledged. As a result, the workstation will time out and try again, but it will try again with the entire 8 K unit, and the process will be repeated, ad infinitum.

By keeping the unit size below the Ethernet packet size limitation, we ensure that any complete Ethernet packet received can be acknowledged individually, avoiding the deadlock situation.

Overruns may still occur when a high-performance workstations is slamming data out to a PC system, but with the better cards, such overruns are not guaranteed on NFS "units". When an overrun occurs, the units affected will be retransmitted, and there will be a fair chance that they will be received, assembled, and acknowledged.

29.4 Network Information System (NIS/YP)

29.4.1 What Is It?

NIS, which stands for Network Information Services, was developed by Sun Microsystems to centralize administration of UNIX (originally SunOS) systems. It has now essentially become an industry standard; all major UNIX like systems (Solaris, HP-UX, AIX®, Linux, NetBSD, OpenBSD, FreeBSD, etc) support NIS.

NIS was formerly known as Yellow Pages, but because of trademark issues, Sun changed the name. The old term (and yp) is still often seen and used.

It is a RPC-based client/server system that allows a group of machines within an NIS domain to share a common set of configuration files. This permits a system administrator to set up NIS client systems with only minimal configuration data and add, remove or modify configuration data from a single location.

It is similar to the Windows NT® domain system; although the internal implementation of the two are not at all similar, the basic functionality can be compared.

29.4.2 Terms/Processes You Should Know

There are several terms and several important user processes that you will come across when attempting to implement NIS on FreeBSD, whether you are trying to create an NIS server or act as an NIS client:

Term	Description
NIS domainname	An NIS master server and all of its clients (including its slave servers) have a NIS domainname. Similar to an Windows NT domain name, the NIS domainname does not have anything to do with DNS.
rpcbind	Must be running in order to enable RPC (Remote Procedure Call, a network protocol used by NIS). If **rpcbind** is not running, it will be impossible to run an NIS server, or to act as an NIS client.
ypbind	"Binds" an NIS client to its NIS server. It will take the NIS domainname from the system, and using RPC, connect to the server. **ypbind** is the core of client-server communication in an NIS environment; if **ypbind** dies on a client machine, it will not be able to access the NIS server.

Term	Description
ypserv	Should only be running on NIS servers; this is the NIS server process itself. If ypserv(8) dies, then the server will no longer be able to respond to NIS requests (hopefully, there is a slave server to take over for it). There are some implementations of NIS (but not the FreeBSD one), that do not try to reconnect to another server if the server it used before dies. Often, the only thing that helps in this case is to restart the server process (or even the whole server) or the **ypbind** process on the client.
rpc.yppasswdd	Another process that should only be running on NIS master servers; this is a daemon that will allow NIS clients to change their NIS passwords. If this daemon is not running, users will have to login to the NIS master server and change their passwords there.

29.4.3 How Does It Work?

There are three types of hosts in an NIS environment: master servers, slave servers, and clients. Servers act as a central repository for host configuration information. Master servers hold the authoritative copy of this information, while slave servers mirror this information for redundancy. Clients rely on the servers to provide this information to them.

Information in many files can be shared in this manner. The `master.passwd`, `group`, and `hosts` files are commonly shared via NIS. Whenever a process on a client needs information that would normally be found in these files locally, it makes a query to the NIS server that it is bound to instead.

29.4.3.1 Machine Types

- A *NIS master server*. This server, analogous to a Windows NT primary domain controller, maintains the files used by all of the NIS clients. The `passwd`, `group`, and other various files used by the NIS clients live on the master server.

  Note: It is possible for one machine to be an NIS master server for more than one NIS domain. However, this will not be covered in this introduction, which assumes a relatively small-scale NIS environment.

- *NIS slave servers*. Similar to the Windows NT backup domain controllers, NIS slave servers maintain copies of the NIS master's data files. NIS slave servers provide the redundancy, which is needed in important environments. They also help to balance the load of the master server: NIS Clients always attach to the NIS server whose response they get first, and this includes slave-server-replies.
- *NIS clients*. NIS clients, like most Windows NT workstations, authenticate against the NIS server (or the Windows NT domain controller in the Windows NT workstations case) to log on.

29.4.4 Using NIS/YP

This section will deal with setting up a sample NIS environment.

29.4.4.1 Planning

Let us assume that you are the administrator of a small university lab. This lab, which consists of 15 FreeBSD machines, currently has no centralized point of administration; each machine has its own `/etc/passwd` and `/etc/master.passwd`. These files are kept in sync with each other only through manual intervention; currently, when you add a user to the lab, you must run `adduser` on all 15 machines. Clearly, this has to change, so you have decided to convert the lab to use NIS, using two of the machines as servers.

Therefore, the configuration of the lab now looks something like:

Machine name	IP address	Machine role
`ellington`	`10.0.0.2`	NIS master
`coltrane`	`10.0.0.3`	NIS slave
`basie`	`10.0.0.4`	Faculty workstation
`bird`	`10.0.0.5`	Client machine
`cli[1-11]`	`10.0.0.[6-17]`	Other client machines

If you are setting up a NIS scheme for the first time, it is a good idea to think through how you want to go about it. No matter what the size of your network, there are a few decisions that need to be made.

29.4.4.1.1 Choosing a NIS Domain Name

This might not be the “domainname” that you are used to. It is more accurately called the “NIS domainname”. When a client broadcasts its requests for info, it includes the name of the NIS domain that it is part of. This is how multiple servers on one network can tell which server should answer which request. Think of the NIS domainname as the name for a group of hosts that are related in some way.

Some organizations choose to use their Internet domainname for their NIS domainname. This is not recommended as it can cause confusion when trying to debug network problems. The NIS domainname should be unique within your network and it is helpful if it describes the group of machines it represents. For example, the Art department at Acme Inc. might be in the “acme-art” NIS domain. For this example, assume you have chosen the name `test-domain`.

However, some operating systems (notably SunOS) use their NIS domain name as their Internet domain name. If one or more machines on your network have this restriction, you *must* use the Internet domain name as your NIS domain name.

29.4.4.1.2 Physical Server Requirements

There are several things to keep in mind when choosing a machine to use as a NIS server. One of the unfortunate things about NIS is the level of dependency the clients have on the server. If a client cannot contact the server for its NIS domain, very often the machine becomes unusable. The lack of user and group information causes most systems to temporarily freeze up. With this in mind you should make sure to choose a machine that will not be prone to being rebooted regularly, or one that might be used for development. The NIS server should ideally be a stand alone machine whose sole purpose in life is to be an NIS server. If you have a network that is not very heavily used, it is acceptable to put the NIS server on a machine running other services, just keep in mind that if the NIS server becomes unavailable, it will affect *all* of your NIS clients adversely.

29.4.4.2 NIS Servers

The canonical copies of all NIS information are stored on a single machine called the NIS master server. The databases used to store the information are called NIS maps. In FreeBSD, these maps are stored in `/var/yp/[domainname]` where `[domainname]` is the name of the NIS domain being served. A single NIS server can support several domains at once, therefore it is possible to have several such directories, one for each supported domain. Each domain will have its own independent set of maps.

NIS master and slave servers handle all NIS requests with the `ypserv` daemon. `ypserv` is responsible for receiving incoming requests from NIS clients, translating the requested domain and map name to a path to the corresponding database file and transmitting data from the database back to the client.

29.4.4.2.1 Setting Up a NIS Master Server

Setting up a master NIS server can be relatively straight forward, depending on your needs. FreeBSD comes with support for NIS out-of-the-box. All you need is to add the following lines to `/etc/rc.conf`, and FreeBSD will do the rest for you.

1. `nisdomainname="test-domain"`
   This line will set the NIS domainname to `test-domain` upon network setup (e.g. after reboot).
2. `nis_server_enable="YES"`
   This will tell FreeBSD to start up the NIS server processes when the networking is next brought up.
3. `nis_yppasswdd_enable="YES"`
   This will enable the `rpc.yppasswdd` daemon which, as mentioned above, will allow users to change their NIS password from a client machine.

> **Note:** Depending on your NIS setup, you may need to add further entries. See the section about NIS servers that are also NIS clients, below, for details.

After setting up the above entries, run the command `/etc/netstart` as superuser. It will set up everything for you, using the values you defined in `/etc/rc.conf`. As a last step, before initializing the NIS maps, start the **ypserv** daemon manually:

```
# /etc/rc.d/ypserv start
```

29.4.4.2.2 Initializing the NIS Maps

The *NIS maps* are database files, that are kept in the `/var/yp` directory. They are generated from configuration files in the `/etc` directory of the NIS master, with one exception: the `/etc/master.passwd` file. This is for a good reason, you do not want to propagate passwords to your `root` and other administrative accounts to all the servers in the NIS domain. Therefore, before we initialize the NIS maps, you should:

```
# cp /etc/master.passwd /var/yp/master.passwd
```

```
# cd /var/yp
# vi master.passwd
```

You should remove all entries regarding system accounts (`bin`, `tty`, `kmem`, `games`, etc), as well as any accounts that you do not want to be propagated to the NIS clients (for example `root` and any other UID 0 (superuser) accounts).

> **Note:** Make sure the `/var/yp/master.passwd` is neither group nor world readable (mode 600)! Use the `chmod` command, if appropriate.

When you have finished, it is time to initialize the NIS maps! FreeBSD includes a script named `ypinit` to do this for you (see its manual page for more information). Note that this script is available on most UNIX Operating Systems, but not on all. On Digital UNIX/Compaq Tru64 UNIX it is called `ypsetup`. Because we are generating maps for an NIS master, we are going to pass the `-m` option to `ypinit`. To generate the NIS maps, assuming you already performed the steps above, run:

```
ellington# ypinit -m test-domain
Server Type: MASTER Domain: test-domain
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

[..output from map generation..]

NIS Map update completed.
ellington has been setup as an YP master server without any errors.
```

`ypinit` should have created `/var/yp/Makefile` from `/var/yp/Makefile.dist`. When created, this file assumes that you are operating in a single server NIS environment with only FreeBSD machines. Since `test-domain` has a slave server as well, you must edit `/var/yp/Makefile`:

```
ellington# vi /var/yp/Makefile
```

You should comment out the line that says

```
NOPUSH = "True"
```

(if it is not commented out already).

29.4.4.2.3 Setting up a NIS Slave Server

Setting up an NIS slave server is even more simple than setting up the master. Log on to the slave server and edit the file `/etc/rc.conf` as you did before. The only difference is that we now must use the `-s` option when running `ypinit`. The `-s` option requires the name of the NIS master be passed to it as well, so our command line looks like:

```
coltrane# ypinit -s ellington test-domain

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.
```

You should now have a directory called `/var/yp/test-domain`. Copies of the NIS master server's maps should be in this directory. You will need to make sure that these stay updated. The following `/etc/crontab` entries on your slave servers should do the job:

```
20      *       *       *       *       root   /usr/libexec/ypxfr passwd.byname
21      *       *       *       *       root   /usr/libexec/ypxfr passwd.byuid
```

These two lines force the slave to sync its maps with the maps on the master server. These entries are not mandatory because the master server automatically attempts to push any map changes to its slaves. However, due to the importance of correct password information on other clients depending on the slave server, it is recommended to specifically force the password map updates frequently. This is especially important on busy networks where map updates might not always complete.

Now, run the command `/etc/netstart` on the slave server as well, which again starts the NIS server.

29.4.4.3 NIS Clients

An NIS client establishes what is called a binding to a particular NIS server using the `ypbind` daemon. `ypbind` checks the system's default domain (as set by the `domainname` command), and begins broadcasting RPC requests on the local network. These requests specify the name of the domain for which `ypbind` is attempting to establish a binding. If a server that has been configured to serve the requested domain receives one of the broadcasts, it will respond to `ypbind`, which will record the server's address. If there are several servers available (a master and several slaves, for example), `ypbind` will use the address of the first one to respond. From that point on, the client system will direct all of its NIS requests to that server. `ypbind` will occasionally "ping" the server to make sure it is still up and running. If it fails to receive a reply to one of its pings within a reasonable amount of time, `ypbind` will mark the domain as unbound and begin broadcasting again in the hopes of locating another server.

29.4.4.3.1 Setting Up a NIS Client

Setting up a FreeBSD machine to be a NIS client is fairly straightforward.

1. Edit the file `/etc/rc.conf` and add the following lines in order to set the NIS domainname and start `ypbind` upon network startup:

   ```
   nisdomainname="test-domain"
   nis_client_enable="YES"
   ```

2. To import all possible password entries from the NIS server, remove all user accounts from your `/etc/master.passwd` file and use `vipw` to add the following line to the end of the file:

   ```
   +:::::::::
   ```

Note: This line will afford anyone with a valid account in the NIS server's password maps an account. There are many ways to configure your NIS client by changing this line. See the netgroups section below for more information. For more detailed reading see O'Reilly's book on `Managing NFS and NIS`.

Note: You should keep at least one local account (i.e. not imported via NIS) in your `/etc/master.passwd` and this account should also be a member of the group `wheel`. If there is something wrong with NIS, this account can be used to log in remotely, become `root`, and fix things.

3. To import all possible group entries from the NIS server, add this line to your `/etc/group` file:

   ```
   +:*::
   ```

To start the NIS client immediately, execute the following commands as the superuser:

```
# /etc/netstart
# /etc/rc.d/ypbind start
```

After completing these steps, you should be able to run `ypcat passwd` and see the NIS server's passwd map.

29.4.5 NIS Security

In general, any remote user can issue an RPC to ypserv(8) and retrieve the contents of your NIS maps, provided the remote user knows your domainname. To prevent such unauthorized transactions, ypserv(8) supports a feature called "securenets" which can be used to restrict access to a given set of hosts. At startup, ypserv(8) will attempt to load the securenets information from a file called `/var/yp/securenets`.

Note: This path varies depending on the path specified with the `-p` option. This file contains entries that consist of a network specification and a network mask separated by white space. Lines starting with "#" are considered to be comments. A sample securenets file might look like this:

```
# allow connections from local host -- mandatory
127.0.0.1     255.255.255.255
# allow connections from any host
# on the 192.168.128.0 network
192.168.128.0 255.255.255.0
# allow connections from any host
# between 10.0.0.0 to 10.0.15.255
# this includes the machines in the testlab
10.0.0.0      255.255.240.0
```

If ypserv(8) receives a request from an address that matches one of these rules, it will process the request normally. If the address fails to match a rule, the request will be ignored and a warning message will be logged. If the `/var/yp/securenets` file does not exist, `ypserv` will allow connections from any host.

The ypserv program also has support for Wietse Venema's **TCP Wrapper** package. This allows the administrator to use the **TCP Wrapper** configuration files for access control instead of /var/yp/securenets.

> **Note:** While both of these access control mechanisms provide some security, they, like the privileged port test, are vulnerable to "IP spoofing" attacks. All NIS-related traffic should be blocked at your firewall.
>
> Servers using /var/yp/securenets may fail to serve legitimate NIS clients with archaic TCP/IP implementations. Some of these implementations set all host bits to zero when doing broadcasts and/or fail to observe the subnet mask when calculating the broadcast address. While some of these problems can be fixed by changing the client configuration, other problems may force the retirement of the client systems in question or the abandonment of /var/yp/securenets.
>
> Using /var/yp/securenets on a server with such an archaic implementation of TCP/IP is a really bad idea and will lead to loss of NIS functionality for large parts of your network.
>
> The use of the **TCP Wrapper** package increases the latency of your NIS server. The additional delay may be long enough to cause timeouts in client programs, especially in busy networks or with slow NIS servers. If one or more of your client systems suffers from these symptoms, you should convert the client systems in question into NIS slave servers and force them to bind to themselves.

29.4.6 Barring Some Users from Logging On

In our lab, there is a machine basie that is supposed to be a faculty only workstation. We do not want to take this machine out of the NIS domain, yet the passwd file on the master NIS server contains accounts for both faculty and students. What can we do?

There is a way to bar specific users from logging on to a machine, even if they are present in the NIS database. To do this, all you must do is add *-username* to the end of the /etc/master.passwd file on the client machine, where *username* is the username of the user you wish to bar from logging in. This should preferably be done using vipw, since vipw will sanity check your changes to /etc/master.passwd, as well as automatically rebuild the password database when you finish editing. For example, if we wanted to bar user bill from logging on to basie we would:

```
basie# vipw
[add -bill to the end, exit]
vipw: rebuilding the database...
vipw: done

basie# cat /etc/master.passwd

root:[password]:0:0::0:0:The super-user:/root:/bin/csh
toor:[password]: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 Using Netgroups

The method shown in the previous section works reasonably well if you need special rules for a very small number of users and/or machines. On larger networks, you *will* forget to bar some users from logging onto sensitive machines, or you may even have to modify each machine separately, thus losing the main benefit of NIS: *centralized* administration.

The NIS developers' solution for this problem is called *netgroups*. Their purpose and semantics can be compared to the normal groups used by UNIX file systems. The main differences are the lack of a numeric ID and the ability to define a netgroup by including both user accounts and other netgroups.

Netgroups were developed to handle large, complex networks with hundreds of users and machines. On one hand, this is a Good Thing if you are forced to deal with such a situation. On the other hand, this complexity makes it almost impossible to explain netgroups with really simple examples. The example used in the remainder of this section demonstrates this problem.

Let us assume that your successful introduction of NIS in your laboratory caught your superiors' interest. Your next job is to extend your NIS domain to cover some of the other machines on campus. The two tables contain the names of the new users and new machines as well as brief descriptions of them.

User Name(s)	Description
`alpha`, `beta`	Normal employees of the IT department
`charlie`, `delta`	The new apprentices of the IT department
`echo`, `foxtrott`, `golf`, ...	Ordinary employees
`able`, `baker`, ...	The current interns

Machine Name(s)	Description
`war`, `death`, `famine`, `pollution`	Your most important servers. Only the IT employees are allowed to log onto these machines.
`pride`, `greed`, `envy`, `wrath`, `lust`, `sloth`	Less important servers. All members of the IT department are allowed to login onto these machines.
`one`, `two`, `three`, `four`, ...	Ordinary workstations. Only the *real* employees are allowed to use these machines.
`trashcan`	A very old machine without any critical data. Even the intern is allowed to use this box.

If you tried to implement these restrictions by separately blocking each user, you would have to add one `-user` line to each system's `passwd` for each user who is not allowed to login onto that system. If you forget just one entry, you could be in trouble. It may be feasible to do this correctly during the initial setup, however you *will* eventually forget

to add the lines for new users during day-to-day operations. After all, Murphy was an optimist.

Handling this situation with netgroups offers several advantages. Each user need not be handled separately; you assign a user to one or more netgroups and allow or forbid logins for all members of the netgroup. If you add a new machine, you will only have to define login restrictions for netgroups. If a new user is added, you will only have to add the user to one or more netgroups. Those changes are independent of each other: no more "for each combination of user and machine do..." If your NIS setup is planned carefully, you will only have to modify exactly one central configuration file to grant or deny access to machines.

The first step is the initialization of the NIS map netgroup. FreeBSD's ypinit(8) does not create this map by default, but its NIS implementation will support it once it has been created. To create an empty map, simply type

```
ellington# vi /var/yp/netgroup
```

and start adding content. For our example, we need at least four netgroups: IT employees, IT apprentices, normal employees and interns.

```
IT_EMP  (,alpha,test-domain)    (,beta,test-domain)
IT_APP  (,charlie,test-domain)  (,delta,test-domain)
USERS   (,echo,test-domain)     (,foxtrott,test-domain) \
        (,golf,test-domain)
INTERNS (,able,test-domain)     (,baker,test-domain)
```

`IT_EMP`, `IT_APP` etc. are the names of the netgroups. Each bracketed group adds one or more user accounts to it. The three fields inside a group are:

1. The name of the host(s) where the following items are valid. If you do not specify a hostname, the entry is valid on all hosts. If you do specify a hostname, you will enter a realm of darkness, horror and utter confusion.
2. The name of the account that belongs to this netgroup.
3. The NIS domain for the account. You can import accounts from other NIS domains into your netgroup if you are one of the unlucky fellows with more than one NIS domain.

Each of these fields can contain wildcards. See netgroup(5) for details.

> **Note:** Netgroup names longer than 8 characters should not be used, especially if you have machines running other operating systems within your NIS domain. The names are case sensitive; using capital letters for your netgroup names is an easy way to distinguish between user, machine and netgroup names.
>
> Some NIS clients (other than FreeBSD) cannot handle netgroups with a large number of entries. For example, some older versions of SunOS start to cause trouble if a netgroup contains more than 15 *entries*. You can circumvent this limit by creating several sub-netgroups with 15 users or less and a real netgroup that consists of the sub-netgroups:
>
> ```
> BIGGRP1  (,joe1,domain)  (,joe2,domain)  (,joe3,domain) [...]
> BIGGRP2  (,joe16,domain)  (,joe17,domain) [...]
> BIGGRP3  (,joe31,domain)  (,joe32,domain)
> BIGGROUP  BIGGRP1 BIGGRP2 BIGGRP3
> ```
>
> You can repeat this process if you need more than 225 users within a single netgroup.

Activating and distributing your new NIS map is easy:

```
ellington# cd /var/yp
ellington# make
```

This will generate the three NIS maps `netgroup`, `netgroup.byhost` and `netgroup.byuser`. Use ypcat(1) to check if your new NIS maps are available:

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

The output of the first command should resemble the contents of `/var/yp/netgroup`. The second command will not produce output if you have not specified host-specific netgroups. The third command can be used to get the list of netgroups for a user.

The client setup is quite simple. To configure the server `war`, you only have to start vipw(8) and replace the line

```
+::::::::::
```

with

```
+@IT_EMP:::::::::
```

Now, only the data for the users defined in the netgroup `IT_EMP` is imported into `war`'s password database and only these users are allowed to login.

Unfortunately, this limitation also applies to the ~ function of the shell and all routines converting between user names and numerical user IDs. In other words, `cd ~`*`user`* will not work, `ls -l` will show the numerical ID instead of the username and `find . -user joe -print` will fail with `No such user`. To fix this, you will have to import all user entries *without allowing them to login onto your servers*.

This can be achieved by adding another line to `/etc/master.passwd`. This line should contain:

`+::::::::::/sbin/nologin`, meaning "Import all entries but replace the shell with `/sbin/nologin` in the imported entries". You can replace any field in the `passwd` entry by placing a default value in your `/etc/master.passwd`.

> **Warning:** Make sure that the line `+::::::::::/sbin/nologin` is placed after `+@IT_EMP:::::::::`. Otherwise, all user accounts imported from NIS will have `/sbin/nologin` as their login shell.

After this change, you will only have to change one NIS map if a new employee joins the IT department. You could use a similar approach for the less important servers by replacing the old `+::::::::::` in their local version of `/etc/master.passwd` with something like this:

```
+@IT_EMP:::::::::
+@IT_APP:::::::::
+::::::::::/sbin/nologin
```

The corresponding lines for the normal workstations could be:

```
+@IT_EMP:::::::::
+@USERS:::::::::
+::::::::::/sbin/nologin
```

And everything would be fine until there is a policy change a few weeks later: The IT department starts hiring interns. The IT interns are allowed to use the normal workstations and the less important servers; and the IT apprentices are allowed to login onto the main servers. You add a new netgroup `IT_INTERN`, add the new IT interns to this netgroup and start to change the configuration on each and every machine... As the old saying goes: “Errors in centralized planning lead to global mess”.

NIS’ ability to create netgroups from other netgroups can be used to prevent situations like these. One possibility is the creation of role-based netgroups. For example, you could create a netgroup called `BIGSRV` to define the login restrictions for the important servers, another netgroup called `SMALLSRV` for the less important servers and a third netgroup called `USERBOX` for the normal workstations. Each of these netgroups contains the netgroups that are allowed to login onto these machines. The new entries for your NIS map netgroup should look like this:

```
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP  ITINTERN
USERBOX   IT_EMP  ITINTERN USERS
```

This method of defining login restrictions works reasonably well if you can define groups of machines with identical restrictions. Unfortunately, this is the exception and not the rule. Most of the time, you will need the ability to define login restrictions on a per-machine basis.

Machine-specific netgroup definitions are the other possibility to deal with the policy change outlined above. In this scenario, the `/etc/master.passwd` of each box contains two lines starting with “+”. The first of them adds a netgroup with the accounts allowed to login onto this machine, the second one adds all other accounts with `/sbin/nologin` as shell. It is a good idea to use the “ALL-CAPS” version of the machine name as the name of the netgroup. In other words, the lines should look like this:

```
+@BOXNAME:::::::::
+::::::::::/sbin/nologin
```

Once you have completed this task for all your machines, you will not have to modify the local versions of `/etc/master.passwd` ever again. All further changes can be handled by modifying the NIS map. Here is an example of a possible netgroup map for this scenario with some additional goodies:

```
# Define groups of users first
IT_EMP    (,alpha,test-domain)    (,beta,test-domain)
IT_APP    (,charlie,test-domain)  (,delta,test-domain)
DEPT1     (,echo,test-domain)     (,foxtrott,test-domain)
DEPT2     (,golf,test-domain)     (,hotel,test-domain)
DEPT3     (,india,test-domain)    (,juliet,test-domain)
ITINTERN  (,kilo,test-domain)     (,lima,test-domain)
D_INTERNS (,able,test-domain)     (,baker,test-domain)
#
# Now, define some groups based on roles
USERS     DEPT1   DEPT2     DEPT3
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP    ITINTERN
USERBOX   IT_EMP  ITINTERN  USERS
#
# And a groups for a special tasks
# Allow echo and golf to access our anti-virus-machine
SECURITY  IT_EMP  (,echo,test-domain)  (,golf,test-domain)
#
# machine-based netgroups
```

```
# Our main servers
WAR       BIGSRV
FAMINE    BIGSRV
# User india needs access to this server
POLLUTION  BIGSRV  (,india,test-domain)
#
# This one is really important and needs more access restrictions
DEATH     IT_EMP
#
# The anti-virus-machine mentioned above
ONE       SECURITY
#
# Restrict a machine to a single user
TWO       (,hotel,test-domain)
# [...more groups to follow]
```

If you are using some kind of database to manage your user accounts, you should be able to create the first part of the map with your database's report tools. This way, new users will automatically have access to the boxes.

One last word of caution: It may not always be advisable to use machine-based netgroups. If you are deploying a couple of dozen or even hundreds of identical machines for student labs, you should use role-based netgroups instead of machine-based netgroups to keep the size of the NIS map within reasonable limits.

29.4.8 Important Things to Remember

There are still a couple of things that you will need to do differently now that you are in an NIS environment.

- Every time you wish to add a user to the lab, you must add it to the master NIS server *only*, and *you must remember to rebuild the NIS maps*. If you forget to do this, the new user will not be able to login anywhere except on the NIS master. For example, if we needed to add a new user `jsmith` to the lab, we would:

  ```
  # pw useradd jsmith
  # cd /var/yp
  # make test-domain
  ```

  You could also run `adduser jsmith` instead of `pw useradd jsmith`.

- *Keep the administration accounts out of the NIS maps*. You do not want to be propagating administrative accounts and passwords to machines that will have users that should not have access to those accounts.

- *Keep the NIS master and slave secure, and minimize their downtime*. If somebody either hacks or simply turns off these machines, they have effectively rendered many people without the ability to login to the lab.

  This is the chief weakness of any centralized administration system. If you do not protect your NIS servers, you will have a lot of angry users!

29.4.9 NIS v1 Compatibility

FreeBSD's **ypserv** has some support for serving NIS v1 clients. FreeBSD's NIS implementation only uses the NIS v2 protocol, however other implementations include support for the v1 protocol for backwards compatibility with older systems. The **ypbind** daemons supplied with these systems will try to establish a binding to an NIS v1 server even though they may never actually need it (and they may persist in broadcasting in search of one even after they receive

a response from a v2 server). Note that while support for normal client calls is provided, this version of **ypserv** does not handle v1 map transfer requests; consequently, it cannot be used as a master or slave in conjunction with older NIS servers that only support the v1 protocol. Fortunately, there probably are not any such servers still in use today.

29.4.10 NIS Servers That Are Also NIS Clients

Care must be taken when running **ypserv** in a multi-server domain where the server machines are also NIS clients. It is generally a good idea to force the servers to bind to themselves rather than allowing them to broadcast bind requests and possibly become bound to each other. Strange failure modes can result if one server goes down and others are dependent upon it. Eventually all the clients will time out and attempt to bind to other servers, but the delay involved can be considerable and the failure mode is still present since the servers might bind to each other all over again.

You can force a host to bind to a particular server by running `ypbind` with the `-S` flag. If you do not want to do this manually each time you reboot your NIS server, you can add the following lines to your `/etc/rc.conf`:

```
nis_client_enable="YES" # run client stuff as well
nis_client_flags="-S NIS domain,server"
```

See ypbind(8) for further information.

29.4.11 Password Formats

One of the most common issues that people run into when trying to implement NIS is password format compatibility. If your NIS server is using DES encrypted passwords, it will only support clients that are also using DES. For example, if you have Solaris NIS clients in your network, then you will almost certainly need to use DES encrypted passwords.

To check which format your servers and clients are using, look at `/etc/login.conf`. If the host is configured to use DES encrypted passwords, then the `default` class will contain an entry like this:

```
default:\
 :passwd_format=des:\
 :copyright=/etc/COPYRIGHT:\
 [Further entries elided]
```

Other possible values for the `passwd_format` capability include `blf` and `md5` (for Blowfish and MD5 encrypted passwords, respectively).

If you have made changes to `/etc/login.conf`, you will also need to rebuild the login capability database, which is achieved by running the following command as `root`:

```
# cap_mkdb /etc/login.conf
```

> **Note:** The format of passwords already in `/etc/master.passwd` will not be updated until a user changes his password for the first time *after* the login capability database is rebuilt.

Next, in order to ensure that passwords are encrypted with the format that you have chosen, you should also check that the `crypt_default` in `/etc/auth.conf` gives precedence to your chosen password format. To do this, place

the format that you have chosen first in the list. For example, when using DES encrypted passwords, the entry would be:

```
crypt_default = des blf md5
```

Having followed the above steps on each of the FreeBSD based NIS servers and clients, you can be sure that they all agree on which password format is used within your network. If you have trouble authenticating on an NIS client, this is a pretty good place to start looking for possible problems. Remember: if you want to deploy an NIS server for a heterogenous network, you will probably have to use DES on all systems because it is the lowest common standard.

29.5 Automatic Network Configuration (DHCP)

29.5.1 What Is DHCP?

DHCP, the Dynamic Host Configuration Protocol, describes the means by which a system can connect to a network and obtain the necessary information for communication upon that network. FreeBSD uses the OpenBSD `dhclient` taken from OpenBSD 3.7. All information here regarding `dhclient` is for use with either of the ISC or OpenBSD DHCP clients. The DHCP server is the one included in the ISC distribution.

29.5.2 What This Section Covers

This section describes both the client-side components of the ISC and OpenBSD DHCP client and server-side components of the ISC DHCP system. The client-side program, `dhclient`, comes integrated within FreeBSD, and the server-side portion is available from the `net/isc-dhcp31-server` port. The dhclient(8), dhcp-options(5), and dhclient.conf(5) manual pages, in addition to the references below, are useful resources.

29.5.3 How It Works

When `dhclient`, the DHCP client, is executed on the client machine, it begins broadcasting requests for configuration information. By default, these requests are on UDP port 68. The server replies on UDP 67, giving the client an IP address and other relevant network information such as netmask, router, and DNS servers. All of this information comes in the form of a DHCP “lease” and is only valid for a certain time (configured by the DHCP server maintainer). In this manner, stale IP addresses for clients no longer connected to the network can be automatically reclaimed.

DHCP clients can obtain a great deal of information from the server. An exhaustive list may be found in dhcp-options(5).

29.5.4 FreeBSD Integration

FreeBSD fully integrates the OpenBSD DHCP client, `dhclient`. DHCP client support is provided within both the installer and the base system, obviating the need for detailed knowledge of network configurations on any network that runs a DHCP server.

DHCP is supported by **sysinstall**. When configuring a network interface within **sysinstall**, the second question asked is: “Do you want to try DHCP configuration of the interface?”. Answering affirmatively will execute `dhclient`, and if successful, will fill in the network configuration information automatically.

There are two things you must do to have your system use DHCP upon startup:

- Make sure that the `bpf` device is compiled into your kernel. To do this, add `device bpf` to your kernel configuration file, and rebuild the kernel. For more information about building kernels, see Chapter 8.

  The `bpf` device is already part of the `GENERIC` kernel that is supplied with FreeBSD, so if you do not have a custom kernel, you should not need to create one in order to get DHCP working.

  > **Note:** For those who are particularly security conscious, you should be warned that `bpf` is also the device that allows packet sniffers to work correctly (although they still have to be run as `root`). `bpf` *is* required to use DHCP, but if you are very sensitive about security, you probably should not add `bpf` to your kernel in the expectation that at some point in the future you will be using DHCP.

- Edit your `/etc/rc.conf` to include the following:

  ```
  ifconfig_fxp0="DHCP"
  ```

  > **Note:** Be sure to replace `fxp0` with the designation for the interface that you wish to dynamically configure, as described in Section 11.8.

  If you are using a different location for `dhclient`, or if you wish to pass additional flags to `dhclient`, also include the following (editing as necessary):

  ```
  dhclient_program="/sbin/dhclient"
  dhclient_flags=""
  ```

The DHCP server, **dhcpd**, is included as part of the `net/isc-dhcp31-server` port in the ports collection. This port contains the ISC DHCP server and documentation.

29.5.5 Files

- `/etc/dhclient.conf`

  `dhclient` requires a configuration file, `/etc/dhclient.conf`. Typically the file contains only comments, the defaults being reasonably sane. This configuration file is described by the dhclient.conf(5) manual page.

- `/sbin/dhclient`

  `dhclient` is statically linked and resides in `/sbin`. The dhclient(8) manual page gives more information about `dhclient`.

- `/sbin/dhclient-script`

  `dhclient-script` is the FreeBSD-specific DHCP client configuration script. It is described in dhclient-script(8), but should not need any user modification to function properly.

- `/var/db/dhclient.leases`

  The DHCP client keeps a database of valid leases in this file, which is written as a log. dhclient.leases(5) gives a slightly longer description.

29.5.6 Further Reading

The DHCP protocol is fully described in RFC 2131 (http://www.freesoft.org/CIE/RFC/2131/). An informational resource has also been set up at http://www.dhcp.org/.

29.5.7 Installing and Configuring a DHCP Server

29.5.7.1 What This Section Covers

This section provides information on how to configure a FreeBSD system to act as a DHCP server using the ISC (Internet Systems Consortium) implementation of the DHCP server.

The server is not provided as part of FreeBSD, and so you will need to install the `net/isc-dhcp31-server` port to provide this service. See Chapter 4 for more information on using the Ports Collection.

29.5.7.2 DHCP Server Installation

In order to configure your FreeBSD system as a DHCP server, you will need to ensure that the bpf(4) device is compiled into your kernel. To do this, add `device bpf` to your kernel configuration file, and rebuild the kernel. For more information about building kernels, see Chapter 8.

The `bpf` device is already part of the `GENERIC` kernel that is supplied with FreeBSD, so you do not need to create a custom kernel in order to get DHCP working.

> **Note:** Those who are particularly security conscious should note that `bpf` is also the device that allows packet sniffers to work correctly (although such programs still need privileged access). `bpf` *is* required to use DHCP, but if you are very sensitive about security, you probably should not include `bpf` in your kernel purely because you expect to use DHCP at some point in the future.

The next thing that you will need to do is edit the sample `dhcpd.conf` which was installed by the `net/isc-dhcp31-server` port. By default, this will be `/usr/local/etc/dhcpd.conf.sample`, and you should copy this to `/usr/local/etc/dhcpd.conf` before proceeding to make changes.

29.5.7.3 Configuring the DHCP Server

`dhcpd.conf` is comprised of declarations regarding subnets and hosts, and is perhaps most easily explained using an example :

```
option domain-name "example.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 mailhost.example.com;(10)
}
```

❶ This option specifies the domain that will be provided to clients as the default search domain. See resolv.conf(5) for more information on what this means.

❷ This option specifies a comma separated list of DNS servers that the client should use.

❸ The netmask that will be provided to clients.

❹ A client may request a specific length of time that a lease will be valid. Otherwise the server will assign a lease with this expiry value (in seconds).

❺ This is the maximum length of time that the server will lease for. Should a client request a longer lease, a lease will be issued, although it will only be valid for `max-lease-time` seconds.

❻ This option specifies whether the DHCP server should attempt to update DNS when a lease is accepted or released. In the ISC implementation, this option is *required*.

❼ This denotes which IP addresses should be used in the pool reserved for allocating to clients. IP addresses between, and including, the ones stated are handed out to clients.

❽ Declares the default gateway that will be provided to clients.

❾ The hardware MAC address of a host (so that the DHCP server can recognize a host when it makes a request).

(10) Specifies that the host should always be given the same IP address. Note that using a hostname is correct here, since the DHCP server will resolve the hostname itself before returning the lease information.

Once you have finished writing your `dhcpd.conf`, you should enable the DHCP server in `/etc/rc.conf`, i.e. by adding:

```
dhcpd_enable="YES"
dhcpd_ifaces="dc0"
```

Replace the `dc0` interface name with the interface (or interfaces, separated by whitespace) that your DHCP server should listen on for DHCP client requests.

Then, you can proceed to start the server by issuing the following command:

```
# /usr/local/etc/rc.d/isc-dhcpd start
```

Should you need to make changes to the configuration of your server in the future, it is important to note that sending a `SIGHUP` signal to **dhcpd** does *not* result in the configuration being reloaded, as it does with most daemons. You will need to send a `SIGTERM` signal to stop the process, and then restart it using the command above.

29.5.7.4 Files

- `/usr/local/sbin/dhcpd`

  dhcpd is statically linked and resides in `/usr/local/sbin`. The dhcpd(8) manual page installed with the port gives more information about **dhcpd**.

- `/usr/local/etc/dhcpd.conf`

  dhcpd requires a configuration file, `/usr/local/etc/dhcpd.conf` before it will start providing service to clients. This file needs to contain all the information that should be provided to clients that are being serviced, along with information regarding the operation of the server. This configuration file is described by the dhcpd.conf(5) manual page installed by the port.

- `/var/db/dhcpd.leases`

  The DHCP server keeps a database of leases it has issued in this file, which is written as a log. The manual page dhcpd.leases(5), installed by the port gives a slightly longer description.

- `/usr/local/sbin/dhcrelay`

  dhcrelay is used in advanced environments where one DHCP server forwards a request from a client to another DHCP server on a separate network. If you require this functionality, then install the `net/isc-dhcp31-relay` port. The dhcrelay(8) manual page provided with the port contains more detail.

29.6 Domain Name System (DNS)

29.6.1 Overview

FreeBSD utilizes, by default, a version of BIND (Berkeley Internet Name Domain), which is the most common implementation of the DNS protocol. DNS is the protocol through which names are mapped to IP addresses, and vice versa. For example, a query for `www.FreeBSD.org` will receive a reply with the IP address of The FreeBSD Project's web server, whereas, a query for `ftp.FreeBSD.org` will return the IP address of the corresponding FTP machine. Likewise, the opposite can happen. A query for an IP address can resolve its hostname. It is not necessary to run a name server to perform DNS lookups on a system.

FreeBSD currently comes with BIND9 DNS server software by default. Our installation provides enhanced security features, a new file system layout and automated chroot(8) configuration.

DNS is coordinated across the Internet through a somewhat complex system of authoritative root, Top Level Domain (TLD), and other smaller-scale name servers which host and cache individual domain information.

Currently, BIND is maintained by the Internet Systems Consortium https://www.isc.org/.

29.6.2 Terminology

To understand this document, some terms related to DNS must be understood.

Term	Definition
Forward DNS	Mapping of hostnames to IP addresses.

Term	Definition
Origin	Refers to the domain covered in a particular zone file.
named, BIND	Common names for the BIND name server package within FreeBSD.
Resolver	A system process through which a machine queries a name server for zone information.
Reverse DNS	Mapping of IP addresses to hostnames.
Root zone	The beginning of the Internet zone hierarchy. All zones fall under the root zone, similar to how all files in a file system fall under the root directory.
Zone	An individual domain, subdomain, or portion of the DNS administered by the same authority.

Examples of zones:

- `.` is how the root zone is usually referred to in documentation.
- `org.` is a Top Level Domain (TLD) under the root zone.
- `example.org.` is a zone under the `org.` TLD.
- `1.168.192.in-addr.arpa` is a zone referencing all IP addresses which fall under the `192.168.1.*` IP address space.

As one can see, the more specific part of a hostname appears to its left. For example, `example.org.` is more specific than `org.`, as `org.` is more specific than the root zone. The layout of each part of a hostname is much like a file system: the `/dev` directory falls within the root, and so on.

29.6.3 Reasons to Run a Name Server

Name servers generally come in two forms: authoritative name servers, and caching name servers.

An authoritative name server is needed when:

- One wants to serve DNS information to the world, replying authoritatively to queries.
- A domain, such as `example.org`, is registered and IP addresses need to be assigned to hostnames under it.
- An IP address block requires reverse DNS entries (IP to hostname).
- A backup or second name server, called a slave, will reply to queries.

A caching name server is needed when:

- A local DNS server may cache and respond more quickly than querying an outside name server.

When one queries for `www.FreeBSD.org`, the resolver usually queries the uplink ISP's name server, and retrieves the reply. With a local, caching DNS server, the query only has to be made once to the outside world by the caching DNS server. Additional queries will not have to go outside the local network, since the information is cached locally.

29.6.4 How It Works

In FreeBSD, the BIND daemon is called **named**.

File	Description
named(8)	The BIND daemon.
rndc(8)	Name server control utility.
`/etc/namedb`	Directory where BIND zone information resides.
`/etc/namedb/named.conf`	Configuration file of the daemon.

Depending on how a given zone is configured on the server, the files related to that zone can be found in the `master`, `slave`, or `dynamic` subdirectories of the `/etc/namedb` directory. These files contain the DNS information that will be given out by the name server in response to queries.

29.6.5 Starting BIND

Since BIND is installed by default, configuring it is relatively simple.

The default **named** configuration is that of a basic resolving name server, running in a chroot(8) environment, and restricted to listening on the local IPv4 loopback address (127.0.0.1). To start the server one time with this configuration, use the following command:

```
# /etc/rc.d/named onestart
```

To ensure the **named** daemon is started at boot each time, put the following line into the `/etc/rc.conf`:

```
named_enable="YES"
```

There are obviously many configuration options for `/etc/namedb/named.conf` that are beyond the scope of this document. However, if you are interested in the startup options for **named** on FreeBSD, take a look at the `named_*` flags in `/etc/defaults/rc.conf` and consult the rc.conf(5) manual page. The Section 11.7 section is also a good read.

29.6.6 Configuration Files

Configuration files for **named** currently reside in `/etc/namedb` directory and will need modification before use unless all that is needed is a simple resolver. This is where most of the configuration will be performed.

29.6.6.1 `/etc/namedb/named.conf`

```
// $FreeBSD$
//
// Refer to the named.conf(5) and named(8) man pages, and the documentation
// in /usr/share/doc/bind9 for more details.
//
// If you are going to set up an authoritative server, make sure you
// understand the hairy details of how DNS works.  Even with
// simple mistakes, you can break connectivity for affected parties,
// or cause huge amounts of useless Internet traffic.

options {
 // Relative to the chroot directory, if any
 directory "/etc/namedb";
```

```
 pid-file "/var/run/named/pid";
 dump-file "/var/dump/named_dump.db";
 statistics-file "/var/stats/named.stats";

// If named is being used only as a local resolver, this is a safe default.
// For named to be accessible to the network, comment this option, specify
// the proper IP address, or delete this option.
 listen-on { 127.0.0.1; };

// If you have IPv6 enabled on this system, uncomment this option for
// use as a local resolver.  To give access to the network, specify
// an IPv6 address, or the keyword "any".
// listen-on-v6 { ::1; };

// These zones are already covered by the empty zones listed below.
// If you remove the related empty zones below, comment these lines out.
 disable-empty-zone "255.255.255.255.IN-ADDR.ARPA";
 disable-empty-zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA";
 disable-empty-zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA";

// If you've got a DNS server around at your upstream provider, enter
// its IP address here, and enable the line below.  This will make you
// benefit from its cache, thus reduce overall DNS traffic in the Internet.
/*
 forwarders {
  127.0.0.1;
 };
*/

// If the 'forwarders' clause is not empty the default is to 'forward first'
// which will fall back to sending a query from your local server if the name
// servers in 'forwarders' do not have the answer.  Alternatively you can
// force your name server to never initiate queries of its own by enabling the
// following line:
// forward only;

// If you wish to have forwarding configured automatically based on
// the entries in /etc/resolv.conf, uncomment the following line and
// set named_auto_forward=yes in /etc/rc.conf.  You can also enable
// named_auto_forward_only (the effect of which is described above).
// include "/etc/namedb/auto_forward.conf";
```

Just as the comment says, to benefit from an uplink's cache, `forwarders` can be enabled here. Under normal circumstances, a name server will recursively query the Internet looking at certain name servers until it finds the answer it is looking for. Having this enabled will have it query the uplink's name server (or name server provided) first, taking advantage of its cache. If the uplink name server in question is a heavily trafficked, fast name server, enabling this may be worthwhile.

> **Warning:** `127.0.0.1` will *not* work here. Change this IP address to a name server at your uplink.

```
 /*
```

```
    Modern versions of BIND use a random UDP port for each outgoing
    query by default in order to dramatically reduce the possibility
    of cache poisoning.  All users are strongly encouraged to utilize
    this feature, and to configure their firewalls to accommodate it.

    AS A LAST RESORT in order to get around a restrictive firewall
    policy you can try enabling the option below.  Use of this option
    will significantly reduce your ability to withstand cache poisoning
    attacks, and should be avoided if at all possible.

    Replace NNNNN in the example with a number between 49160 and 65530.
 */
 // query-source address * port NNNNN;
};

// If you enable a local name server, don't forget to enter 127.0.0.1
// first in your /etc/resolv.conf so this server will be queried.
// Also, make sure to enable it in /etc/rc.conf.

// The traditional root hints mechanism. Use this, OR the slave zones below.
zone "." { type hint; file "named.root"; };

/* Slaving the following zones from the root name servers has some
 significant advantages:
 1. Faster local resolution for your users
 2. No spurious traffic will be sent from your network to the roots
 3. Greater resilience to any potential root server failure/DDoS

 On the other hand, this method requires more monitoring than the
 hints file to be sure that an unexpected failure mode has not
 incapacitated your server.  Name servers that are serving a lot
 of clients will benefit more from this approach than individual
 hosts.  Use with caution.

 To use this mechanism, uncomment the entries below, and comment
 the hint zone above.
*/
/*
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;
};
*/

/* Serving the following zones locally will prevent any queries
 for these zones leaving your network and going to the root
 name servers.  This has two significant advantages:
 1. Faster local resolution for your users
 2. No spurious traffic will be sent from your network to the roots
*/
// 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"; };

// RFC 1912-style zone for IPv6 localhost address
zone "0.ip6.arpa" { type master; file "master/localhost-reverse.db"; };

// "This" Network (RFCs 1912 and 3330)
zone "0.in-addr.arpa"  { type master; file "master/empty.db"; };

// Private Use Networks (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"; };

// Link-local/APIPA (RFCs 3330 and 3927)
zone "254.169.in-addr.arpa" { type master; file "master/empty.db"; };

// TEST-NET for Documentation (RFC 3330)
zone "2.0.192.in-addr.arpa" { type master; file "master/empty.db"; };
```

```
// Router Benchmark Testing (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"; };

// IANA Reserved - Old Class E Space
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"; };

// IPv6 Unassigned Addresses (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 Link Local (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"; };

// IPv6 Deprecated Site-Local Addresses (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"; };

// IP6.INT is Deprecated (RFC 4159)
zone "ip6.int"   { type master; file "master/empty.db"; };

// NB: Do not use the IP addresses below, they are faked, and only
// serve demonstration/documentation purposes!
//
// Example slave zone config entries.  It can be convenient to become
// a slave at least for the zone your own domain is in.  Ask
// your network administrator for the IP address of the responsible
// master name server.
//
// Do not forget to include the reverse lookup zone!
// This is named after the first bytes of the IP address, in reverse
// order, with ".IN-ADDR.ARPA" appended, or ".IP6.ARPA" for IPv6.
//
// Before starting to set up a master zone, make sure you fully
// understand how DNS and BIND work.  There are sometimes
// non-obvious pitfalls.  Setting up a slave zone is usually simpler.
//
// NB: Don't blindly enable the examples below. :-)  Use actual names
// and addresses instead.

/* An example dynamic zone
key "exampleorgkey" {
 algorithm hmac-md5;
 secret "sf87HJqjkqh8ac87a02lla==";
};
zone "example.org" {
 type master;
 allow-update {
  key "exampleorgkey";
 };
 file "dynamic/example.org";
```

```
};
*/

/* Example of a slave reverse zone
zone "1.168.192.in-addr.arpa" {
 type slave;
 file "slave/1.168.192.in-addr.arpa";
 masters {
  192.168.1.1;
 };
};
*/
```

In `named.conf`, these are examples of slave entries for a forward and reverse zone.

For each new zone served, a new zone entry must be added to `named.conf`.

For example, the simplest zone entry for `example.org` can look like:

```
zone "example.org" {
 type master;
 file "master/example.org";
};
```

The zone is a master, as indicated by the `type` statement, holding its zone information in `/etc/namedb/master/example.org` indicated by the `file` statement.

```
zone "example.org" {
 type slave;
 file "slave/example.org";
};
```

In the slave case, the zone information is transferred from the master name server for the particular zone, and saved in the file specified. If and when the master server dies or is unreachable, the slave name server will have the transferred zone information and will be able to serve it.

29.6.6.2 Zone Files

An example master zone file for `example.org` (existing within `/etc/namedb/master/example.org`) is as follows:

```
$TTL 3600        ; 1 hour default TTL
example.org.    IN      SOA      ns1.example.org. admin.example.org. (
                                 2006051501      ; Serial
                                 10800           ; Refresh
                                 3600            ; Retry
                                 604800          ; Expire
                                 300             ; Negative Reponse TTL
                         )

; DNS Servers
                IN      NS      ns1.example.org.
                IN      NS      ns2.example.org.
```

```
; MX Records
                IN      MX 10   mx.example.org.
                IN      MX 20   mail.example.org.

                IN      A       192.168.1.1

; Machine Names
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
mail            IN      A       192.168.1.5

; Aliases
www             IN      CNAME   example.org.
```

Note that every hostname ending in a "." is an exact hostname, whereas everything without a trailing "." is relative to the origin. For example, `ns1` is translated into `ns1.`*`example.org.`*

The format of a zone file follows:

```
recordname      IN recordtype   value
```

The most commonly used DNS records:

SOA

: start of zone authority

NS

: an authoritative name server

A

: a host address

CNAME

: the canonical name for an alias

MX

: mail exchanger

PTR

: a domain name pointer (used in reverse DNS)

```
example.org. IN SOA ns1.example.org. admin.example.org. (
                        2006051501      ; Serial
                        10800           ; Refresh after 3 hours
                        3600            ; Retry after 1 hour
                        604800          ; Expire after 1 week
                        300 )           ; Negative Reponse TTL
```

`example.org.`

the domain name, also the origin for this zone file.

`ns1.example.org.`

the primary/authoritative name server for this zone.

`admin.example.org.`

the responsible person for this zone, email address with "@" replaced. (<`admin@example.org`> becomes `admin.example.org`)

`2006051501`

the serial number of the file. This must be incremented each time the zone file is modified. Nowadays, many admins prefer a `yyyymmddrr` format for the serial number. `2006051501` would mean last modified 05/15/2006, the latter `01` being the first time the zone file has been modified this day. The serial number is important as it alerts slave name servers for a zone when it is updated.

```
    IN NS           ns1.example.org.
```

This is an NS entry. Every name server that is going to reply authoritatively for the zone must have one of these entries.

```
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
mail            IN      A       192.168.1.5
```

The A record indicates machine names. As seen above, `ns1.example.org` would resolve to `192.168.1.2`.

```
                IN      A       192.168.1.1
```

This line assigns IP address `192.168.1.1` to the current origin, in this case `example.org`.

```
www             IN CNAME        @
```

The canonical name record is usually used for giving aliases to a machine. In the example, `www` is aliased to the "master" machine whose name happens to be the same as the domain name `example.org` (`192.168.1.1`). CNAMEs can never be used together with another kind of record for the same hostname.

```
               IN MX   10      mail.example.org.
```

The MX record indicates which mail servers are responsible for handling incoming mail for the zone. `mail.example.org` is the hostname of a mail server, and 10 is the priority of that mail server.

One can have several mail servers, with priorities of 10, 20 and so on. A mail server attempting to deliver to `example.org` would first try the highest priority MX (the record with the lowest priority number), then the second highest, etc, until the mail can be properly delivered.

For in-addr.arpa zone files (reverse DNS), the same format is used, except with PTR entries instead of A or CNAME.

```
$TTL 3600
```

```
1.168.192.in-addr.arpa. IN SOA ns1.example.org. admin.example.org. (
                        2006051501      ; Serial
                        10800           ; Refresh
                        3600            ; Retry
                        604800          ; Expire
                        300 )           ; Negative Reponse TTL

        IN      NS      ns1.example.org.
        IN      NS      ns2.example.org.

1       IN      PTR     example.org.
2       IN      PTR     ns1.example.org.
3       IN      PTR     ns2.example.org.
4       IN      PTR     mx.example.org.
5       IN      PTR     mail.example.org.
```

This file gives the proper IP address to hostname mappings for the above fictitious domain.

It is worth noting that all names on the right side of a PTR record need to be fully qualified (i.e., end in a “.”).

29.6.7 Caching Name Server

A caching name server is a name server whose primary role is to resolve recursive queries. It simply asks queries of its own, and remembers the answers for later use.

29.6.8 Security

Although BIND is the most common implementation of DNS, there is always the issue of security. Possible and exploitable security holes are sometimes found.

While FreeBSD automatically drops **named** into a chroot(8) environment; there are several other security mechanisms in place which could help to lure off possible DNS service attacks.

It is always good idea to read CERT (http://www.cert.org/)’s security advisories and to subscribe to the FreeBSD security notifications mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications) to stay up to date with the current Internet and FreeBSD security issues.

> **Tip:** If a problem arises, keeping sources up to date and having a fresh build of **named** may help.

29.6.9 Further Reading

BIND/**named** manual pages: rndc(8) named(8) named.conf(5)

- Official ISC BIND Page (https://www.isc.org/software/bind)
- Official ISC BIND Forum (https://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 Apache HTTP Server

29.7.1 Overview

FreeBSD is used to run some of the busiest web sites in the world. The majority of web servers on the Internet are using the **Apache HTTP Server**. **Apache** software packages should be included on your FreeBSD installation media. If you did not install **Apache** when you first installed FreeBSD, then you can install it from the `www/apache13` or `www/apache22` port.

Once **Apache** has been installed successfully, it must be configured.

> **Note:** This section covers version 1.3.X of the **Apache HTTP Server** as that is the most widely used version for FreeBSD. **Apache** 2.X introduces many new technologies but they are not discussed here. For more information about **Apache** 2.X, please see http://httpd.apache.org/.

29.7.2 Configuration

The main **Apache HTTP Server** configuration file is installed as `/usr/local/etc/apache/httpd.conf` on FreeBSD. This file is a typical UNIX text configuration file with comment lines beginning with the # character. A comprehensive description of all possible configuration options is outside the scope of this book, so only the most frequently modified directives will be described here.

`ServerRoot "/usr/local"`

This specifies the default directory hierarchy for the **Apache** installation. Binaries are stored in the `bin` and `sbin` subdirectories of the server root, and configuration files are stored in `etc/apache`.

`ServerAdmin you@your.address`

The address to which problems with the server should be emailed. This address appears on some server-generated pages, such as error documents.

`ServerName www.example.com`

`ServerName` allows you to set a host name which is sent back to clients for your server if it is different to the one that the host is configured with (i.e., use `www` instead of the host's real name).

`DocumentRoot "/usr/local/www/data"`

`DocumentRoot`: The directory out of which you will serve your documents. By default, all requests are taken from this directory, but symbolic links and aliases may be used to point to other locations.

It is always a good idea to make backup copies of your **Apache** configuration file before making changes. Once you are satisfied with your initial configuration you are ready to start running **Apache**.

29.7.3 Running Apache

Apache does not run from the **inetd** super server as many other network servers do. It is configured to run standalone for better performance for incoming HTTP requests from client web browsers. A shell script wrapper is included to make starting, stopping, and restarting the server as simple as possible. To start up **Apache** for the first time, just run:

```
# /usr/local/sbin/apachectl start
```

You can stop the server at any time by typing:

```
# /usr/local/sbin/apachectl stop
```

After making changes to the configuration file for any reason, you will need to restart the server:

```
# /usr/local/sbin/apachectl restart
```

To restart **Apache** without aborting current connections, run:

```
# /usr/local/sbin/apachectl graceful
```

Additional information available at apachectl(8) manual page.

To launch **Apache** at system startup, add the following line to `/etc/rc.conf`:

```
apache_enable="YES"
```

or for **Apache** 2.2:

```
apache22_enable="YES"
```

If you would like to supply additional command line options for the **Apache** `httpd` program started at system boot, you may specify them with an additional line in `rc.conf`:

```
apache_flags=""
```

Now that the web server is running, you can view your web site by pointing a web browser to `http://localhost/`. The default web page that is displayed is `/usr/local/www/data/index.html`.

29.7.4 Virtual Hosting

Apache supports two different types of Virtual Hosting. The first method is Name-based Virtual Hosting. Name-based virtual hosting uses the clients HTTP/1.1 headers to figure out the hostname. This allows many different domains to share the same IP address.

To setup **Apache** to use Name-based Virtual Hosting add an entry like the following to your `httpd.conf`:

```
NameVirtualHost *
```

If your webserver was named `www.domain.tld` and you wanted to setup a virtual domain for `www.someotherdomain.tld` then you would add the following entries to `httpd.conf`:

```
<VirtualHost *>
ServerName www.domain.tld
DocumentRoot /www/domain.tld
```

```
</VirtualHost>

<VirtualHost *>
ServerName www.someotherdomain.tld
DocumentRoot /www/someotherdomain.tld
</VirtualHost>
```

Replace the addresses with the addresses you want to use and the path to the documents with what you are using.

For more information about setting up virtual hosts, please consult the official **Apache** documentation at: http://httpd.apache.org/docs/vhosts/.

29.7.5 Apache Modules

There are many different **Apache** modules available to add functionality to the basic server. The FreeBSD Ports Collection provides an easy way to install **Apache** together with some of the more popular add-on modules.

29.7.5.1 mod_ssl

The **mod_ssl** module uses the OpenSSL library to provide strong cryptography via the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols. This module provides everything necessary to request a signed certificate from a trusted certificate signing authority so that you can run a secure web server on FreeBSD.

If you have not yet installed **Apache**, then a version of **Apache** 1.3.X that includes **mod_ssl** may be installed with the `www/apache13-modssl` port. SSL support is also available for **Apache** 2.X in the `www/apache22` port, where it is enabled by default.

29.7.5.2 Language Bindings

There are Apache modules for most major scripting languages. These modules typically make it possible to write **Apache** modules entirely in a scripting language. They are also often used as a persistent interpreter embedded into the server that avoids the overhead of starting an external interpreter and the startup-time penalty for dynamic websites, as described in the next section.

29.7.6 Dynamic Websites

In the last decade, more businesses have turned to the Internet in order to enhance their revenue and increase exposure. This has also increased the need for interactive web content. While some companies, such as Microsoft, have introduced solutions into their proprietary products, the open source community answered the call. Modern options for dynamic web content include Django, Ruby on Rails, **mod_perl**, and **mod_php**.

29.7.6.1 Django

Django is a BSD licensed framework designed to allow developers to write high performance, elegant web applications quickly. It provides an object-relational mapper so that data types are developed as Python objects, and a rich dynamic database-access API is provided for those objects without the developer ever having to write SQL. It also provides an extensible template system so that the logic of the application is separated from the HTML presentation.

Django depends on **mod_python**, **Apache**, and an SQL database engine of your choice. The FreeBSD Port will install all of these pre-requisites for you with the appropriate flags.

Example 29-3. Installing Django with Apache2, mod_python3, and PostgreSQL

```
# cd /usr/ports/www/py-django; make all install clean -DWITH_MOD_PYTHON3 -DWITH_POSTGRESQL
```

Once Django and these pre-requisites are installed, you will need to create a Django project directory and then configure Apache to use the embedded Python interpreter to call your application for specific URLs on your site.

Example 29-4. Apache Configuration for Django/mod_python

You will need to add a line to the apache `httpd.conf` file to configure Apache to pass requests for certain URLs to your web application:

```
<Location "/">
    SetHandler python-program
    PythonPath "['/dir/to/your/django/packages/'] + sys.path"
    PythonHandler django.core.handlers.modpython
    SetEnv DJANGO_SETTINGS_MODULE mysite.settings
    PythonAutoReload On
    PythonDebug On
</Location>
```

29.7.6.2 Ruby on Rails

Ruby on Rails is another open source web framework that provides a full development stack and is optimized to make web developers more productive and capable of writing powerful applications quickly. It can be installed easily from the ports system.

```
# cd /usr/ports/www/rubygem-rails; make all install clean
```

29.7.6.3 mod_perl

The **Apache**/Perl integration project brings together the full power of the Perl programming language and the **Apache HTTP Server**. With the **mod_perl** module it is possible to write **Apache** modules entirely in Perl. In addition, the persistent interpreter embedded in the server avoids the overhead of starting an external interpreter and the penalty of Perl start-up time.

mod_perl is available a few different ways. To use **mod_perl** remember that **mod_perl** 1.0 only works with **Apache** 1.3 and **mod_perl** 2.0 only works with **Apache** 2.X. **mod_perl** 1.0 is available in `www/mod_perl` and a statically compiled version is available in `www/apache13-modperl`. **mod_perl** 2.0 is available in `www/mod_perl2`.

29.7.6.4 mod_php

PHP, also known as "PHP: Hypertext Preprocessor" is a general-purpose scripting language that is especially suited for Web development. Capable of being embedded into HTML its syntax draws upon C, Java, and Perl with the intention of allowing web developers to write dynamically generated webpages quickly.

To gain support for PHP5 for the **Apache** web server, begin by installing the `lang/php5` port.

If the `lang/php5` port is being installed for the first time, available `OPTIONS` will be displayed automatically. If a menu is not displayed, i.e. because the `lang/php5` port has been installed some time in the past, it is always possible to bring the options dialog up again by running:

```
# make config
```

in the port directory.

In the options dialog, check the `APACHE` option to build **mod_php5** as a loadable module for the **Apache** web server.

> **Note:** A lot of sites are still using PHP4 for various reasons (i.e. compatibility issues or already deployed web applications). If the **mod_php4** is needed instead of **mod_php5**, then please use the `lang/php4` port. The `lang/php4` port supports many of the configuration and build-time options of the `lang/php5` port.

This will install and configure the modules required to support dynamic PHP applications. Check to ensure the following sections have been added to `/usr/local/etc/apache/httpd.conf`:

```
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>
```

Once completed, a simple call to the `apachectl` command for a graceful restart is needed to load the PHP module:

```
# apachectl graceful
```

For future upgrades of PHP, the `make config` command will not be required; the selected `OPTIONS` are saved automatically by the FreeBSD Ports framework.

The PHP support in FreeBSD is extremely modular so the base install is very limited. It is very easy to add support using the `lang/php5-extensions` port. This port provides a menu driven interface to PHP extension installation. Alternatively, individual extensions can be installed using the appropriate port.

For instance, to add support for the **MySQL** database server to PHP5, simply install the port `databases/php5-mysql`.

After installing an extension, the **Apache** server must be reloaded to pick up the new configuration changes:

```
# apachectl graceful
```

29.8 File Transfer Protocol (FTP)

29.8.1 Overview

The File Transfer Protocol (FTP) provides users with a simple way to transfer files to and from an FTP server. FreeBSD includes FTP server software, **ftpd**, in the base system. This makes setting up and administering an FTP server on FreeBSD very straightforward.

29.8.2 Configuration

The most important configuration step is deciding which accounts will be allowed access to the FTP server. A normal FreeBSD system has a number of system accounts used for various daemons, but unknown users should not be allowed to log in with these accounts. The `/etc/ftpusers` file is a list of users disallowed any FTP access. By default, it includes the aforementioned system accounts, but it is possible to add specific users here that should not be allowed access to FTP.

You may want to restrict the access of some users without preventing them completely from using FTP. This can be accomplished with the `/etc/ftpchroot` file. This file lists users and groups subject to FTP access restrictions. The ftpchroot(5) manual page has all of the details so it will not be described in detail here.

If you would like to enable anonymous FTP access to your server, then you must create a user named `ftp` on your FreeBSD system. Users will then be able to log on to your FTP server with a username of `ftp` or `anonymous` and with any password (by convention an email address for the user should be used as the password). The FTP server will call chroot(2) when an anonymous user logs in, to restrict access to only the home directory of the `ftp` user.

There are two text files that specify welcome messages to be displayed to FTP clients. The contents of the file `/etc/ftpwelcome` will be displayed to users before they reach the login prompt. After a successful login, the contents of the file `/etc/ftpmotd` will be displayed. Note that the path to this file is relative to the login environment, so the file `~ftp/etc/ftpmotd` would be displayed for anonymous users.

Once the FTP server has been configured properly, it must be enabled in `/etc/inetd.conf`. All that is required here is to remove the comment symbol “#” from in front of the existing **ftpd** line :

```
ftp stream tcp nowait root /usr/libexec/ftpd ftpd -l
```

As explained in Example 29-1, the **inetd** configuration must be reloaded after this configuration file is changed. Please refer to Section 29.2.2 for details on enabling **inetd** on your system.

Alternatively, **ftpd** can also be started as a stand-alone server. In this case, it is sufficient to set the appropriate variable in `/etc/rc.conf`:

```
ftpd_enable="YES"
```

After setting the above variable, the stand-alone server will be started at the next reboot, or it can be started manually by executing the following command as `root`:

```
# /etc/rc.d/ftpd start
```

You can now log on to your FTP server by typing:

```
% ftp localhost
```

29.8.3 Maintaining

The **ftpd** daemon uses syslog(3) to log messages. By default, the system log daemon will put messages related to FTP in the `/var/log/xferlog` file. The location of the FTP log can be modified by changing the following line in `/etc/syslog.conf`:

```
ftp.info      /var/log/xferlog
```

Be aware of the potential problems involved with running an anonymous FTP server. In particular, you should think twice about allowing anonymous users to upload files. You may find that your FTP site becomes a forum for the trade of unlicensed commercial software or worse. If you do need to allow anonymous FTP uploads, then you should set up the permissions so that these files can not be read by other anonymous users until they have been reviewed.

29.9 File and Print Services for Microsoft Windows clients (Samba)

29.9.1 Overview

Samba is a popular open source software package that provides file and print services for Microsoft Windows clients. Such clients can connect to and use FreeBSD filespace as if it was a local disk drive, or FreeBSD printers as if they were local printers.

Samba software packages should be included on your FreeBSD installation media. If you did not install **Samba** when you first installed FreeBSD, then you can install it from the `net/samba34` port or package.

29.9.2 Configuration

A default **Samba** configuration file is installed as `/usr/local/share/examples/samba34/smb.conf.default`. This file must be copied to `/usr/local/etc/smb.conf` and customized before **Samba** can be used.

The `smb.conf` file contains runtime configuration information for **Samba**, such as definitions of the printers and "file system shares" that you would like to share with Windows clients. The **Samba** package includes a web based tool called **swat** which provides a simple way of configuring the `smb.conf` file.

29.9.2.1 Using the Samba Web Administration Tool (SWAT)

The Samba Web Administration Tool (SWAT) runs as a daemon from **inetd**. Therefore, the following line in `/etc/inetd.conf` should be uncommented before **swat** can be used to configure **Samba**:

```
swat    stream  tcp     nowait/400      root    /usr/local/sbin/swat    swat
```

As explained in Example 29-1, the **inetd** configuration must be reloaded after this configuration file is changed.

Once **swat** has been enabled in `inetd.conf`, you can use a browser to connect to http://localhost:901. You will first have to log on with the system `root` account.

Once you have successfully logged on to the main **Samba** configuration page, you can browse the system documentation, or begin by clicking on the Globals tab. The Globals section corresponds to the variables that are set in the `[global]` section of `/usr/local/etc/smb.conf`.

29.9.2.2 Global Settings

Whether you are using **swat** or editing `/usr/local/etc/smb.conf` directly, the first directives you are likely to encounter when configuring **Samba** are:

`workgroup`

NT Domain-Name or Workgroup-Name for the computers that will be accessing this server.

`netbios name`

This sets the NetBIOS name by which a **Samba** server is known. By default it is the same as the first component of the host's DNS name.

`server string`

This sets the string that will be displayed with the `net view` command and some other networking tools that seek to display descriptive text about the server.

29.9.2.3 Security Settings

Two of the most important settings in `/usr/local/etc/smb.conf` are the security model chosen, and the backend password format for client users. The following directives control these options:

`security`

The two most common options here are `security = share` and `security = user`. If your clients use usernames that are the same as their usernames on your FreeBSD machine then you will want to use user level security. This is the default security policy and it requires clients to first log on before they can access shared resources.

In share level security, client do not need to log onto the server with a valid username and password before attempting to connect to a shared resource. This was the default security model for older versions of **Samba**.

`passdb backend`

Samba has several different backend authentication models. You can authenticate clients with LDAP, NIS+, a SQL database, or a modified password file. The default authentication method is `smbpasswd`, and that is all that will be covered here.

Assuming that the default `smbpasswd` backend is used, the `/usr/local/etc/samba/smbpasswd` file must be created to allow **Samba** to authenticate clients. If you would like to give your UNIX user accounts access from Windows clients, use the following command:

```
# smbpasswd -a username
```

Note: The recommended backend is now `tdbsam`, and the following command should be used to add user accounts:

```
# pdbedit -a -u username
```

Please see the Official Samba HOWTO (http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/) for additional information about configuration options. With the basics outlined here, you should have everything you need to start running **Samba**.

29.9.3 Starting Samba

The `net/samba34` port adds a new startup script, which can be used to control **Samba**. To enable this script, so that it can be used for example to start, stop or restart **Samba**, add the following line to the `/etc/rc.conf` file:

```
samba_enable="YES"
```

Or, for fine grain control:

```
nmbd_enable="YES"
```

```
smbd_enable="YES"
```

> **Note:** This will also configure **Samba** to automatically start at system boot time.

It is possible then to start **Samba** at any time by typing:

```
# /usr/local/etc/rc.d/samba start
Starting SAMBA: removing stale tdbs :
Starting nmbd.
Starting smbd.
```

Please refer to Section 11.7 for more information about using rc scripts.

Samba actually consists of three separate daemons. You should see that both the **nmbd** and **smbd** daemons are started by the `samba` script. If you enabled winbind name resolution services in `smb.conf`, then you will also see that the **winbindd** daemon is started.

You can stop **Samba** at any time by typing :

```
# /usr/local/etc/rc.d/samba stop
```

Samba is a complex software suite with functionality that allows broad integration with Microsoft Windows networks. For more information about functionality beyond the basic installation described here, please see http://www.samba.org.

29.10 Clock Synchronization with NTP

29.10.1 Overview

Over time, a computer’s clock is prone to drift. The Network Time Protocol (NTP) is one way to ensure your clock stays accurate.

Many Internet services rely on, or greatly benefit from, computers' clocks being accurate. For example, a web server may receive requests to send a file if it has been modified since a certain time. In a local area network environment, it is essential that computers sharing files from the same file server have synchronized clocks so that file timestamps stay consistent. Services such as cron(8) also rely on an accurate system clock to run commands at the specified times.

FreeBSD ships with the ntpd(8) NTP server which can be used to query other NTP servers to set the clock on your machine or provide time services to others.

29.10.2 Choosing Appropriate NTP Servers

In order to synchronize your clock, you will need to find one or more NTP servers to use. Your network administrator or ISP may have set up an NTP server for this purpose—check their documentation to see if this is the case. There is an online list of publicly accessible NTP servers (http://ntp.isc.org/bin/view/Servers/WebHome) which you can use to find an NTP server near to you. Make sure you are aware of the policy for any servers you choose, and ask for permission if required.

Choosing several unconnected NTP servers is a good idea in case one of the servers you are using becomes unreachable or its clock is unreliable. ntpd(8) uses the responses it receives from other servers intelligently—it will favor unreliable servers less than reliable ones.

29.10.3 Configuring Your Machine

29.10.3.1 Basic Configuration

If you only wish to synchronize your clock when the machine boots up, you can use ntpdate(8). This may be appropriate for some desktop machines which are frequently rebooted and only require infrequent synchronization, but most machines should run ntpd(8).

Using ntpdate(8) at boot time is also a good idea for machines that run ntpd(8). The ntpd(8) program changes the clock gradually, whereas ntpdate(8) sets the clock, no matter how great the difference between a machine's current clock setting and the correct time.

To enable ntpdate(8) at boot time, add `ntpdate_enable="YES"` to `/etc/rc.conf`. You will also need to specify all servers you wish to synchronize with and any flags to be passed to ntpdate(8) in `ntpdate_flags`.

29.10.3.2 General Configuration

NTP is configured by the `/etc/ntp.conf` file in the format described in ntp.conf(5). Here is a simple example:

```
server ntplocal.example.com prefer
server timeserver.example.org
server ntp2a.example.net

driftfile /var/db/ntp.drift
```

The `server` option specifies which servers are to be used, with one server listed on each line. If a server is specified with the `prefer` argument, as with `ntplocal.example.com`, that server is preferred over other servers. A response from a preferred server will be discarded if it differs significantly from other servers' responses, otherwise

it will be used without any consideration to other responses. The `prefer` argument is normally used for NTP servers that are known to be highly accurate, such as those with special time monitoring hardware.

The `driftfile` option specifies which file is used to store the system clock's frequency offset. The ntpd(8) program uses this to automatically compensate for the clock's natural drift, allowing it to maintain a reasonably correct setting even if it is cut off from all external time sources for a period of time.

The `driftfile` option specifies which file is used to store information about previous responses from the NTP servers you are using. This file contains internal information for NTP. It should not be modified by any other process.

29.10.3.3 Controlling Access to Your Server

By default, your NTP server will be accessible to all hosts on the Internet. The `restrict` option in `/etc/ntp.conf` allows you to control which machines can access your server.

If you want to deny all machines from accessing your NTP server, add the following line to `/etc/ntp.conf`:

```
restrict default ignore
```

> **Note:** This will also prevent access from your server to any servers listed in your local configuration. If you need to synchronise your NTP server with an external NTP server you should allow the specific server. See the ntp.conf(5) manual for more information.

If you only want to allow machines within your own network to synchronize their clocks with your server, but ensure they are not allowed to configure the server or used as peers to synchronize against, add

```
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

instead, where `192.168.1.0` is an IP address on your network and `255.255.255.0` is your network's netmask.

`/etc/ntp.conf` can contain multiple `restrict` options. For more details, see the `Access Control Support` subsection of ntp.conf(5).

29.10.4 Running the NTP Server

To ensure the NTP server is started at boot time, add the line `ntpd_enable="YES"` to `/etc/rc.conf`. If you wish to pass additional flags to ntpd(8), edit the `ntpd_flags` parameter in `/etc/rc.conf`.

To start the server without rebooting your machine, run `ntpd` being sure to specify any additional parameters from `ntpd_flags` in `/etc/rc.conf`. For example:

```
# ntpd -p /var/run/ntpd.pid
```

29.10.5 Using ntpd with a Temporary Internet Connection

The ntpd(8) program does not need a permanent connection to the Internet to function properly. However, if you have a temporary connection that is configured to dial out on demand, it is a good idea to prevent NTP traffic from

triggering a dial out or keeping the connection alive. If you are using user PPP, you can use `filter` directives in `/etc/ppp/ppp.conf`. For example:

```
set filter dial 0 deny udp src eq 123
# Prevent NTP traffic from initiating dial out
set filter dial 1 permit 0 0
set filter alive 0 deny udp src eq 123
# Prevent incoming NTP traffic from keeping the connection open
set filter alive 1 deny udp dst eq 123
# Prevent outgoing NTP traffic from keeping the connection open
set filter alive 2 permit 0/0 0/0
```

For more details see the `PACKET FILTERING` section in ppp(8) and the examples in `/usr/share/examples/ppp/`.

> **Note:** Some Internet access providers block low-numbered ports, preventing NTP from functioning since replies never reach your machine.

29.10.6 Further Information

Documentation for the NTP server can be found in `/usr/share/doc/ntp/` in HTML format.

29.11 Remote Host Logging with `syslogd`

Interacting with system logs is a crucial aspect of both security and system administration. Monitoring the log files of multiple hosts can get very unwieldy when these hosts are distributed across medium or large networks, or when they are parts of various different types of networks. In these cases, configuring remote logging may make the whole process a lot more comfortable.

Centralized logging to a specific logging host can reduce some of the administrative burden of log file administration. Log file aggregation, merging and rotation can be configured in one location, using the native tools of FreeBSD, such as syslogd(8) and newsyslog(8). In the following example configuration, host `A`, named `logserv.example.com`, will collect logging information for the local network. Host `B`, named `logclient.example.com` will pass logging information to the server system. In live configurations, both hosts require proper forward and reverse DNS or entries in `/etc/hosts`. Otherwise, data will be rejected by the server.

29.11.1 Log Server Configuration

Log servers are machines configured to accept logging information from remote hosts. In most cases this is to ease configuration, in other cases it may just be a better administration move. Regardless of reason, there are a few requirements before continuing.

A properly configured logging server has met the following minimal requirements:

- The firewall ruleset allows for UDP to be passed on port 514 on both the client and server;
- syslogd has been configured to accept remote messages from client machines;

- The syslogd server and all client machines must have valid entries for both forward and reverse DNS, or be properly configured in `/etc/hosts`.

To configure the log server, the client must be listed in `/etc/syslog.conf`, and the logging facility must be specified:

```
+logclient.example.com
*.*     /var/log/logclient.log
```

> **Note:** More information on various supported and available *facilities* may be found in the syslog.conf(5) manual page.

Once added, all `facility` messages will be logged to the file specified previously, `/var/log/logclient.log`.

The server machine must also have the following listing placed inside `/etc/rc.conf`:

```
syslogd_enable="YES"
syslogd_flags="-a logclient.example.com -vv"
```

The first option will enable the `syslogd` daemon on boot up, and the second option allows data from the specified client to be accepted on this server. The latter part, using `-vv`, will increase the verbosity of logged messages. This is extremely useful for tweaking facilities as administrators are able to see what type of messages are being logged under which facility.

Multiple `-a` options may be specified to allow logging from multiple clients. IP addresses and whole netblocks may also be specified, see the syslog(3) manual page for a full list of possible options.

Finally, the log file should be created. The method used does not matter, but touch(1) works great for situations such as this:

```
# touch /var/log/logclient.log
```

At this point, the `syslogd` daemon should be restarted and verified:

```
# /etc/rc.d/syslogd restart
# pgrep syslog
```

If a PID is returned, the server has been restarted successfully, and client configuration may begin. If the server has not restarted, consult the `/var/log/messages` log for any output.

29.11.2 Log Client Configuration

A logging client is a machine which sends log information to a logging server in addition to keeping local copies.

Similar to log servers, clients must also meet a few minimum requirements:

- syslogd(8) must be configured to send messages of specific types to a log server, which must accept them;
- The firewall must allow UDP packets through on port 514;
- Both forward and reverse DNS must be configured or have proper entries in the `/etc/hosts`.

Client configuration is a bit more relaxed when compared to that of the servers. The client machine must have the following listing placed inside `/etc/rc.conf`:

```
syslogd_enable="YES"
syslogd_flags="-s -vv"
```

As before, these entries will enable the `syslogd` daemon on boot up, and increases the verbosity of logged messages. The `-s` option prevents logs from being accepted by this client from other hosts.

Facilities describe the system part for which a message is generated. For an example, ftp and ipfw are both facilities. When log messages are generated for those two services, they will normally include those two utilities in any log messages. Facilities are accompanied with a priority or level, which is used to mark how important a log message is. The most common will be the `warning` and `info`. Please refer to the syslog(3) manual page for a full list of available facilities and priorities.

The logging server must be defined in the client's `/etc/syslog.conf`. In this instance, the `@` symbol is used to send logging data to a remote server and would look similar to the following entry:

```
*.*  @logserv.example.com
```

Once added, `syslogd` must be restarted for the changes to take effect:

```
# /etc/rc.d/syslogd restart
```

To test that log messages are being sent across the network, use logger(1) on the client to send a message to `syslogd`:

```
# logger "Test message from logclient"
```

This message should now exist both in `/var/log/messages` on the client, and `/var/log/logclient.log` on the log server.

29.11.3 Debugging Log Servers

In certain cases, debugging may be required if messages are not being received on the log server. There are several reasons this may occur; however, the most common two are network connection issues and DNS issues. To test these cases, ensure both hosts are able to reach one another using the hostname specified in `/etc/rc.conf`. If this appears to be working properly, an alternation to the `syslogd_flags` option in `/etc/rc.conf` will be required.

In the following example, `/var/log/logclient.log` is empty, and the `/var/log/messages` files indicate no reason for the failure. To increase debugging output, change the `syslogd_flags` option to look like the following example, and issue a restart:

```
syslogd_flags="-d -a logclien.example.com -vv"
```

```
# /etc/rc.d/syslogd restart
```

Debugging data similar to the following will flash on the screen immediately after the restart:

```
logmsg: pri 56, flags 4, from logserv.example.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from logserv.example.com, msg syslogd: kernel boot file is /boot/kernel/k
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 logclient.example.com;
rejected in rule 0 due to name mismatch.
```

It appears obvious the messages are being rejected due to a name mismatch. After reviewing the configuration bit by bit, it appears a typo in the following `/etc/rc.conf` line has an issue:

```
syslogd_flags="-d -a logclien.example.com -vv"
```

The line should contain `logclient`, not `logclien`. After the proper alterations are made, a restart is issued with expected results:

```
# /etc/rc.d/syslogd restart
logmsg: pri 56, flags 4, from logserv.example.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from logserv.example.com, msg syslogd: kernel boot file is /boot/kernel/k
syslogd: kernel boot file is /boot/kernel/kernel
logmsg: pri 166, flags 17, from logserv.example.com,
msg Dec 10 20:55:02 <syslog.err> logserv.example.com syslogd: exiting on signal 2
cvthname(192.168.1.10)
validate: dgram from IP 192.168.1.10, port 514, name logclient.example.com;
accepted in rule 0.
logmsg: pri 15, flags 0, from logclient.example.com, msg Dec 11 02:01:28 trhodes: Test message 2
Logging to FILE /var/log/logclient.log
Logging to FILE /var/log/messages
```

At this point, the messages are being properly received and placed in the correct file.

29.11.4 Security Considerations

As with any network service, security requirements should be considered before implementing this configuration. At times, log files may contain sensitive data about services enabled on the local host, user accounts, and configuration data. Network data sent from the client to the server will not be encrypted nor password protected. If a need for encryption exists, it might be possible to use `security/stunnel`, which will transmit data over an encrypted tunnel.

Local security is also an issue. Log files are not encrypted during use or after log rotation. Local users may access these files to gain additional insight on system configuration. In those cases, setting proper permissions on these files will be critical. The newsyslog(8) utility supports setting permissions on newly created and rotated log files. Setting log files to mode `600` should prevent any unwanted snooping by local users.

Chapter 30
Firewalls

30.1 Introduction

Firewalls make it possible to filter incoming and outgoing traffic that flows through your system. A firewall can use one or more sets of “rules” to inspect the network packets as they come in or go out of your network connections and either allows the traffic through or blocks it. The rules of a firewall can inspect one or more characteristics of the packets, including but not limited to the protocol type, the source or destination host address, and the source or destination port.

Firewalls can greatly enhance the security of a host or a network. They can be used to do one or more of the following things:

- To protect and insulate the applications, services and machines of your internal network from unwanted traffic coming in from the public Internet.
- To limit or disable access from hosts of the internal network to services of the public Internet.
- To support network address translation (NAT), which allows your internal network to use private IP addresses and share a single connection to the public Internet (either with a single IP address or by a shared pool of automatically assigned public addresses).

After reading this chapter, you will know:

- How to properly define packet filtering rules.
- The differences between the firewalls built into FreeBSD.
- How to use and configure the OpenBSD **PF** firewall.
- How to use and configure **IPFILTER**.
- How to use and configure **IPFW**.

Before reading this chapter, you should:

- Understand basic FreeBSD and Internet concepts.

30.2 Firewall Concepts

There are two basic ways to create firewall rulesets: “inclusive” or “exclusive”. An exclusive firewall allows all traffic through except for the traffic matching the ruleset. An inclusive firewall does the reverse. It only allows traffic matching the rules through and blocks everything else.

An inclusive firewall offers much better control of the outgoing traffic, making it a better choice for systems that offer services to the public Internet. It also controls the type of traffic originating from the public Internet that can gain access to your private network. All traffic that does not match the rules, is blocked and logged by design. Inclusive firewalls are generally safer than exclusive firewalls because they significantly reduce the risk of allowing unwanted traffic to pass through them.

> **Note:** Unless noted otherwise, all configuration and example rulesets in this chapter, create inclusive type firewalls.

Security can be tightened further using a “stateful firewall”. This type of firewall keeps track of which connections are opened through the firewall and will only allow traffic through which either matches an existing connection or opens a new one. The disadvantage of a stateful firewall is that it can be vulnerable to Denial of Service (DoS) attacks if a lot of new connections are opened very fast. With most firewalls it is possible to use a combination of stateful and non-stateful behavior to make an optimal firewall for the site.

30.3 Firewall Packages

FreeBSD has three different firewall packages built into the base system. They are: *IPFILTER* (also known as IPF), *IPFIREWALL* (also known as IPFW), and *OpenBSD’s PacketFilter* (also known as PF). FreeBSD also has two built in packages for traffic shaping (basically controlling bandwidth usage): altq(4) and dummynet(4). Dummynet has traditionally been closely tied with IPFW, and ALTQ with PF. Traffic shaping for IPFILTER can currently be done with IPFILTER for NAT and filtering and IPFW with dummynet(4) *or* by using PF with ALTQ. IPFW, and PF all use rules to control the access of packets to and from your system, although they go about it different ways and have a different rule syntax.

The reason that FreeBSD has multiple built in firewall packages is that different people have different requirements and preferences. No single firewall package is the best.

The author prefers IPFILTER because its stateful rules are much less complicated to use in a NAT environment and it has a built in ftp proxy that simplifies the rules to allow secure outbound FTP usage.

Since all firewalls are based on inspecting the values of selected packet control fields, the creator of the firewall rulesets must have an understanding of how TCP/IP works, what the different values in the packet control fields are and how these values are used in a normal session conversation. For a good explanation go to: http://www.ipprimer.com/overview.cfm.

30.4 The OpenBSD Packet Filter (PF) and ALTQ

As of July 2003 the OpenBSD firewall software application known as PF was ported to FreeBSD and made available in the FreeBSD Ports Collection. Released in 2004, FreeBSD 5.3 was the first release that contained PF as an integrated part of the base system. PF is a complete, full-featured firewall that has optional support for ALTQ (Alternate Queuing). ALTQ provides Quality of Service (QoS) functionality.

The OpenBSD Project does an outstanding job of maintaining the PF FAQ (http://www.openbsd.org/faq/pf/). As such, this section of the Handbook will focus on PF as it pertains to FreeBSD while providing some general information regarding usage. For detailed usage information please refer to the PF FAQ (http://www.openbsd.org/faq/pf/).

More information about PF for FreeBSD can be found at http://pf4freebsd.love2party.net/.

30.4.1 Using the PF loadable kernel modules

To load the PF Kernel Module add the following line to `/etc/rc.conf`:

```
pf_enable="YES"
```

Then run the startup script to load the module:

```
# /etc/rc.d/pf start
```

Note that the PF Module will not load if it cannot find the ruleset config file. The default location is `/etc/pf.conf`. If the PF ruleset is located somewhere else, PF can be instructed to look there by adding a line like the following to `/etc/rc.conf`:

```
pf_rules="/path/to/pf.conf"
```

The sample `pf.conf` can be found in `/usr/share/examples/pf/`.

The PF module can also be loaded manually from the command line:

```
# kldload pf.ko
```

Logging support for PF is provided by the `pflog.ko` and can be loaded by adding the following line to `/etc/rc.conf`:

```
pflog_enable="YES"
```

Then run the startup script to load the module:

```
# /etc/rc.d/pflog start
```

If you need other PF features you will need to compile PF support into the kernel.

30.4.2 PF kernel options

While it is not necessary that you compile PF support into the FreeBSD kernel, you may want to do so to take advantage of one of PF's advanced features that is not included in the loadable module, namely pfsync(4), which is a pseudo-device that exposes certain changes to the state table used by PF. It can be paired with carp(4) to create failover firewalls using PF. More information on CARP can be found in Section 31.13 of the Handbook.

The PF kernel options can be found in `/usr/src/sys/conf/NOTES` and are reproduced below:

```
device pf
device pflog
device pfsync
```

The `device pf` option enables support for the "Packet Filter" firewall (pf(4)).

The `device pflog` option enables the optional pflog(4) pseudo network device which can be used to log traffic to a bpf(4) descriptor. The pflogd(8) daemon can be used to store the logging information to disk.

The `device pfsync` option enables the optional pfsync(4) pseudo-network device that is used to monitor "state changes".

30.4.3 Available rc.conf Options

The following rc.conf(5) statements configure PF and pflog(4) at boot:

```
pf_enable="YES"                 # Enable PF (load module if required)
pf_rules="/etc/pf.conf"         # rules definition file for pf
pf_flags=""                     # additional flags for pfctl startup
pflog_enable="YES"              # start pflogd(8)
pflog_logfile="/var/log/pflog"  # where pflogd should store the logfile
pflog_flags=""                  # additional flags for pflogd startup
```

If you have a LAN behind this firewall and have to forward packets for the computers on the LAN or want to do NAT, you will need the following option as well:

```
gateway_enable="YES"            # Enable as LAN gateway
```

30.4.4 Creating Filtering Rules

PF reads its configuration rules from pf.conf(5) (`/etc/pf.conf` by default) and it modifies, drops, or passes packets according to the rules or definitions specified there. The FreeBSD installation includes several sample files located in `/usr/share/examples/pf/`. Please refer to the PF FAQ (http://www.openbsd.org/faq/pf/) for complete coverage of PF rulesets.

> **Warning:** When browsing the PF FAQ (http://www.openbsd.org/faq/pf/), please keep in mind that different versions of FreeBSD can contain different versions of PF. Currently, FreeBSD is using the same version of PF as OpenBSD 4.1.

The FreeBSD packet filter mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf) is a good place to ask questions about configuring and running the PF firewall. Do not forget to check the mailing list archives before asking questions!

30.4.5 Working with PF

Use pfctl(8) to control PF. Below are some useful commands (be sure to review the pfctl(8) man page for all available options):

Command	Purpose
`pfctl -e`	Enable PF
`pfctl -d`	Disable PF
`pfctl -F all -f /etc/pf.conf`	Flush all rules (nat, filter, state, table, etc.) and reload from the file `/etc/pf.conf`
`pfctl -s [ rules | nat | state ]`	Report on the filter rules, nat rules, or state table

Command	Purpose
`pfctl -vnf /etc/pf.conf`	Check `/etc/pf.conf` for errors, but do not load ruleset

30.4.6 Enabling ALTQ

ALTQ is only available by compiling support for it into the FreeBSD kernel. ALTQ is not supported by all of the available network card drivers. Please see the altq(4) manual page for a list of drivers that are supported in your release of FreeBSD.

The following kernel options will enable ALTQ and add additional functionality:

```
options         ALTQ
options         ALTQ_CBQ        # Class Bases Queuing (CBQ)
options         ALTQ_RED        # Random Early Detection (RED)
options         ALTQ_RIO        # RED In/Out
options         ALTQ_HFSC       # Hierarchical Packet Scheduler (HFSC)
options         ALTQ_PRIQ       # Priority Queuing (PRIQ)
options         ALTQ_NOPCC      # Required for SMP build
```

`options ALTQ` enables the ALTQ framework.

`options ALTQ_CBQ` enables *Class Based Queuing* (CBQ). CBQ allows you to divide a connection's bandwidth into different classes or queues to prioritize traffic based on filter rules.

`options ALTQ_RED` enables *Random Early Detection* (RED). RED is used to avoid network congestion. RED does this by measuring the length of the queue and comparing it to the minimum and maximum thresholds for the queue. If the queue is over the maximum all new packets will be dropped. True to its name, RED drops packets from different connections randomly.

`options ALTQ_RIO` enables *Random Early Detection In and Out*.

`options ALTQ_HFSC` enables the *Hierarchical Fair Service Curve Packet Scheduler*. For more information about HFSC see: http://www-2.cs.cmu.edu/~hzhang/HFSC/main.html.

`options ALTQ_PRIQ` enables *Priority Queuing* (PRIQ). PRIQ will always pass traffic that is in a higher queue first.

`options ALTQ_NOPCC` enables SMP support for ALTQ. This option is required on SMP systems.

30.5 The IPFILTER (IPF) Firewall

The author of IPFILTER is Darren Reed. IPFILTER is not operating system dependent: it is an open source application and has been ported to FreeBSD, NetBSD, OpenBSD, SunOS, HP/UX, and Solaris operating systems. IPFILTER is actively being supported and maintained, with updated versions being released regularly.

IPFILTER is based on a kernel-side firewall and NAT mechanism that can be controlled and monitored by userland interface programs. The firewall rules can be set or deleted with the ipf(8) utility. The NAT rules can be set or deleted with the ipnat(1) utility. The ipfstat(8) utility can print run-time statistics for the kernel parts of IPFILTER. The ipmon(8) program can log IPFILTER actions to the system log files.

IPF was originally written using a rule processing logic of "the last matching rule wins" and used only stateless type of rules. Over time IPF has been enhanced to include a "quick" option and a stateful "keep state" option which

drastically modernized the rules processing logic. IPF's official documentation covers only the legacy rule coding parameters and rule file processing logic. The modernized functions are only included as additional options, completely understating their benefits in producing a far superior and more secure firewall.

The instructions contained in this section are based on using rules that contain the "quick" option and the stateful "keep state" option. This is the basic framework for coding an inclusive firewall ruleset.

For detailed explanation of the legacy rules processing method see:
http://www.obfuscation.org/ipf/ipf-howto.html#TOC_1 and http://coombs.anu.edu.au/~avalon/ip-filter.html.

The IPF FAQ is at http://www.phildev.net/ipf/index.html.

A searchable archive of the open-source IPFilter mailing list is available at http://marc.theaimsgroup.com/?l=ipfilter.

30.5.1 Enabling IPF

IPF is included in the basic FreeBSD install as a separate run time loadable module. The system will dynamically load the IPF kernel loadable module when the `rc.conf` statement `ipfilter_enable="YES"` is used. The loadable module was created with logging enabled and the `default pass all` options. There is no need to compile IPF into the FreeBSD kernel just to change the default to `block all`. This can be done just by adding a `block all` rule at the end of your ruleset.

30.5.2 Kernel options

It is not a mandatory requirement to enable IPF by compiling the following options into the FreeBSD kernel. It is only presented here as background information. Compiling IPF into the kernel causes the loadable module to never be used.

Sample kernel config IPF option statements are in the `/usr/src/sys/conf/NOTES` kernel source and are reproduced here:

```
options IPFILTER
options IPFILTER_LOG
options IPFILTER_DEFAULT_BLOCK
```

`options IPFILTER` enables support for the "IPFILTER" firewall.

`options IPFILTER_LOG` enables the option to have IPF log traffic by writing to the `ipl` packet logging pseudo—device for every rule that has the `log` keyword.

`options IPFILTER_DEFAULT_BLOCK` changes the default behavior so any packet not matching a firewall `pass` rule gets blocked.

These settings will take effect only after installing a kernel that has been built with the above options set.

30.5.3 Available rc.conf Options

To activate IPF at boot time, the following statements need to be added to `/etc/rc.conf`:

```
ipfilter_enable="YES"             # Start ipf firewall
ipfilter_rules="/etc/ipf.rules"   # loads rules definition text file
ipmon_enable="YES"                # Start IP monitor log
ipmon_flags="-Ds"                 # D = start as daemon
```

```
                                # s = log to syslog
                                # v = log tcp window, ack, seq
                                # n = map IP & port to names
```

If there is a LAN behind this firewall that uses the reserved private IP address ranges, the following lines will have to be added to enable NAT functionality:

```
gateway_enable="YES"              # Enable as LAN gateway
ipnat_enable="YES"                # Start ipnat function
ipnat_rules="/etc/ipnat.rules"    # rules definition file for ipnat
```

30.5.4 IPF

The ipf(8) command is used to load your ruleset file. Your custom rules would normally be placed in a file, and the following command could then be used to replace in mass the currently running firewall rules:

```
# ipf -Fa -f /etc/ipf.rules
```

`-Fa` means flush all internal rules tables.

`-f` means this is the file to read for the rules to load.

This gives you the ability to make changes to your custom rules file, run the above IPF command, and thus update the running firewall with a fresh copy of all the rules without having to reboot the system. This method is very convenient for testing new rules as the procedure can be executed as many times as needed.

See the ipf(8) manual page for details on the other flags available with this command.

The ipf(8) command expects the rules file to be a standard text file. It will not accept a rules file written as a script with symbolic substitution.

There is a way to build IPF rules that utilizes the power of script symbolic substitution. For more information, see Section 30.5.9.

30.5.5 IPFSTAT

The default behavior of ipfstat(8) is to retrieve and display the totals of the accumulated statistics gathered as a result of applying the user coded rules against packets going in and out of the firewall since it was last started, or since the last time the accumulators were reset to zero by the `ipf -Z` command.

See the ipfstat(8) manual page for details.

The default ipfstat(8) command output will look something like this:

```
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)
```

When supplied with either `-i` for inbound or `-o` for outbound, the command will retrieve and display the appropriate list of filter rules currently installed and in use by the kernel.

`ipfstat -in` displays the inbound internal rules table with rule number.

`ipfstat -on` displays the outbound internal rules table with the rule number.

The output will look something like this:

```
@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
```

`ipfstat -ih` displays the inbound internal rules table, prefixing each rule with a count of how many times the rule was matched.

`ipfstat -oh` displays the outbound internal rules table, prefixing each rule with a count of how many times the rule was matched.

The output will look something like this:

```
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
```

One of the most important functions of the `ipfstat` command is the `-t` flag which displays the state table in a way similar to the way top(1) shows the FreeBSD running process table. When your firewall is under attack, this function gives you the ability to identify, drill down to, and see the attacking packets. The optional sub-flags give the ability to select the destination or source IP, port, or protocol that you want to monitor in real time. See the ipfstat(8) manual page for details.

30.5.6 IPMON

In order for `ipmon` to work properly, the kernel option `IPFILTER_LOG` must be turned on. This command has two different modes that it can be used in. Native mode is the default mode when the command is typed on the command line without the `-D` flag.

Daemon mode is for when a continuous system log file is desired, so that logging of past events may be reviewed. This is how FreeBSD and IPFILTER are configured to work together. FreeBSD has a built in facility to automatically rotate system logs. That is why outputting the log information to syslogd(8) is better than the default of outputting to a regular file. In the default `rc.conf` file, the `ipmon_flags` statement uses the `-Ds` flags:

```
ipmon_flags="-Ds" # D = start as daemon
                  # s = log to syslog
                  # v = log tcp window, ack, seq
```

```
# n = map IP & port to names
```

The benefits of logging are obvious. It provides the ability to review, after the fact, information such as which packets had been dropped, what addresses they came from and where they were going. These can all provide a significant edge in tracking down attackers.

Even with the logging facility enabled, IPF will not generate any rule logging on its own. The firewall administrator decides what rules in the ruleset he wants to log and adds the log keyword to those rules. Normally only deny rules are logged.

It is very customary to include a default deny everything rule with the log keyword included as your last rule in the ruleset. This makes it possible to see all the packets that did not match any of the rules in the ruleset.

30.5.7 IPMON Logging

Syslogd uses its own special method for segregation of log data. It uses special groupings called “facility” and “level”. IPMON in `-Ds` mode uses `local0` as the “facility” name by default. The following levels can be used to further segregate the logged data if desired:

```
LOG_INFO - packets logged using the "log" keyword as the action rather than pass or block.
LOG_NOTICE - packets logged which are also passed
LOG_WARNING - packets logged which are also blocked
LOG_ERR - packets which have been logged and which can be considered short
```

To setup IPFILTER to log all data to `/var/log/ipfilter.log`, the file will need to be created beforehand. The following command will do that:

```
# touch /var/log/ipfilter.log
```

The syslogd(8) function is controlled by definition statements in the `/etc/syslog.conf` file. The `syslog.conf` file offers considerable flexibility in how **syslog** will deal with system messages issued by software applications like IPF.

Add the following statement to `/etc/syslog.conf`:

```
local0.* /var/log/ipfilter.log
```

The `local0.*` means to write all the logged messages to the coded file location.

To activate the changes to `/etc/syslog.conf` you can reboot or bump the syslogd(8) daemon into re-reading `/etc/syslog.conf` by running `/etc/rc.d/syslogd reload`

Do not forget to change `/etc/newsyslog.conf` to rotate the new log created above.

30.5.8 The Format of Logged Messages

Messages generated by `ipmon` consist of data fields separated by white space. Fields common to all messages are:

1. The date of packet receipt.
2. The time of packet receipt. This is in the form HH:MM:SS.F, for hours, minutes, seconds, and fractions of a second (which can be several digits long).

3. The name of the interface the packet was processed on, e.g. `dc0`.
4. The group and rule number of the rule, e.g. `@0:17`.

These can be viewed with `ipfstat -in`.

1. The action: p for passed, b for blocked, S for a short packet, n did not match any rules, L for a log rule. The order of precedence in showing flags is: S, p, b, n, L. A capital P or B means that the packet has been logged due to a global logging setting, not a particular rule.
2. The addresses. This is actually three fields: the source address and port (separated by a comma), the -> symbol, and the destination address and port, e.g.: `209.53.17.22,80 -> 198.73.220.17,1722`.
3. `PR` followed by the protocol name or number, e.g.: `PR tcp`.
4. `len` followed by the header length and total length of the packet, e.g.: `len 20 40`.

If the packet is a TCP packet, there will be an additional field starting with a hyphen followed by letters corresponding to any flags that were set. See the ipf(5) manual page for a list of letters and their flags.

If the packet is an ICMP packet, there will be two fields at the end, the first always being “ICMP”, and the next being the ICMP message and sub-message type, separated by a slash, e.g. ICMP 3/3 for a port unreachable message.

30.5.9 Building the Rule Script with Symbolic Substitution

Some experienced IPF users create a file containing the rules and code them in a manner compatible with running them as a script with symbolic substitution. The major benefit of doing this is that only the value associated with the symbolic name needs to be changed, and when the script is run all the rules containing the symbolic name will have the value substituted in the rules. Being a script, symbolic substitution can be used to code frequently used values and substitute them in multiple rules. This can be seen in the following example.

The script syntax used here is compatible with the sh(1), csh(1), and tcsh(1) shells.

Symbolic substitution fields are prefixed with a dollar sign: `$`.

Symbolic fields do not have the $ prefix.

The value to populate the symbolic field must be enclosed with double quotes (").

Start your rule file with something like this:

```
############# Start of IPF rules script ########################

oif="dc0"            # name of the outbound interface
odns="192.0.2.11"    # ISP’s DNS server IP address
myip="192.0.2.7"     # my static IP address from ISP
ks="keep state"
fks="flags S keep state"

# You can choose between building /etc/ipf.rules file
# from this script or running this script "as is".
#
# Uncomment only one line and comment out another.
#
# 1) This can be used for building /etc/ipf.rules:
#cat > /etc/ipf.rules << EOF
```

```
#
# 2) This can be used to run script "as is":
/sbin/ipf -Fa -f - << EOF

# Allow out access to my ISP's Domain name server.
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

# Allow out non-secure standard www function
pass out quick on $oif proto tcp from $myip to any port = 80 $fks

# Allow out secure www function https over TLS SSL
pass out quick on $oif proto tcp from $myip to any port = 443 $fks
EOF
################## End of IPF rules script ########################
```

That is all there is to it. The rules are not important in this example; how the symbolic substitution fields are populated and used are. If the above example was in a file named `/etc/ipf.rules.script`, these rules could be reloaded by entering the following command:

```
# sh /etc/ipf.rules.script
```

There is one problem with using a rules file with embedded symbolics: IPF does not understand symbolic substitution, and cannot read such scripts directly.

This script can be used in one of two ways:

- Uncomment the line that begins with `cat`, and comment out the line that begins with `/sbin/ipf`. Place `ipfilter_enable="YES"` into `/etc/rc.conf` as usual, and run script once after each modification to create or update `/etc/ipf.rules`.
- Disable IPFILTER in system startup scripts by adding `ipfilter_enable="NO"` (this is default value) into `/etc/rc.conf` file.

  Add a script like the following to your `/usr/local/etc/rc.d/` startup directory. The script should have an obvious name like `ipf.loadrules.sh`. The `.sh` extension is mandatory.

  ```
  #!/bin/sh
  sh /etc/ipf.rules.script
  ```

  The permissions on this script file must be read, write, execute for owner `root`.

  ```
  # chmod 700 /usr/local/etc/rc.d/ipf.loadrules.sh
  ```

Now, when your system boots, your IPF rules will be loaded.

30.5.10 IPF Rulesets

A ruleset is a group of IPF rules coded to pass or block packets based on the values contained in the packet. The bi-directional exchange of packets between hosts comprises a session conversation. The firewall ruleset processes both the packets arriving from the public Internet, as well as the packets produced by the system as a response to them. Each TCP/IP service (i.e.: telnet, www, mail, etc.) is predefined by its protocol and privileged (listening) port. Packets destined for a specific service, originate from the source address using an unprivileged (high order) port and

target the specific service port on the destination address. All the above parameters (i.e.: ports and addresses) can be used as selection criteria to create rules which will pass or block services.

IPF was originally written using a rules processing logic of "the last matching rule wins" and used only stateless rules. Over time IPF has been enhanced to include a "quick" option and a stateful "keep state" option which drastically modernized the rule processing logic.

The instructions contained in this section are based on using rules that contain the "quick" option and the stateful "keep state" option. This is the basic framework for coding an inclusive firewall rule set.

> **Warning:** When working with the firewall rules, be *very careful*. Some configurations *will lock you out* of the server. To be on the safe side, you may wish to consider performing the initial firewall configuration from the local console rather than doing it remotely e.g. via **ssh**.

30.5.11 Rule Syntax

The rule syntax presented here has been simplified to only address the modern stateful rule context and "first matching rule wins" logic. For the complete legacy rule syntax description see the ipf(8) manual page.

A # character is used to mark the start of a comment and may appear at the end of a rule line or on its own line. Blank lines are ignored.

Rules contain keywords. These keywords have to be coded in a specific order from left to right on the line. Keywords are identified in bold type. Some keywords have sub-options which may be keywords themselves and also include more sub-options. Each of the headings in the below syntax has a bold section header which expands on the content.

ACTION IN-OUT OPTIONS SELECTION STATEFUL PROTO SRC_ADDR,DST_ADDR OBJECT PORT_NUM TCP_FLAG STATEFUL

ACTION = block | pass

IN-OUT = in | out

OPTIONS = log | quick | on interface-name

SELECTION = proto value | source/destination IP | port = number | flags flag-value

PROTO = tcp/udp | udp | tcp | icmp

SRC_ADD,DST_ADDR = all | from object to object

OBJECT = IP address | any

PORT_NUM = port number

TCP_FLAG = S

STATEFUL = keep state

30.5.11.1 ACTION

The action indicates what to do with the packet if it matches the rest of the filter rule. Each rule *must* have an action. The following actions are recognized:

`block` indicates that the packet should be dropped if the selection parameters match the packet.

`pass` indicates that the packet should exit the firewall if the selection parameters match the packet.

30.5.11.2 IN-OUT

A mandatory requirement is that each filter rule explicitly state which side of the I/O it is to be used on. The next keyword must be either `in` or `out` and one or the other has to be coded or the rule will not pass syntax checks.

`in` means this rule is being applied against an inbound packet which has just been received on the interface facing the public Internet.

`out` means this rule is being applied against an outbound packet destined for the interface facing the public Internet.

30.5.11.3 OPTIONS

> **Note:** These options must be used in the order shown here.

`log` indicates that the packet header will be written to the `ipl` log (as described in the LOGGING section below) if the selection parameters match the packet.

`quick` indicates that if the selection parameters match the packet, this rule will be the last rule checked, allowing a "short-circuit" path to avoid processing any following rules for this packet. This option is a mandatory requirement for the modernized rules processing logic.

`on` indicates the interface name to be incorporated into the selection parameters. Interface names are as displayed by ifconfig(8). Using this option, the rule will only match if the packet is going through that interface in the specified direction (in/out). This option is a mandatory requirement for the modernized rules processing logic.

When a packet is logged, the headers of the packet are written to the IPL packet logging pseudo-device. Immediately following the `log` keyword, the following qualifiers may be used (in this order):

`body` indicates that the first 128 bytes of the packet contents will be logged after the headers.

`first` If the `log` keyword is being used in conjunction with a `keep state` option, it is recommended that this option is also applied so that only the triggering packet is logged and not every packet which thereafter matches the "keep state" information.

30.5.11.4 SELECTION

The keywords described in this section are used to describe attributes of the packet to be checked when determining whether rules match or not. There is a keyword subject, and it has sub-option keywords, one of which has to be selected. The following general-purpose attributes are provided for matching, and must be used in this order:

30.5.11.5 PROTO

`proto` is the subject keyword and must be coded along with one of its corresponding keyword sub-option values. The value allows a specific protocol to be matched against. This option is a mandatory requirement for the modernized rules processing logic.

`tcp/udp | udp | tcp | icmp` or any protocol names found in `/etc/protocols` are recognized and may be used. The special protocol keyword `tcp/udp` may be used to match either a TCP or a UDP packet, and has been added as a convenience to save duplication of otherwise identical rules.

30.5.11.6 SRC_ADDR/DST_ADDR

The `all` keyword is essentially a synonym for "from any to any" with no other match parameters.

`from src to dst`: the `from` and `to` keywords are used to match against IP addresses. Rules must specify *both* source and destination parameters. `any` is a special keyword that matches any IP address. Examples of use: `from any to any` or `from 0.0.0.0/0 to any` or `from any to 0.0.0.0/0` or `from 0.0.0.0 to any` or `from any to 0.0.0.0`.

There is no way to match ranges of IP addresses which do not express themselves easily using the dotted numeric form / mask-length notation. The `net-mgmt/ipcalc` port may be used to ease up the calculations. Additional information is available in the utility's web page: http://jodies.de/ipcalc.

30.5.11.7 PORT

If a port match is included, for either or both of source and destination, then it is only applied to TCP and UDP packets. When composing port comparisons, either the service name from `/etc/services` or an integer port number may be used. When the port appears as part of the `from` object, it matches the source port number; when it appears as part of the `to` object, it matches the destination port number. The use of the port option with the `to` object is a mandatory requirement for the modernized rules processing logic. Example of use: `from any to any port = 80`

Single port comparisons may be done in a number of ways, using a number of different comparison operators. Port ranges may also be specified.

port "=" | "!=" | "<" | ">" | "<=" | ">=" | "eq" | "ne" | "lt" | "gt" | "le" | "ge".

To specify port ranges, port "<>" | "><"

> **Warning:** Following the source and destination matching parameters, the following two parameters are mandatory requirements for the modernized rules processing logic.

30.5.11.8 TCP_FLAG

Flags are only effective for TCP filtering. The letters represent one of the possible flags that can be matched against the TCP packet header.

The modernized rules processing logic uses the `flags S` parameter to identify the tcp session start request.

30.5.11.9 STATEFUL

`keep state` indicates that on a pass rule, any packets that match the rules selection parameters should activate the stateful filtering facility.

Note: This option is a mandatory requirement for the modernized rules processing logic.

30.5.12 Stateful Filtering

Stateful filtering treats traffic as a bi-directional exchange of packets comprising a session conversation. When activated, keep-state dynamically generates internal rules for each anticipated packet being exchanged during the bi-directional session conversation. It has sufficient matching capabilities to determine if the session conversation between the originating sender and the destination are following the valid procedure of bi-directional packet exchange. Any packets that do not properly fit the session conversation template are automatically rejected as impostors.

Keep state will also allow ICMP packets related to a TCP or UDP session through. So if you get ICMP type 3 code 4 in response to some web surfing allowed out by a keep state rule, they will be automatically allowed in. Any packet that IPF can be certain is part of an active session, even if it is a different protocol, will be let in.

What happens is:

Packets destined to go out through the interface connected to the public Internet are first checked against the dynamic state table. If the packet matches the next expected packet comprising an active session conversation, then it exits the firewall and the state of the session conversation flow is updated in the dynamic state table. Packets that do not belong to an already active session, are simply checked against the outbound ruleset.

Packets coming in from the interface connected to the public Internet are first checked against the dynamic state table. If the packet matches the next expected packet comprising an active session conversation, then it exits the firewall and the state of the session conversation flow is updated in the dynamic state table. Packets that do not belong to an already active session, are simply checked against the inbound ruleset.

When the conversation completes it is removed from the dynamic state table.

Stateful filtering allows you to focus on blocking/passing new sessions. If the new session is passed, all its subsequent packets will be allowed through automatically and any impostors automatically rejected. If a new session is blocked, none of its subsequent packets will be allowed through. Stateful filtering has technically advanced matching abilities capable of defending against the flood of different attack methods currently employed by attackers.

30.5.13 Inclusive Ruleset Example

The following ruleset is an example of how to code a very secure inclusive type of firewall. An inclusive firewall only allows services matching `pass` rules through, and blocks all others by default. Firewalls intended to protect other machines, also called “network firewalls”, should have at least two interfaces, which are generally configured to trust one side (the LAN) and not the other (the public Internet). Alternatively, a firewall might be configured to protect only the system it is running on—this is called a “host based firewall”, and is particularly appropriate for servers on an untrusted network.

All UNIX flavored systems including FreeBSD are designed to use interface `lo0` and IP address `127.0.0.1` for internal communication within the operating system. The firewall rules must contain rules to allow free unmolested movement of these special internally used packets.

The interface which faces the public Internet is the one to place the rules that authorize and control access of the outbound and inbound connections. This can be your user PPP `tun0` interface or your NIC that is connected to your DSL or cable modem.

In cases where one or more NICs are cabled to private network segments, those interfaces may require rules to allow packets originating from those LAN interfaces transit to each other and/or to the outside (Internet).

The rules should be organized into three major sections: first trusted interfaces, then the public interface outbound, and last the public untrusted interface inbound.

The rules in each of the public interface sections should have the most frequently matched rules placed before less commonly matched rules, with the last rule in the section blocking and logging all packets on that interface and direction.

The Outbound section in the following ruleset only contains `pass` rules which contain selection values that uniquely identify the service that is authorized for public Internet access. All the rules have the `quick`, `on`, `proto`, `port`, and `keep state` options set. The `proto tcp` rules have the `flag` option included to identify the session start request as the triggering packet to activate the stateful facility.

The Inbound section has all the blocking of undesirable packets first, for two different reasons. The first is that malicious packets may be partial matches for legitimate traffic. These packets have to be discarded rather than allowed in, based on their partial matches against `allow` rules. The second reason is that known and uninteresting rejects may be blocked silently, rather than being caught and logged by the last rules in the section. The final rule in each section, blocks and logs all packets and can be used to create the legal evidence needed to prosecute the people who are attacking your system.

Another thing that should be taken care of, is to ensure there is no response returned for any of the undesirable traffic. Invalid packets should just get dropped and vanish. This way the attacker has no knowledge if his packets have reached your system. The less the attackers can learn about your system, the more time they must invest before actually doing something bad. Rules that include a `log first` option, will only log the event the first time they are triggered. This option is included in the sample `nmap OS fingerprint` rule. The `security/nmap` utility is commonly used by attackers who attempt to identify the operating system of your server.

Any time there are logged messages on a rule with the `log first` option, an `ipfstat -hio` command should be executed to evaluate how many times the rule has actually matched. Large number of matches usually indicate that the system is being flooded (i.e.: under attack).

The `/etc/services` file may be used to lookup unknown port numbers. Alternatively, visit http://www.securitystats.com/tools/portsearch.php and do a port number lookup to find the purpose of a particular port number.

Check out this link for port numbers used by Trojans http://www.simovits.com/trojans/trojans.html.

The following ruleset creates a complete and very secure `inclusive` type of firewall ruleset that has been tested on production systems. It can be easily modified for your own system. Just comment out any `pass` rules for services that should not be authorized.

To avoid logging unwanted messages, just add a `block` rule in the inbound section.

The `dc0` interface name has to be changed in every rule to the real interface name of the NIC card that connects your system to the public Internet. For user PPP it would be `tun0`.

Add the following statements to `/etc/ipf.rules`:

```
################################################################
# No restrictions on Inside LAN Interface for private network
```

```
# Not needed unless you have LAN
#####################################################################

#pass out quick on xl0 all
#pass in quick on xl0 all

#####################################################################
# No restrictions on Loopback Interface
#####################################################################
pass in quick on lo0 all
pass out quick on lo0 all

#####################################################################
# Interface facing Public Internet (Outbound Section)
# Match session start requests originating from behind the
# firewall on the private network
# or from this gateway server destined for the public Internet.
#####################################################################

# Allow out access to my ISP's Domain name server.
# xxx must be the IP address of your ISP's DNS.
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
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

# Allow out access to my ISP's DHCP server for cable or DSL networks.
# This rule is not needed for 'user ppp' type connection to the
# public Internet, so you can delete this whole group.
# Use the following rule and check log for IP address.
# Then put IP address in commented out rule & delete first rule
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


# Allow out non-secure standard www function
pass out quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Allow out secure www function https over TLS SSL
pass out quick on dc0 proto tcp from any to any port = 443 flags S keep state

# Allow out send & get email function
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

# Allow out Time
pass out quick on dc0 proto tcp from any to any port = 37 flags S keep state

# Allow out nntp news
pass out quick on dc0 proto tcp from any to any port = 119 flags S keep state

# Allow out gateway & LAN users' non-secure FTP ( both passive & active modes)
# This function uses the IPNAT built in FTP proxy function coded in
```

```
# the nat rules file to make this single rule function correctly.
# If you want to use the pkg_add command to install application packages
# on your gateway system you need this rule.
pass out quick on dc0 proto tcp from any to any port = 21 flags S keep state

# Allow out ssh/sftp/scp (telnet/rlogin/FTP replacements)
# This function is using SSH (secure shell)
pass out quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Allow out insecure Telnet
pass out quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Allow out FreeBSD CVSup
pass out quick on dc0 proto tcp from any to any port = 5999 flags S keep state

# Allow out ping to public Internet
pass out quick on dc0 proto icmp from any to any icmp-type 8 keep state

# Allow out whois from LAN to public Internet
pass out quick on dc0 proto tcp from any to any port = 43 flags S keep state

# Block and log only the first occurrence of everything
# else that's trying to get out.
# This rule implements the default block
block out log first quick on dc0 all

#################################################################
# Interface facing Public Internet (Inbound Section)
# Match packets originating from the public Internet
# destined for this gateway server or the private network.
#################################################################

# Block all inbound traffic from non-routable or reserved address spaces
block in quick on dc0 from 192.168.0.0/16 to any    #RFC 1918 private IP
block in quick on dc0 from 172.16.0.0/12 to any     #RFC 1918 private IP
block in quick on dc0 from 10.0.0.0/8 to any        #RFC 1918 private IP
block in quick on dc0 from 127.0.0.0/8 to any       #loopback
block in quick on dc0 from 0.0.0.0/8 to any         #loopback
block in quick on dc0 from 169.254.0.0/16 to any    #DHCP auto-config
block in quick on dc0 from 192.0.2.0/24 to any      #reserved for docs
block in quick on dc0 from 204.152.64.0/23 to any   #Sun cluster interconnect
block in quick on dc0 from 224.0.0.0/3 to any       #Class D & E multicast

##### Block a bunch of different nasty things. ############
# That I do not want to see in the log

# Block frags
block in quick on dc0 all with frags

# Block short tcp packets
block in quick on dc0 proto tcp all with short

# block source routed packets
```

```
block in quick on dc0 all with opt lsrr
block in quick on dc0 all with opt ssrr

# Block nmap OS fingerprint attempts
# Log first occurrence of these so I can get their IP address
block in log first quick on dc0 proto tcp from any to any flags FUP

# Block anything with special options
block in quick on dc0 all with ipopts

# Block public pings
block in quick on dc0 proto icmp all icmp-type 8

# Block ident
block in quick on dc0 proto tcp from any to any port = 113

# Block all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
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

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it's the only
# authorized source to send this packet type. Only necessary for
# cable or DSL configurations. This rule is not needed for
# 'user ppp' type connection to the public Internet.
# This is the same IP address you captured and
# used in the outbound section.
pass in quick on dc0 proto udp from z.z.z.z to any port = 68 keep state

# Allow in standard www function because I have apache server
pass in quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID/PW passed over public Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
#pass in quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Allow in secure FTP, Telnet, and SCP from public Internet
# This function is using SSH (secure shell)
pass in quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Block and log only first occurrence of all remaining traffic
# coming into the firewall. The logging of only the first
# occurrence avoids filling up disk with Denial of Service logs.
# This rule implements the default block.
block in log first quick on dc0 all
################### End of rules file ######################################
```

30.5.14 NAT

NAT stands for *Network Address Translation*. To those familiar with Linux, this concept is called IP Masquerading; NAT and IP Masquerading are the same thing. One of the many things the IPF NAT function enables is the ability to have a private Local Area Network (LAN) behind the firewall sharing a single ISP assigned IP address on the public Internet.

You may ask why would someone want to do this. ISPs normally assign a dynamic IP address to their non-commercial users. Dynamic means that the IP address can be different each time you dial in and log on to your ISP, or for cable and DSL modem users, when the modem is power cycled. This dynamic IP address is used to identify your system to the public Internet.

Now lets say you have five PCs at home and each one needs Internet access. You would have to pay your ISP for an individual Internet account for each PC and have five phone lines.

With NAT only a single account is needed with your ISP. The other four PCs may then be cabled to a switch and the switch to the NIC in your FreeBSD system which is going to service your LAN as a gateway. NAT will automatically translate the private LAN IP address for each separate PC on the LAN to the single public IP address as it exits the firewall bound for the public Internet. It also does the reverse translation for returning packets.

There is a special range of IP addresses reserved for NATed private LANs. According to RFC 1918, the following IP ranges may be used for private nets which will never be routed directly to the public Internet:

Start IP `10.0.0.0`	-	Ending IP `10`
Start IP `172.16.0.0`	-	Ending IP `17`
Start IP `192.168.0.0`	-	Ending IP `19`

30.5.15 IPNAT

NAT rules are loaded by using the `ipnat` command. Typically the NAT rules are stored in `/etc/ipnat.rules`. See ipnat(1) for details.

When changing the NAT rules after NAT has been started, make your changes to the file containing the NAT rules, then run the `ipnat` command with the `-CF` flags to delete the internal in use NAT rules and flush the contents of the translation table of all active entries.

To reload the NAT rules issue a command like this:

```
# ipnat -CF -f /etc/ipnat.rules
```

To display some statistics about your NAT, use this command:

```
# ipnat -s
```

To list the NAT table’s current mappings, use this command:

```
# ipnat -l
```

To turn verbose mode on, and display information relating to rule processing and active rules/table entries:

```
# ipnat -v
```

30.5.16 IPNAT Rules

NAT rules are very flexible and can accomplish many different things to fit the needs of commercial and home users.

The rule syntax presented here has been simplified to what is most commonly used in a non-commercial environment. For a complete rule syntax description see the ipnat(5) manual page.

The syntax for a NAT rule looks something like this:

```
map IF LAN_IP_RANGE -> PUBLIC_ADDRESS
```

The keyword `map` starts the rule.

Replace *`IF`* with the external interface.

The *`LAN_IP_RANGE`* is what your internal clients use for IP Addressing, usually this is something like `192.168.1.0/24`.

The *`PUBLIC_ADDRESS`* can either be the external IP address or the special keyword `0/32`, which means to use the IP address assigned to *`IF`*.

30.5.17 How NAT works

A packet arrives at the firewall from the LAN with a public destination. It passes through the outbound filter rules, NAT gets its turn at the packet and applies its rules top down, first matching rule wins. NAT tests each of its rules against the packet's interface name and source IP address. When a packet's interface name matches a NAT rule then the source IP address (i.e.: private LAN IP address) of the packet is checked to see if it falls within the IP address range specified to the left of the arrow symbol on the NAT rule. On a match the packet has its source IP address rewritten with the public IP address obtained by the `0/32` keyword. NAT posts an entry in its internal NAT table so when the packet returns from the public Internet it can be mapped back to its original private IP address and then passed to the filter rules for processing.

30.5.18 Enabling IPNAT

To enable IPNAT add these statements to `/etc/rc.conf`.

To enable your machine to route traffic between interfaces:

```
gateway_enable="YES"
```

To start IPNAT automatically each time:

```
ipnat_enable="YES"
```

To specify where to load the IPNAT rules from:

```
ipnat_rules="/etc/ipnat.rules"
```

30.5.19 NAT for a very large LAN

For networks that have large numbers of PC's on the LAN or networks with more than a single LAN, the process of funneling all those private IP addresses into a single public IP address becomes a resource problem that may cause

problems with the same port numbers being used many times across many NATed LAN PC's, causing collisions. There are two ways to relieve this resource problem.

30.5.19.1 Assigning Ports to Use

A normal NAT rule would look like:

```
map dc0 192.168.1.0/24 -> 0/32
```

In the above rule the packet's source port is unchanged as the packet passes through IPNAT. By adding the `portmap` keyword, IPNAT can be directed to only use source ports in the specified range. For example the following rule will tell IPNAT to modify the source port to be within the range shown:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:60000
```

Additionally we can make things even easier by using the `auto` keyword to tell IPNAT to determine by itself which ports are available to use:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp auto
```

30.5.19.2 Using a Pool of Public Addresses

In very large LANs there comes a point where there are just too many LAN addresses to fit into a single public address. If a block of public IP addresses is available, these addresses can be used as a "pool", and IPNAT may pick one of the public IP addresses as packet-addresses are mapped on their way out.

For example, instead of mapping all packets through a single public IP address, as in:

```
map dc0 192.168.1.0/24 -> 204.134.75.1
```

A range of public IP addresses can be specified either with a netmask:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/255.255.255.0
```

or using CIDR notation:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/24
```

30.5.20 Port Redirection

A very common practice is to have a web server, email server, database server and DNS server each segregated to a different PC on the LAN. In this case the traffic from these servers still have to be NATed, but there has to be some way to direct the inbound traffic to the correct LAN PCs. IPNAT has the redirection facilities of NAT to solve this problem. For example, assuming a web server operating on LAN address `10.0.10.25` and using a single public IP address of `20.20.20.5` the rule would be coded as follows:

```
rdr dc0 20.20.20.5/32 port 80 -> 10.0.10.25 port 80
```

or:

```
rdr dc0 0.0.0.0/0 port 80 -> 10.0.10.25 port 80
```

or for a LAN DNS Server on LAN address of `10.0.10.33` that needs to receive public DNS requests:

```
rdr dc0 20.20.20.5/32 port 53 -> 10.0.10.33 port 53 udp
```

30.5.21 FTP and NAT

FTP is a dinosaur left over from the time before the Internet as it is known today, when research universities were leased lined together and FTP was used to share files among research Scientists. This was a time when data security was not a consideration. Over the years the FTP protocol became buried into the backbone of the emerging Internet and its username and password being sent in clear text was never changed to address new security concerns. FTP has two flavors, it can run in active mode or passive mode. The difference is in how the data channel is acquired. Passive mode is more secure as the data channel is acquired by the ordinal ftp session requester. For a real good explanation of FTP and the different modes see http://www.slacksite.com/other/ftp.html.

30.5.21.1 IPNAT Rules

IPNAT has a special built in FTP proxy option which can be specified on the NAT map rule. It can monitor all outbound packet traffic for FTP active or passive start session requests and dynamically create temporary filter rules containing only the port number really in use for the data channel. This eliminates the security risk FTP normally exposes the firewall to from having large ranges of high order port numbers open.

This rule will handle all the traffic for the internal LAN:

```
map dc0 10.0.10.0/29 -> 0/32 proxy port 21 ftp/tcp
```

This rule handles the FTP traffic from the gateway:

```
map dc0 0.0.0.0/0 -> 0/32 proxy port 21 ftp/tcp
```

This rule handles all non-FTP traffic from the internal LAN:

```
map dc0 10.0.10.0/29 -> 0/32
```

The FTP map rule goes before our regular map rule. All packets are tested against the first rule from the top. Matches on interface name, then private LAN source IP address, and then is it a FTP packet. If all that matches then the special FTP proxy creates temp filter rules to let the FTP session packets pass in and out, in addition to also NATing the FTP packets. All LAN packets that are not FTP do not match the first rule and fall through to the third rule and are tested, matching on interface and source IP, then are NATed.

30.5.21.2 IPNAT FTP Filter Rules

Only one filter rule is needed for FTP if the NAT FTP proxy is used.

Without the FTP Proxy, the following three rules will be needed:

```
# Allow out LAN PC client FTP to public Internet
# Active and passive modes
pass out quick on rl0 proto tcp from any to any port = 21 flags S keep state
```

```
# Allow out passive mode data channel high order port numbers
pass out quick on rl0 proto tcp from any to any port > 1024 flags S keep state

# Active mode let data channel in from FTP server
pass in quick on rl0 proto tcp from any to any port = 20 flags S keep state
```

30.6 IPFW

The IPFIREWALL (IPFW) is a FreeBSD sponsored firewall software application authored and maintained by FreeBSD volunteer staff members. It uses the legacy stateless rules and a legacy rule coding technique to achieve what is referred to as Simple Stateful logic.

The IPFW sample ruleset (found in `/etc/rc.firewall` and `/etc/rc.firewall6`) in the standard FreeBSD install is rather simple and it is not expected to be used directly without modifications. The example does not use stateful filtering, which is beneficial in most setups, so it will not be used as base for this section.

The IPFW stateless rule syntax is empowered with technically sophisticated selection capabilities which far surpasses the knowledge level of the customary firewall installer. IPFW is targeted at the professional user or the advanced technical computer hobbyist who have advanced packet selection requirements. A high degree of detailed knowledge into how different protocols use and create their unique packet header information is necessary before the power of the IPFW rules can be unleashed. Providing that level of explanation is out of the scope of this section of the Handbook.

IPFW is composed of seven components, the primary component is the kernel firewall filter rule processor and its integrated packet accounting facility, the logging facility, the `divert` rule which triggers the NAT facility, and the advanced special purpose facilities, the dummynet traffic shaper facilities, the `fwd rule` forward facility, the bridge facility, and the ipstealth facility. IPFW supports both IPv4 and IPv6.

30.6.1 Enabling IPFW

IPFW is included in the basic FreeBSD install as a separate run time loadable module. The system will dynamically load the kernel module when the `rc.conf` statement `firewall_enable="YES"` is used. There is no need to compile IPFW into the FreeBSD kernel unless NAT functionality is desired.

After rebooting your system with `firewall_enable="YES"` in `rc.conf` the following white highlighted message is displayed on the screen as part of the boot process:

```
ipfw2 initialized, divert disabled, rule-based forwarding disabled, default to deny, logging disa
```

The loadable module does have logging ability compiled in. To enable logging and set the verbose logging limit, there is a knob that can be set in `/etc/sysctl.conf`. By adding these statements, logging will be enabled on future reboots:

```
net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose_limit=5
```

30.6.2 Kernel Options

It is not a mandatory requirement to enable IPFW by compiling the following options into the FreeBSD kernel, unless NAT functionality is required. It is presented here as background information.

```
options     IPFIREWALL
```

This option enables IPFW as part of the kernel

```
options     IPFIREWALL_VERBOSE
```

Enables logging of packets that pass through IPFW and have the `log` keyword specified in the ruleset.

```
options     IPFIREWALL_VERBOSE_LIMIT=5
```

Limits the number of packets logged through syslogd(8) on a per entry basis. This option may be used in hostile environments, when firewall activity logging is desired. This will close a possible denial of service attack via syslog flooding.

```
options     IPFIREWALL_DEFAULT_TO_ACCEPT
```

This option will allow everything to pass through the firewall by default, which is a good idea when the firewall is being set up for the first time.

```
options     IPDIVERT
```

This enables the use of NAT functionality.

> **Note:** The firewall will block all incoming and outgoing packets if either the `IPFIREWALL_DEFAULT_TO_ACCEPT` kernel option or a rule to explicitly allow these connections are missing.

30.6.3 `/etc/rc.conf` Options

Enable the firewall:

```
firewall_enable="YES"
```

To select one of the default firewall types provided by FreeBSD, select one by reading the `/etc/rc.firewall` file and place it in the following:

```
firewall_type="open"
```

Available values for this setting are:

- `open` — pass all traffic.
- `client` — will protect only this machine.
- `simple` — protect the whole network.
- `closed` — entirely disables IP traffic except for the loopback interface.

- `UNKNOWN` — disables the loading of firewall rules.
- *`filename`* — absolute path of file containing firewall rules.

It is possible to use two different ways to load custom rules for **ipfw** firewall. One is by setting `firewall_type` variable to absolute path of file, which contains *firewall rules* without any command-line options for ipfw(8) itself. The following is a simple example of a ruleset file that blocks all incoming and outgoing traffic:

```
add deny in
add deny out
```

On the other hand, it is possible to set the `firewall_script` variable to the absolute path of an executable script that includes `ipfw` commands being executed at system boot time. A valid ruleset script that would be equivalent to the ruleset file shown above would be the following:

```
#!/bin/sh

ipfw -q flush

ipfw add deny in
ipfw add deny out
```

> **Note:** If `firewall_type` is set to either `client` or `simple`, the default rules found in `/etc/rc.firewall` should be reviewed to fit to the configuration of the given machine. Also note that the examples used in this chapter expect that the `firewall_script` is set to `/etc/ipfw.rules`.

Enable logging:

```
firewall_logging="YES"
```

> **Warning:** The only thing that the `firewall_logging` variable will do is setting the `net.inet.ip.fw.verbose` sysctl variable to the value of `1` (see Section 30.6.1). There is no `rc.conf` variable to set log limitations, but it can be set via sysctl variable, manually or from the `/etc/sysctl.conf` file:
>
> ```
> net.inet.ip.fw.verbose_limit=5
> ```

If your machine is acting as a gateway, i.e. providing Network Address Translation (NAT) via natd(8), please refer to Section 31.9 for information regarding the required `/etc/rc.conf` options.

30.6.4 The IPFW Command

The `ipfw` command is the normal vehicle for making manual single rule additions or deletions to the active firewall internal rules while it is running. The problem with using this method is once your system is shutdown or halted all the rules that were added, changed or deleted are lost. Writing all your rules in a file and using that file to load the rules at boot time, or to replace in mass the currently running firewall rules with changes you made to the files content, is the recommended method used here.

The `ipfw` command is still a very useful way to display the running firewall rules to the console screen. The IPFW accounting facility dynamically creates a counter for each rule that counts each packet that matches the rule. During the process of testing a rule, listing the rule with its counter is one of the ways of determining if the rule is functioning.

To list all the rules in sequence:

```
# ipfw list
```

To list all the rules with a time stamp of when the last time the rule was matched:

```
# ipfw -t list
```

The next example lists accounting information, the packet count for matched rules along with the rules themselves. The first column is the rule number, followed by the number of outgoing matched packets, followed by the number of incoming matched packets, and then the rule itself.

```
# ipfw -a list
```

List the dynamic rules in addition to the static rules:

```
# ipfw -d list
```

Also show the expired dynamic rules:

```
# ipfw -d -e list
```

Zero the counters:

```
# ipfw zero
```

Zero the counters for just the rule with number *NUM*:

```
# ipfw zero NUM
```

30.6.5 IPFW Rulesets

A ruleset is a group of IPFW rules coded to allow or deny packets based on the values contained in the packet. The bi-directional exchange of packets between hosts comprises a session conversation. The firewall ruleset processes both the packets arriving from the public Internet, as well as the packets originating from the system as a response to them. Each TCP/IP service (i.e.: telnet, www, mail, etc.) is predefined by its protocol and privileged (listening) port. Packets destined for a specific service, originate from the source address using an unprivileged (high order) port and target the specific service port on the destination address. All the above parameters (i.e. ports and addresses) can be used as selection criteria to create rules which will pass or block services.

When a packet enters the firewall it is compared against the first rule in the ruleset and progresses one rule at a time moving from top to bottom of the set in ascending rule number sequence order. When the packet matches the selection parameters of a rule, the rules' action field value is executed and the search of the ruleset terminates for that packet. This is referred to as "the first match wins" search method. If the packet does not match any of the rules, it gets caught by the mandatory IPFW default rule, number 65535 which denies all packets and discards them without any reply back to the originating destination.

Note: The search continues after `count`, `skipto` and `tee` rules.

The instructions contained here are based on using rules that contain the stateful `keep state`, `limit`, `in`, `out` and `via` options. This is the basic framework for coding an inclusive type firewall ruleset.

Warning: Be careful when working with firewall rules, as it is easy to end up locking yourself out.

30.6.5.1 Rule Syntax

The rule syntax presented here has been simplified to what is necessary to create a standard inclusive type firewall ruleset. For a complete rule syntax description see the ipfw(8) manual page.

Rules contain keywords: these keywords have to be coded in a specific order from left to right on the line. Keywords are identified in bold type. Some keywords have sub-options which may be keywords them selves and also include more sub-options.

is used to mark the start of a comment and may appear at the end of a rule line or on its own lines. Blank lines are ignored.

`CMD RULE_NUMBER ACTION LOGGING SELECTION STATEFUL`

30.6.5.1.1 CMD

Each new rule has to be prefixed with *`add`* to add the rule to the internal table.

30.6.5.1.2 RULE_NUMBER

Each rule has to have a rule number to go with it.

30.6.5.1.3 ACTION

A rule can be associated with one of the following actions, which will be executed when the packet matches the selection criterion of the rule.

`allow | accept | pass | permit`

These all mean the same thing which is to allow packets that match the rule to exit the firewall rule processing. The search terminates at this rule.

`check-state`

Checks the packet against the dynamic rules table. If a match is found, execute the action associated with the rule which generated this dynamic rule, otherwise move to the next rule. The check-state rule does not have selection criterion. If no check-state rule is present in the ruleset, the dynamic rules table is checked at the first keep-state or limit rule.

`deny | drop`

Both words mean the same thing which is to discard packets that match this rule. The search terminates.

30.6.5.1.4 Logging

`log` or *`logamount`*

When a packet matches a rule with the `log` keyword, a message will be logged to syslogd(8) with a facility name of SECURITY. The logging only occurs if the number of packets logged so far for that particular rule does not exceed the `logamount` parameter. If no `logamount` is specified, the limit is taken from the sysctl variable `net.inet.ip.fw.verbose_limit`. In both cases, a value of zero removes the logging limit. Once the limit is reached, logging can be re-enabled by clearing the logging counter or the packet counter for that rule, see the `ipfw reset log` command.

> **Note:** Logging is done after all other packet matching conditions have been successfully verified, and before performing the final action (accept, deny) on the packet. It is up to you to decide which rules you want to enable logging on.

30.6.5.1.5 Selection

The keywords described in this section are used to describe attributes of the packet to be checked when determining whether rules match the packet or not. The following general-purpose attributes are provided for matching, and must be used in this order:

`udp | tcp | icmp`

Any other protocol names found in `/etc/protocols` are also recognized and may be used. The value specified is the protocol to be matched against. This is a mandatory requirement.

`from src to dst`

The `from` and `to` keywords are used to match against IP addresses. Rules must specify *both* source and destination parameters. `any` is a special keyword that matches any IP address. `me` is a special keyword that matches any IP address configured on an interface in your FreeBSD system to represent the PC the firewall is running on (i.e.: this box) as in `from me to any` or `from any to me` or `from 0.0.0.0/0 to any` or `from any to 0.0.0.0/0` or `from 0.0.0.0 to any` or `from any to 0.0.0.0` or `from me to 0.0.0.0`. IP addresses are specified as a dotted IP address numeric form/mask-length (CIDR notation), or as single dotted IP address numeric form. This is a mandatory requirement. The `net-mgmt/ipcalc` port may be used to ease up the calculations. Additional information is available in the utility's web page: http://jodies.de/ipcalc.

`port number`

For protocols which support port numbers (such as TCP and UDP), it is mandatory to code the port number of the service that will be matched. Service names (from `/etc/services`) may be used instead of numeric port values.

`in | out`

Matches incoming or outgoing packets, respectively. The `in` and `out` are keywords and it is mandatory that one or the other is coded as part of your rule matching criterion.

`via IF`

Matches packets going through the interface specified by exact name. The `via` keyword causes the interface to always be checked as part of the match process.

`setup`

This is a mandatory keyword that identifies the session start request for TCP packets.

keep-state

This is a mandatory keyword. Upon a match, the firewall will create a dynamic rule, whose default behavior is to match bidirectional traffic between source and destination IP/port using the same protocol.

limit {src-addr | src-port | dst-addr | dst-port}

The firewall will only allow *N* connections with the same set of parameters as specified in the rule. One or more of source and destination addresses and ports can be specified. The `limit` and `keep-state` can not be used on the same rule. The `limit` option provides the same stateful function as `keep-state`, plus its own functions.

30.6.5.2 Stateful Rule Option

Stateful filtering treats traffic as a bi-directional exchange of packets comprising a session conversation. It has the matching capabilities to determine if the session conversation between the originating sender and the destination are following the valid procedure of bi-directional packet exchange. Any packets that do not properly fit the session conversation template are automatically rejected as impostors.

The `check-state` option is used to identify where in the IPFW rules set the packet is to be tested against the dynamic rules facility. On a match the packet exits the firewall to continue on its way and a new rule is dynamically created for the next anticipated packet being exchanged during this bi-directional session conversation. On a no match the packet advances to the next rule in the ruleset for testing.

The dynamic rules facility is vulnerable to resource depletion from a SYN-flood attack which would open a huge number of dynamic rules. To counter this attack, FreeBSD added another new option named `limit`. This option is used to limit the number of simultaneous session conversations by checking the rules source or destinations fields as directed by the `limit` option and using the packet's IP address found there, in a search of the open dynamic rules counting the number of times this rule and IP address combination occurred, if this count is greater that the value specified on the `limit` option, the packet is discarded.

30.6.5.3 Logging Firewall Messages

The benefits of logging are obvious: it provides the ability to review after the fact the rules you activated logging on which provides information like, what packets had been dropped, what addresses they came from and where they were going, giving you a significant edge in tracking down attackers.

Even with the logging facility enabled, IPFW will not generate any rule logging on it's own. The firewall administrator decides what rules in the ruleset will be logged, and adds the `log` verb to those rules. Normally only deny rules are logged, like the deny rule for incoming ICMP pings. It is very customary to duplicate the "ipfw default deny everything" rule with the `log` verb included as your last rule in the ruleset. This way it is possible to see all the packets that did not match any of the rules in the ruleset.

Logging is a two edged sword, if you are not careful, you can lose yourself in the over abundance of log data and fill your disk up with growing log files. DoS attacks that fill up disk drives is one of the oldest attacks around. These log messages are not only written to **syslogd**, but also are displayed on the root console screen and soon become very annoying.

The `IPFIREWALL_VERBOSE_LIMIT=5` kernel option limits the number of consecutive messages sent to the system logger syslogd(8), concerning the packet matching of a given rule. When this option is enabled in the kernel, the number of consecutive messages concerning a particular rule is capped at the number specified. There is nothing to

be gained from 200 log messages saying the same identical thing. For instance, five consecutive messages concerning a particular rule would be logged to **syslogd**, the remainder identical consecutive messages would be counted and posted to **syslogd** with a phrase like the following:

```
last message repeated 45 times
```

All logged packets messages are written by default to `/var/log/security` file, which is defined in the `/etc/syslog.conf` file.

30.6.5.4 Building a Rule Script

Most experienced IPFW users create a file containing the rules and code them in a manner compatible with running them as a script. The major benefit of doing this is the firewall rules can be refreshed in mass without the need of rebooting the system to activate them. This method is very convenient in testing new rules as the procedure can be executed as many times as needed. Being a script, symbolic substitution can be used to code frequent used values and substitute them in multiple rules. This is shown in the following example.

The script syntax used here is compatible with the sh(1), csh(1), tcsh(1) shells. Symbolic substitution fields are prefixed with a dollar sign $. Symbolic fields do not have the $ prefix. The value to populate the symbolic field must be enclosed in "double quotes".

Start your rules file like this:

```
############### start of example ipfw rules script #############
#
ipfw -q -f flush       # Delete all rules
# Set defaults
oif="tun0"             # out interface
odns="192.0.2.11"      # ISP's DNS server IP address
cmd="ipfw -q add "     # build rule prefix
ks="keep-state"        # just too lazy to key this each time
$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
################### End of example ipfw rules script ############
```

That is all there is to it. The rules are not important in this example, how the symbolic substitution field are populated and used are.

If the above example was in the `/etc/ipfw.rules` file, the rules could be reloaded by entering the following on the command line.

```
# sh /etc/ipfw.rules
```

The `/etc/ipfw.rules` file could be located anywhere you want and the file could be named any thing you would like.

The same thing could also be accomplished by running these commands by hand:

```
# 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 Stateful Ruleset

The following non-NATed ruleset is an example of how to code a very secure 'inclusive' type of firewall. An inclusive firewall only allows services matching pass rules through and blocks all other by default. Firewalls designed to protect entire network segments, have at minimum two interfaces which must have rules to allow the firewall to function.

All UNIX flavored operating systems, FreeBSD included, are designed to use interface `lo0` and IP address `127.0.0.1` for internal communication with in the operating system. The firewall rules must contain rules to allow free unmolested movement of these special internally used packets.

The interface which faces the public Internet is the one to place the rules that authorize and control access of the outbound and inbound connections. This can be your user PPP `tun0` interface or your NIC that is connected to your DSL or cable modem.

In cases where one or more than one NICs are connected to a private LAN behind the firewall, those interfaces must have rules coded to allow free unmolested movement of packets originating from those LAN interfaces.

The rules should be first organized into three major sections, all the free unmolested interfaces, public interface outbound, and the public interface inbound.

The order of the rules in each of the public interface sections should be in order of the most used rules being placed before less often used rules with the last rule in the section blocking and logging all packets on that interface and direction.

The Outbound section in the following ruleset only contains `allow` rules which contain selection values that uniquely identify the service that is authorized for public Internet access. All the rules have the `proto`, `port`, `in/out`, `via` and `keep state` option coded. The `proto tcp` rules have the `setup` option included to identify the start session request as the trigger packet to be posted to the keep state stateful table.

The Inbound section has all the blocking of undesirable packets first, for two different reasons. The first is that malicious packets may be partial matches for legitimate traffic. These packets have to be discarded rather than allowed in, based on their partial matches against `allow` rules. The second reason is that known and uninteresting rejects may be blocked silently, rather than being caught and logged by the last rules in the section. The final rule in each section, blocks and logs all packets and can be used to create the legal evidence needed to prosecute the people who are attacking your system.

Another thing that should be taken care of, is to insure there is no response returned for any of the undesirable stuff. Invalid packets should just get dropped and vanish. This way the attacker has no knowledge if his packets have reached your system. The less the attackers can learn about your system, the more secure it is. Packets with unrecognized port numbers may be looked up in `/etc/services/` or go to http://www.securitystats.com/tools/portsearch.php and do a port number lookup to find the purpose of the particular port number is. Check out this link for port numbers used by Trojans: http://www.simovits.com/trojans/trojans.html.

30.6.5.6 An Example Inclusive Ruleset

The following non-NATed ruleset is a complete inclusive type ruleset. It is safe to use this ruleset on your own systems. Just comment out any `pass` rules for services that are not required. To avoid logging undesired messages, add a `deny` rule in the inbound section. The `dc0` interface will will have to be changed in every rule, with the actual name of the interface (NIC) that connects your system to the public Internet. For user PPP, this would be `tun0`.

There is a noticeable pattern in the usage of these rules.

- All statements that are a request to start a session to the public Internet use `keep-state`.
- All the authorized services that originate from the public Internet have the `limit` option to stop flooding.
- All rules use `in` or `out` to clarify direction.
- All rules use `via` *`interface-name`* to specify the interface the packet is traveling over.

The following rules go into `/etc/ipfw.rules`.

```
################ Start of IPFW rules file ###############################
# Flush out the list before we begin.
ipfw -q -f flush

# Set rules command prefix
cmd="ipfw -q add"
pif="dc0"     # public interface name of NIC
              # facing the public Internet

######################################################################
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN.
# Change xl0 to your LAN NIC interface name
######################################################################
#$cmd 00005 allow all from any to any via xl0

######################################################################
# No restrictions on Loopback Interface
######################################################################
$cmd 00010 allow all from any to any via lo0

######################################################################
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
######################################################################
$cmd 00015 check-state

######################################################################
# Interface facing Public Internet (Outbound Section)
# Interrogate session start requests originating from behind the
# firewall on the private network or from this gateway server
# destine for the public Internet.
######################################################################

# Allow out access to my ISP's Domain name server.
# x.x.x.x must be the IP address of your ISP.s DNS
```

```
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
$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

# Allow out access to my ISP's DHCP server for cable/DSL configurations.
# This rule is not needed for .user ppp. connection to the public Internet.
# so you can delete this whole group.
# Use the following rule and check log for IP address.
# Then put IP address in commented out rule & delete first rule
$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

# Allow out non-secure standard www function
$cmd 00200 allow tcp from any to any 80 out via $pif setup keep-state

# Allow out secure www function https over TLS SSL
$cmd 00220 allow tcp from any to any 443 out via $pif setup keep-state

# Allow out send & get email function
$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

# Allow out FBSD (make install & CVSUP) functions
# Basically give user root "GOD" privileges.
$cmd 00240 allow tcp from me to any out via $pif setup keep-state uid root

# Allow out ping
$cmd 00250 allow icmp from any to any out via $pif keep-state

# Allow out Time
$cmd 00260 allow tcp from any to any 37 out via $pif setup keep-state

# Allow out nntp news (i.e. news groups)
$cmd 00270 allow tcp from any to any 119 out via $pif setup keep-state

# Allow out secure FTP, Telnet, and SCP
# This function is using SSH (secure shell)
$cmd 00280 allow tcp from any to any 22 out via $pif setup keep-state

# Allow out whois
$cmd 00290 allow tcp from any to any 43 out via $pif setup keep-state

# deny and log everything else that.s trying to get out.
# This rule enforces the block all by default logic.
$cmd 00299 deny log all from any to any out via $pif

########################################################################
# Interface facing Public Internet (Inbound Section)
# Check packets originating from the public Internet
# destined for this gateway server or the private network.
########################################################################
```

```
# Deny all inbound traffic from non-routable reserved address spaces
$cmd 00300 deny all from 192.168.0.0/16 to any in via $pif  #RFC 1918 private IP
$cmd 00301 deny all from 172.16.0.0/12 to any in via $pif     #RFC 1918 private IP
$cmd 00302 deny all from 10.0.0.0/8 to any in via $pif          #RFC 1918 private IP
$cmd 00303 deny all from 127.0.0.0/8 to any in via $pif        #loopback
$cmd 00304 deny all from 0.0.0.0/8 to any in via $pif            #loopback
$cmd 00305 deny all from 169.254.0.0/16 to any in via $pif   #DHCP auto-config
$cmd 00306 deny all from 192.0.2.0/24 to any in via $pif       #reserved for docs
$cmd 00307 deny all from 204.152.64.0/23 to any in via $pif  #Sun cluster interconnect
$cmd 00308 deny all from 224.0.0.0/3 to any in via $pif         #Class D & E multicast

# Deny public pings
$cmd 00310 deny icmp from any to any in via $pif

# Deny ident
$cmd 00315 deny tcp from any to any 113 in via $pif

# Deny all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
$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

# Deny any late arriving packets
$cmd 00330 deny all from any to any frag in via $pif

# Deny ACK packets that did not match the dynamic rule table
$cmd 00332 deny tcp from any to any established in via $pif

# Allow traffic in from ISP’s DHCP server. This rule must contain
# the IP address of your ISP.s DHCP server as it.s the only
# authorized source to send this packet type.
# Only necessary for cable or DSL configurations.
# This rule is not needed for .user ppp. type connection to
# the public Internet. This is the same IP address you captured
# and used in the outbound section.
#$cmd 00360 allow udp from any to x.x.x.x 67 in via $pif keep-state

# Allow in standard www function because I have apache server
$cmd 00400 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow in secure FTP, Telnet, and SCP from public Internet
$cmd 00410 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID & PW are passed over public
# Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
$cmd 00420 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Reject & Log all incoming connections from the outside
```

```
$cmd 00499 deny log all from any to any in via $pif

# Everything else is denied by default
# deny and log all packets that fell through to see what they are
$cmd 00999 deny log all from any to any
################ End of IPFW rules file ################################
```

30.6.5.7 An Example NAT and Stateful Ruleset

There are some additional configuration statements that need to be enabled to activate the NAT function of IPFW. The kernel source needs `option IPDIVERT` statement added to the other IPFIREWALL statements compiled into a custom kernel.

In addition to the normal IPFW options in `/etc/rc.conf`, the following are needed.

```
natd_enable="YES"                   # Enable NATD function
natd_interface="rl0"                # interface name of public Internet NIC
natd_flags="-dynamic -m"            # -m = preserve port numbers if possible
```

Utilizing stateful rules with `divert natd` rule (Network Address Translation) greatly complicates the ruleset coding logic. The positioning of the `check-state`, and `divert natd` rules in the ruleset becomes very critical. This is no longer a simple fall-through logic flow. A new action type is used, called `skipto`. To use the `skipto` command it is mandatory that each rule is numbered, so the `skipto` rule number knows exactly where it is jumping to.

The following is an uncommented example of one coding method, selected here to explain the sequence of the packet flow through the rulesets.

The processing flow starts with the first rule from the top of the rule file and progress one rule at a time deeper into the file until the end is reached or the packet being tested to the selection criteria matches and the packet is released out of the firewall. It is important to take notice of the location of rule numbers 100 101, 450, 500, and 510. These rules control the translation of the outbound and inbound packets so their entries in the keep-state dynamic table always register the private LAN IP address. Next notice that all the allow and deny rules specify the direction the packet is going (i.e.: outbound or inbound) and the interface. Also notice that the start outbound session requests, all `skipto rule 500` for the network address translation.

Lets say a LAN user uses their web browser to get a web page. Web pages are transmitted over port 80. So the packet enters the firewall. It does not match rule 100 because it is headed out rather than in. It passes rule 101 because this is the first packet, so it has not been posted to the keep-state dynamic table yet. The packet finally comes to rule 125 a matches. It is outbound through the NIC facing the public Internet. The packet still has it's source IP address as a private LAN IP address. On the match to this rule, two actions take place. The `keep-state` option will post this rule into the keep-state dynamic rules table and the specified action is executed. The action is part of the info posted to the dynamic table. In this case it is `skipto rule 500`. Rule 500 NATs the packet IP address and out it goes. Remember this, this is very important. This packet makes its way to the destination, where a response packet is generated and sent back. This new packet enters the top of the ruleset. This time it does match rule 100 and has it destination IP address mapped back to its corresponding LAN IP address. It then is processed by the `check-state` rule, it is found in the table as an existing session conversation and released to the LAN. It goes to the LAN PC that sent it and a new packet is sent requesting another segment of the data from the remote server. This time it gets checked by the `check-state` rule and its outbound entry is found, the associated action, `skipto 500`, is executed. The packet jumps to rule 500 gets NATed and released on it's way out.

On the inbound side, everything coming in that is part of an existing session conversation is being automatically handled by the `check-state` rule and the properly placed `divert natd` rules. All we have to address is denying all the bad packets and only allowing in the authorized services. Lets say there is an apache server running on the firewall box and we want people on the public Internet to be able to access the local web site. The new inbound start request packet matches rule 100 and its IP address is mapped to LAN IP for the firewall box. The packet is then matched against all the nasty things that need to be checked for and finally matches against rule 425. On a match two things occur. The packet rule is posted to the keep-state dynamic table but this time any new session requests originating from that source IP address is limited to 2. This defends against DoS attacks of service running on the specified port number. The action is `allow` so the packet is released to the LAN. The packet generated as a response, is recognized by the `check-state` as belonging to an existing session conversation. It is then sent to rule 500 for NATing and released to the outbound interface.

Example Ruleset #1:

```
#!/bin/sh
cmd="ipfw -q add"
skip="skipto 500"
pif=rl0
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 xl0  # exclude LAN traffic
$cmd 003 allow all from any to any via lo0  # exclude loopback traffic

$cmd 100 divert natd ip from any to any in via $pif
$cmd 101 check-state

# Authorized outbound packets
$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


# Deny all inbound traffic from non-routable reserved address spaces
$cmd 300 deny all from 192.168.0.0/16  to any in via $pif  #RFC 1918 private IP
$cmd 301 deny all from 172.16.0.0/12   to any in via $pif  #RFC 1918 private IP
$cmd 302 deny all from 10.0.0.0/8      to any in via $pif  #RFC 1918 private IP
$cmd 303 deny all from 127.0.0.0/8     to any in via $pif  #loopback
$cmd 304 deny all from 0.0.0.0/8       to any in via $pif  #loopback
$cmd 305 deny all from 169.254.0.0/16  to any in via $pif  #DHCP auto-config
$cmd 306 deny all from 192.0.2.0/24    to any in via $pif  #reserved for docs
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif  #Sun cluster
$cmd 308 deny all from 224.0.0.0/3     to any in via $pif  #Class D & E multicast

# Authorized inbound packets
$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

# This is skipto location for outbound stateful rules
$cmd 500 divert natd ip from any to any out via $pif
$cmd 510 allow ip from any to any

######################## end of rules  ##################
```

The following is pretty much the same as above, but uses a self documenting coding style full of description comments to help the inexperienced IPFW rule writer to better understand what the rules are doing.

Example Ruleset #2:

```
#!/bin/sh
############### Start of IPFW rules file ###############################
# Flush out the list before we begin.
ipfw -q -f flush

# Set rules command prefix
cmd="ipfw -q add"
skip="skipto 800"
pif="rl0"     # public interface name of NIC
              # facing the public Internet

######################################################################
# No restrictions on Inside LAN Interface for private network
# Change xl0 to your LAN NIC interface name
######################################################################
$cmd 005 allow all from any to any via xl0

######################################################################
# No restrictions on Loopback Interface
######################################################################
$cmd 010 allow all from any to any via lo0

######################################################################
# check if packet is inbound and nat address if it is
######################################################################
$cmd 014 divert natd ip from any to any in via $pif

######################################################################
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
######################################################################
$cmd 015 check-state

######################################################################
# Interface facing Public Internet (Outbound Section)
# Check session start requests originating from behind the
# firewall on the private network or from this gateway server
# destined for the public Internet.
######################################################################
```

```
# Allow out access to my ISP's Domain name server.
# x.x.x.x must be the IP address of your ISP's DNS
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
$cmd 020 $skip tcp from any to x.x.x.x 53 out via $pif setup keep-state


# Allow out access to my ISP's DHCP server for cable/DSL configurations.
$cmd 030 $skip udp from any to x.x.x.x 67 out via $pif keep-state

# Allow out non-secure standard www function
$cmd 040 $skip tcp from any to any 80 out via $pif setup keep-state

# Allow out secure www function https over TLS SSL
$cmd 050 $skip tcp from any to any 443 out via $pif setup keep-state

# Allow out send & get email function
$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

# Allow out FreeBSD (make install & CVSUP) functions
# Basically give user root "GOD" privileges.
$cmd 070 $skip tcp from me to any out via $pif setup keep-state uid root

# Allow out ping
$cmd 080 $skip icmp from any to any out via $pif keep-state

# Allow out Time
$cmd 090 $skip tcp from any to any 37 out via $pif setup keep-state

# Allow out nntp news (i.e. news groups)
$cmd 100 $skip tcp from any to any 119 out via $pif setup keep-state

# Allow out secure FTP, Telnet, and SCP
# This function is using SSH (secure shell)
$cmd 110 $skip tcp from any to any 22 out via $pif setup keep-state

# Allow out whois
$cmd 120 $skip tcp from any to any 43 out via $pif setup keep-state

# Allow ntp time server
$cmd 130 $skip udp from any to any 123 out via $pif keep-state

#####################################################################
# Interface facing Public Internet (Inbound Section)
# Check packets originating from the public Internet
# destined for this gateway server or the private network.
#####################################################################

# Deny all inbound traffic from non-routable reserved address spaces
$cmd 300 deny all from 192.168.0.0/16  to any in via $pif  #RFC 1918 private IP
$cmd 301 deny all from 172.16.0.0/12   to any in via $pif  #RFC 1918 private IP
$cmd 302 deny all from 10.0.0.0/8      to any in via $pif  #RFC 1918 private IP
```

```
$cmd 303 deny all from 127.0.0.0/8     to any in via $pif  #loopback
$cmd 304 deny all from 0.0.0.0/8       to any in via $pif  #loopback
$cmd 305 deny all from 169.254.0.0/16  to any in via $pif  #DHCP auto-config
$cmd 306 deny all from 192.0.2.0/24    to any in via $pif  #reserved for docs
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif  #Sun cluster
$cmd 308 deny all from 224.0.0.0/3     to any in via $pif  #Class D & E multicast

# Deny ident
$cmd 315 deny tcp from any to any 113 in via $pif

# Deny all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
$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

# Deny any late arriving packets
$cmd 330 deny all from any to any frag in via $pif

# Deny ACK packets that did not match the dynamic rule table
$cmd 332 deny tcp from any to any established in via $pif

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it's the only
# authorized source to send this packet type.
# Only necessary for cable or DSL configurations.
# This rule is not needed for 'user ppp' type connection to
# the public Internet. This is the same IP address you captured
# and used in the outbound section.
$cmd 360 allow udp from x.x.x.x to any 68 in via $pif keep-state

# Allow in standard www function because I have Apache server
$cmd 370 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow in secure FTP, Telnet, and SCP from public Internet
$cmd 380 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID & PW are passed over public
# Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
$cmd 390 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Reject & Log all unauthorized incoming connections from the public Internet
$cmd 400 deny log all from any to any in via $pif

# Reject & Log all unauthorized out going connections to the public Internet
$cmd 450 deny log all from any to any out via $pif

# This is skipto location for outbound stateful rules
$cmd 800 divert natd ip from any to any out via $pif
```

```
$cmd 801 allow ip from any to any

# Everything else is denied by default
# deny and log all packets that fell through to see what they are
$cmd 999 deny log all from any to any
################ End of IPFW rules file ################################
```

Chapter 31
Advanced Networking

31.1 Synopsis

This chapter will cover a number of advanced networking topics.

After reading this chapter, you will know:

- The basics of gateways and routes.
- How to set up IEEE® 802.11 and Bluetooth devices.
- How to make FreeBSD act as a bridge.
- How to set up network booting on a diskless machine.
- How to set up network address translation.
- How to connect two computers via PLIP.
- How to set up IPv6 on a FreeBSD machine.
- How to configure ATM.
- How to enable and utilize the features of CARP, the Common Address Redundancy Protocol in FreeBSD

Before reading this chapter, you should:

- Understand the basics of the `/etc/rc` scripts.
- Be familiar with basic network terminology.
- Know how to configure and install a new FreeBSD kernel (Chapter 8).
- Know how to install additional third-party software (Chapter 4).

31.2 Gateways and Routes

For one machine to be able to find another over a network, there must be a mechanism in place to describe how to get from one to the other. This is called *routing*. A “route” is a defined pair of addresses: a “destination” and a “gateway”. The pair indicates that if you are trying to get to this *destination*, communicate through this *gateway*. There are three types of destinations: individual hosts, subnets, and “default”. The “default route” is used if none of the other routes apply. We will talk a little bit more about default routes later on. There are also three types of gateways: individual hosts, interfaces (also called “links”), and Ethernet hardware addresses (MAC addresses).

31.2.1 An Example

To illustrate different aspects of routing, we will use the following example from `netstat`:

```
% 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
```

The first two lines specify the default route (which we will cover in the next section) and the `localhost` route.

The interface (`Netif` column) that this routing table specifies to use for `localhost` is `lo0`, also known as the loopback device. This says to keep all traffic for this destination internal, rather than sending it out over the LAN, since it will only end up back where it started.

The next thing that stands out are the addresses beginning with `0:e0:`. These are Ethernet hardware addresses, which are also known as MAC addresses. FreeBSD will automatically identify any hosts (`test0` in the example) on the local Ethernet and add a route for that host, directly to it over the Ethernet interface, `ed0`. There is also a timeout (`Expire` column) associated with this type of route, which is used if we fail to hear from the host in a specific amount of time. When this happens, the route to this host will be automatically deleted. These hosts are identified using a mechanism known as RIP (Routing Information Protocol), which figures out routes to local hosts based upon a shortest path determination.

FreeBSD will also add subnet routes for the local subnet (`10.20.30.255` is the broadcast address for the subnet `10.20.30`, and `example.com` is the domain name associated with that subnet). The designation `link#1` refers to the first Ethernet card in the machine. You will notice no additional interface is specified for those.

Both of these groups (local network hosts and local subnets) have their routes automatically configured by a daemon called **routed**. If this is not run, then only routes which are statically defined (i.e. entered explicitly) will exist.

The `host1` line refers to our host, which it knows by Ethernet address. Since we are the sending host, FreeBSD knows to use the loopback interface (`lo0`) rather than sending it out over the Ethernet interface.

The two `host2` lines are an example of what happens when we use an ifconfig(8) alias (see the section on Ethernet for reasons why we would do this). The `=>` symbol after the `lo0` interface says that not only are we using the loopback (since this address also refers to the local host), but specifically it is an alias. Such routes only show up on the host that supports the alias; all other hosts on the local network will simply have a `link#1` line for such routes.

The final line (destination subnet `224`) deals with multicasting, which will be covered in another section.

Finally, various attributes of each route can be seen in the `Flags` column. Below is a short table of some of these flags and their meanings:

U	Up: The route is active.
H	Host: The route destination is a single host.

G	Gateway: Send anything for this destination on to this remote system, which will figure out from there where to send it.
S	Static: This route was configured manually, not automatically generated by the system.
C	Clone: Generates a new route based upon this route for machines we connect to. This type of route is normally used for local networks.
W	WasCloned: Indicated a route that was auto-configured based upon a local area network (Clone) route.
L	Link: Route involves references to Ethernet hardware.

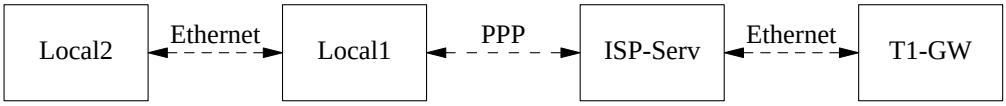
31.2.2 Default Routes

When the local system needs to make a connection to a remote host, it checks the routing table to determine if a known path exists. If the remote host falls into a subnet that we know how to reach (Cloned routes), then the system checks to see if it can connect along that interface.

If all known paths fail, the system has one last option: the "default" route. This route is a special type of gateway route (usually the only one present in the system), and is always marked with a `c` in the flags field. For hosts on a local area network, this gateway is set to whatever machine has a direct connection to the outside world (whether via PPP link, DSL, cable modem, T1, or another network interface).

If you are configuring the default route for a machine which itself is functioning as the gateway to the outside world, then the default route will be the gateway machine at your Internet Service Provider's (ISP) site.

Let us look at an example of default routes. This is a common configuration:



The hosts `Local1` and `Local2` are at your site. `Local1` is connected to an ISP via a dial up PPP connection. This PPP server computer is connected through a local area network to another gateway computer through an external interface to the ISPs Internet feed.

The default routes for each of your machines will be:

Host	Default Gateway	Interface
Local2	Local1	Ethernet
Local1	T1-GW	PPP

A common question is "Why (or how) would we set the `T1-GW` to be the default gateway for `Local1`, rather than the ISP server it is connected to?".

Remember, since the PPP interface is using an address on the ISP's local network for your side of the connection, routes for any other machines on the ISP's local network will be automatically generated. Hence, you will already know how to reach the `T1-GW` machine, so there is no need for the intermediate step of sending traffic to the ISP server.

It is common to use the address `X.X.X.1` as the gateway address for your local network. So (using the same example), if your local class-C address space was `10.20.30` and your ISP was using `10.9.9` then the default routes

would be:

Host	Default Route
Local2 (10.20.30.2)	Local1 (10.20.30.1)
Local1 (10.20.30.1, 10.9.9.30)	T1-GW (10.9.9.1)

You can easily define the default route via the `/etc/rc.conf` file. In our example, on the `Local2` machine, we added the following line in `/etc/rc.conf`:

```
defaultrouter="10.20.30.1"
```

It is also possible to do it directly from the command line with the route(8) command:

```
# route add default 10.20.30.1
```

For more information on manual manipulation of network routing tables, consult route(8) manual page.

31.2.3 Dual Homed Hosts

There is one other type of configuration that we should cover, and that is a host that sits on two different networks. Technically, any machine functioning as a gateway (in the example above, using a PPP connection) counts as a dual-homed host. But the term is really only used to refer to a machine that sits on two local-area networks.

In one case, the machine has two Ethernet cards, each having an address on the separate subnets. Alternately, the machine may only have one Ethernet card, and be using ifconfig(8) aliasing. The former is used if two physically separate Ethernet networks are in use, the latter if there is one physical network segment, but two logically separate subnets.

Either way, routing tables are set up so that each subnet knows that this machine is the defined gateway (inbound route) to the other subnet. This configuration, with the machine acting as a router between the two subnets, is often used when we need to implement packet filtering or firewall security in either or both directions.

If you want this machine to actually forward packets between the two interfaces, you need to tell FreeBSD to enable this ability. See the next section for more details on how to do this.

31.2.4 Building a Router

A network router is simply a system that forwards packets from one interface to another. Internet standards and good engineering practice prevent the FreeBSD Project from enabling this by default in FreeBSD. You can enable this feature by changing the following variable to `YES` in rc.conf(5):

```
gateway_enable="YES"          # Set to YES if this host will be a gateway
```

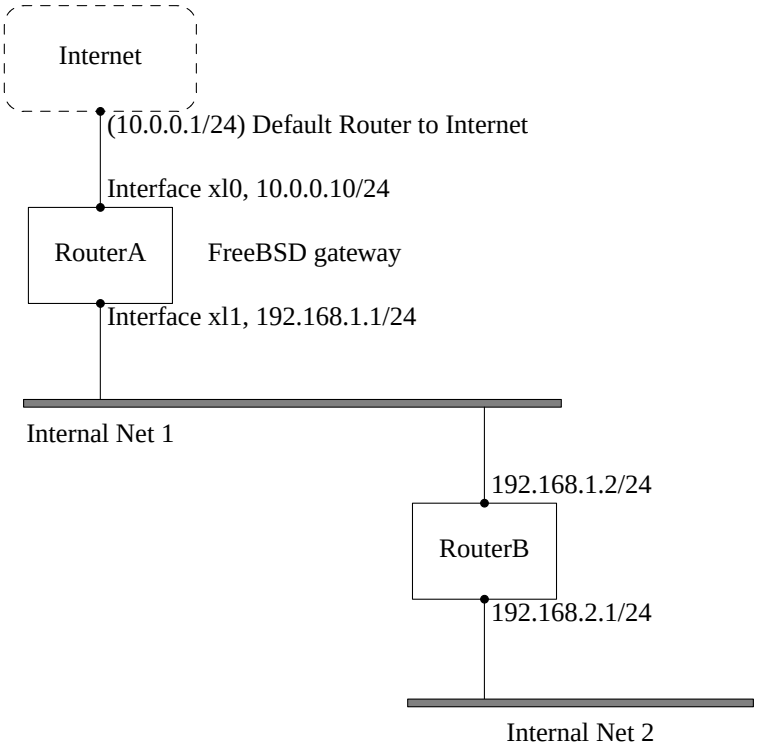
This option will set the sysctl(8) variable `net.inet.ip.forwarding` to `1`. If you should need to stop routing temporarily, you can reset this to `0` temporarily.

Your new router will need routes to know where to send the traffic. If your network is simple enough you can use static routes. FreeBSD also comes with the standard BSD routing daemon routed(8), which speaks RIP (both version 1 and version 2) and IRDP. Support for BGP v4, OSPF v2, and other sophisticated routing protocols is available with the `net/zebra` package. Commercial products such as **GateD®** are also available for more complex network routing solutions.

31.2.5 Setting Up Static Routes

31.2.5.1 Manual Configuration

Let us assume we have a network as follows:



In this scenario, `RouterA` is our FreeBSD machine that is acting as a router to the rest of the Internet. It has a default route set to `10.0.0.1` which allows it to connect with the outside world. We will assume that `RouterB` is already configured properly and knows how to get wherever it needs to go. (This is simple in this picture. Just add a default route on `RouterB` using `192.168.1.1` as the gateway.)

If we look at the routing table for `RouterA` we would see something like the following:

```
% netstat -nr
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif  Expire
default            10.0.0.1           UGS         0    49378    xl0
127.0.0.1          127.0.0.1          UH          0        6    lo0
10.0.0/24          link#1             UC          0        0    xl0
192.168.1/24       link#2             UC          0        0    xl1
```

With the current routing table `RouterA` will not be able to reach our Internal Net 2. It does not have a route for `192.168.2.0/24`. One way to alleviate this is to manually add the route. The following command would add the Internal Net 2 network to `RouterA`'s routing table using `192.168.1.2` as the next hop:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Now RouterA can reach any hosts on the 192.168.2.0/24 network.

31.2.5.2 Persistent Configuration

The above example is perfect for configuring a static route on a running system. However, one problem is that the routing information will not persist if you reboot your FreeBSD machine. The way to handle the addition of a static route is to put it in your /etc/rc.conf file:

```
# Add Internal Net 2 as a static route
static_routes="internalnet2"
route_internalnet2="-net 192.168.2.0/24 192.168.1.2"
```

The static_routes configuration variable is a list of strings separated by a space. Each string references to a route name. In our above example we only have one string in static_routes. This string is *internalnet2*. We then add a configuration variable called route_*internalnet2* where we put all of the configuration parameters we would give to the route(8) command. For our example above we would have used the command:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

so we need "-net 192.168.2.0/24 192.168.1.2".

As said above, we can have more than one string in static_routes. This allows us to create multiple static routes. The following lines shows an example of adding static routes for the 192.168.0.0/24 and 192.168.1.0/24 networks on an imaginary router:

```
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 Routing Propagation

We have already talked about how we define our routes to the outside world, but not about how the outside world finds us.

We already know that routing tables can be set up so that all traffic for a particular address space (in our examples, a class-C subnet) can be sent to a particular host on that network, which will forward the packets inbound.

When you get an address space assigned to your site, your service provider will set up their routing tables so that all traffic for your subnet will be sent down your PPP link to your site. But how do sites across the country know to send to your ISP?

There is a system (much like the distributed DNS information) that keeps track of all assigned address-spaces, and defines their point of connection to the Internet Backbone. The “Backbone” are the main trunk lines that carry Internet traffic across the country, and around the world. Each backbone machine has a copy of a master set of tables, which direct traffic for a particular network to a specific backbone carrier, and from there down the chain of service providers until it reaches your network.

It is the task of your service provider to advertise to the backbone sites that they are the point of connection (and thus the path inward) for your site. This is known as route propagation.

31.2.7 Troubleshooting

Sometimes, there is a problem with routing propagation, and some sites are unable to connect to you. Perhaps the most useful command for trying to figure out where routing is breaking down is the traceroute(8) command. It is equally useful if you cannot seem to make a connection to a remote machine (i.e. ping(8) fails).

The traceroute(8) command is run with the name of the remote host you are trying to connect to. It will show the gateway hosts along the path of the attempt, eventually either reaching the target host, or terminating because of a lack of connection.

For more information, see the manual page for traceroute(8).

31.2.8 Multicast Routing

FreeBSD supports both multicast applications and multicast routing natively. Multicast applications do not require any special configuration of FreeBSD; applications will generally run out of the box. Multicast routing requires that support be compiled into the kernel:

```
options MROUTING
```

In addition, the multicast routing daemon, mrouted(8) must be configured to set up tunnels and DVMRP via `/etc/mrouted.conf`. More details on multicast configuration may be found in the manual page for mrouted(8).

> **Note:** The mrouted(8) multicast routing daemon implements the DVMRP multicast routing protocol, which has largely been replaced by pim(4) in many multicast installations. mrouted(8) and the related map-mbone(8) and mrinfo(8) utilities are available in the FreeBSD Ports Collection as `net/mrouted`.

31.3 Wireless Networking

31.3.1 Wireless Networking Basics

Most wireless networks are based on the IEEE 802.11 standards. A basic wireless network consists of multiple stations communicating with radios that broadcast in either the 2.4GHz or 5GHz band (though this varies according to the locale and is also changing to enable communication in the 2.3GHz and 4.9GHz ranges).

802.11 networks are organized in two ways: in *infrastructure mode* one station acts as a master with all the other stations associating to it; the network is known as a BSS and the master station is termed an access point (AP). In a BSS all communication passes through the AP; even when one station wants to communicate with another wireless station messages must go through the AP. In the second form of network there is no master and stations communicate directly. This form of network is termed an IBSS and is commonly known as an *ad-hoc network*.

802.11 networks were first deployed in the 2.4GHz band using protocols defined by the IEEE 802.11 and 802.11b standard. These specifications include the operating frequencies, MAC layer characteristics including framing and transmission rates (communication can be done at various rates). Later the 802.11a standard defined operation in the 5GHz band, including different signalling mechanisms and higher transmission rates. Still later the 802.11g standard was defined to enable use of 802.11a signalling and transmission mechanisms in the 2.4GHz band in such a way as to be backwards compatible with 802.11b networks.

Separate from the underlying transmission techniques 802.11 networks have a variety of security mechanisms. The original 802.11 specifications defined a simple security protocol called WEP. This protocol uses a fixed pre-shared key and the RC4 cryptographic cipher to encode data transmitted on a network. Stations must all agree on the fixed key in order to communicate. This scheme was shown to be easily broken and is now rarely used except to discourage transient users from joining networks. Current security practice is given by the IEEE 802.11i specification that defines new cryptographic ciphers and an additional protocol to authenticate stations to an access point and exchange keys for doing data communication. Further, cryptographic keys are periodically refreshed and there are mechanisms for detecting intrusion attempts (and for countering intrusion attempts). Another security protocol specification commonly used in wireless networks is termed WPA. This was a precursor to 802.11i defined by an industry group as an interim measure while waiting for 802.11i to be ratified. WPA specifies a subset of the requirements found in 802.11i and is designed for implementation on legacy hardware. Specifically WPA requires only the TKIP cipher that is derived from the original WEP cipher. 802.11i permits use of TKIP but also requires support for a stronger cipher, AES-CCM, for encrypting data. (The AES cipher was not required in WPA because it was deemed too computationally costly to be implemented on legacy hardware.)

Other than the above protocol standards the other important standard to be aware of is 802.11e. This defines protocols for deploying multi-media applications such as streaming video and voice over IP (VoIP) in an 802.11 network. Like 802.11i, 802.11e also has a precursor specification termed WME (later renamed WMM) that has been defined by an industry group as a subset of 802.11e that can be deployed now to enable multi-media applications while waiting for the final ratification of 802.11e. The most important thing to know about 802.11e and WME/WMM is that it enables prioritized traffic use of a wireless network through Quality of Service (QoS) protocols and enhanced media access protocols. Proper implementation of these protocols enable high speed bursting of data and prioritized traffic flow.

FreeBSD supports networks that operate using 802.11a, 802.11b, and 802.11g. The WPA and 802.11i security protocols are likewise supported (in conjunction with any of 11a, 11b, and 11g) and QoS and traffic prioritization required by the WME/WMM protocols are supported for a limited set of wireless devices.

31.3.2 Basic Setup

31.3.2.1 Kernel Configuration

To use wireless networking, you need a wireless networking card and to configure the kernel with the appropriate wireless networking support. The latter is separated into multiple modules so that you only need to configure the software you are actually going to use.

The first thing you need is a wireless device. The most commonly used devices are those that use parts made by Atheros. These devices are supported by the ath(4) driver and require the following line to be added to the `/boot/loader.conf` file:

```
if_ath_load="YES"
```

The Atheros driver is split up into three separate pieces: the proper driver (ath(4)), the hardware support layer that handles chip-specific functions (ath_hal(4)), and an algorithm for selecting which of several possible rates for transmitting frames (ath_rate_sample here). When this support is loaded as kernel modules, these dependencies are automatically handled for you. If, instead of an Atheros device, you had another device you would select the module for that device; e.g.:

```
if_wi_load="YES"
```

for devices based on the Intersil Prism parts (wi(4) driver).

Note: In the rest of this document, we will use an ath(4) device, the device name in the examples must be changed according to your configuration. A list of available wireless drivers and supported adapters can be found in the FreeBSD Hardware Notes. Copies of these notes for various releases and architectures are available on the Release Information (http://www.FreeBSD.org/releases/index.html) page of the FreeBSD Web site. If a native FreeBSD driver for your wireless device does not exist, it may be possible to directly use the Windows driver with the help of the NDIS driver wrapper.

Under FreeBSD 7.X, with a device driver you need to also bring in the 802.11 networking support required by the driver. For the ath(4) driver these are at least the wlan(4), `wlan_scan_ap` and `wlan_scan_sta` modules; the wlan(4) module is automatically loaded with the wireless device driver, the remaining modules must be loaded at boot time via the `/boot/loader.conf` file:

```
wlan_scan_ap_load="YES"
wlan_scan_sta_load="YES"
```

Since FreeBSD 8.0, these modules are part of the base wlan(4) driver which is dynamically loaded with the adapter driver.

With that, you will need the modules that implement cryptographic support for the security protocols you intend to use. These are intended to be dynamically loaded on demand by the wlan(4) module but for now they must be manually configured. The following modules are available: wlan_wep(4), wlan_ccmp(4) and wlan_tkip(4). Both wlan_ccmp(4) and wlan_tkip(4) drivers are only needed if you intend to use the WPA and/or 802.11i security protocols. If your network does not use encryption, you will not need wlan_wep(4) support. To load these modules at boot time, add the following lines to `/boot/loader.conf`:

```
wlan_wep_load="YES"
wlan_ccmp_load="YES"
wlan_tkip_load="YES"
```

With this information in the system bootstrap configuration file (i.e., `/boot/loader.conf`), you have to reboot your FreeBSD box. If you do not want to reboot your machine for the moment, you can load the modules by hand using kldload(8).

Note: If you do not want to use modules, it is possible to compile these drivers into the kernel by adding the following lines to your kernel configuration file:

```
device wlan              # 802.11 support
device wlan_wep          # 802.11 WEP support
device wlan_ccmp         # 802.11 CCMP support
device wlan_tkip         # 802.11 TKIP support
device wlan_amrr         # AMRR transmit rate control algorithm
device ath               # Atheros pci/cardbus NIC's
device ath_hal           # pci/cardbus chip support
options AH_SUPPORT_AR5416 # enable AR5416 tx/rx descriptors
device ath_rate_sample   # SampleRate tx rate control for ath
```

Both following lines are also required by FreeBSD 7.X, other FreeBSD versions do not need them:

```
device wlan_scan_ap      # 802.11 AP mode scanning
device wlan_scan_sta     # 802.11 STA mode scanning
```

With this information in the kernel configuration file, recompile the kernel and reboot your FreeBSD machine.

When the system is up, we could find some information about the wireless device in the boot messages, like this:

```
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 Infrastructure Mode

The infrastructure mode or BSS mode is the mode that is typically used. In this mode, a number of wireless access points are connected to a wired network. Each wireless network has its own name, this name is called the SSID of the network. Wireless clients connect to the wireless access points.

31.3.3.1 FreeBSD Clients

31.3.3.1.1 How to Find Access Points

To scan for networks, use the `ifconfig` command. This request may take a few moments to complete as it requires that the system switches to each available wireless frequency and probes for available access points. Only the super-user can initiate such a scan:

```
# 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
```

> **Note:** You must mark the interface `up` before you can scan. Subsequent scan requests do not require you to mark the interface up again.

> **Note:** Under FreeBSD 7.X, the adapter device, for example *`ath0`*, is used directly instead of the `wlan0` device. This requires you to replace the both previous lines with:
>
> ```
> # ifconfig ath0 up scan
> ```
>
> In the rest of this document, FreeBSD 7.X users will need to change the command and configuration lines according to that scheme.

The output of a scan request lists each BSS/IBSS network found. Beside the name of the network, `SSID`, we find the `BSSID` which is the MAC address of the access point. The `CAPS` field identifies the type of each network and the capabilities of the stations operating there:

`E`

Extended Service Set (ESS). Indicates that the station is part of an infrastructure network (in contrast to an IBSS/ad-hoc network).

`I`

IBSS/ad-hoc network. Indicates that the station is part of an ad-hoc network (in contrast to an ESS network).

`P`

Privacy. Data confidentiality is required for all data frames exchanged within the BSS. This means that this BSS requires the station to use cryptographic means such as WEP, TKIP or AES-CCMP to encrypt/decrypt data frames being exchanged with others.

`S`

Short Preamble. Indicates that the network is using short preambles (defined in 802.11b High Rate/DSSS PHY, short preamble utilizes a 56 bit sync field in contrast to a 128 bit field used in long preamble mode).

`s`

Short slot time. Indicates that the 802.11g network is using a short slot time because there are no legacy (802.11b) stations present.

One can also display the current list of known networks with:

```
# ifconfig wlan0 list scan
```

This information may be updated automatically by the adapter or manually with a `scan` request. Old data is automatically removed from the cache, so over time this list may shrink unless more scans are done.

31.3.3.1.2 Basic Settings

This section provides a simple example of how to make the wireless network adapter work in FreeBSD without encryption. After you are familiar with these concepts, we strongly recommend using WPA to set up your wireless network.

There are three basic steps to configure a wireless network: selecting an access point, authenticating your station, and configuring an IP address. The following sections discuss each step.

31.3.3.1.2.1 Selecting an Access Point

Most of time it is sufficient to let the system choose an access point using the builtin heuristics. This is the default behaviour when you mark an interface up or otherwise configure an interface by listing it in `/etc/rc.conf`, e.g.:

```
wlans_ath0="wlan0"
ifconfig_wlan0="DHCP"
```

> **Note:** As previously mentioned, FreeBSD 7.X will only require a line related to the adapter device:
>
> ```
> ifconfig_ath0="DHCP"
> ```

If there are multiple access points and you want to select a specific one, you can select it by its SSID:

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid your_ssid_here DHCP"
```

In an environment where there are multiple access points with the same SSID (often done to simplify roaming) it may be necessary to associate to one specific device. In this case you can also specify the BSSID of the access point (you can also leave off the SSID):

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid your_ssid_here bssid xx:xx:xx:xx:xx:xx DHCP"
```

There are other ways to constrain the choice of an access point such as limiting the set of frequencies the system will scan on. This may be useful if you have a multi-band wireless card as scanning all the possible channels can be time-consuming. To limit operation to a specific band you can use the `mode` parameter; e.g.:

```
wlans_ath0="wlan0"
ifconfig_wlan0="mode 11g ssid your_ssid_here DHCP"
```

will force the card to operate in 802.11g which is defined only for 2.4GHz frequencies so any 5GHz channels will not be considered. Other ways to do this are the `channel` parameter, to lock operation to one specific frequency, and the `chanlist` parameter, to specify a list of channels for scanning. More information about these parameters can be found in the ifconfig(8) manual page.

31.3.3.1.2.2 Authentication

Once you have selected an access point your station needs to authenticate before it can pass data. Authentication can happen in several ways. The most common scheme used is termed open authentication and allows any station to join the network and communicate. This is the authentication you should use for test purpose the first time you set up a wireless network. Other schemes require cryptographic handshakes be completed before data traffic can flow; either using pre-shared keys or secrets, or more complex schemes that involve backend services such as RADIUS. Most users will use open authentication which is the default setting. Next most common setup is WPA-PSK, also known as WPA Personal, which is described below.

> **Note:** If you have an Apple AirPort® Extreme base station for an access point you may need to configure shared-key authentication together with a WEP key. This can be done in the `/etc/rc.conf` file or using the wpa_supplicant(8) program. If you have a single AirPort base station you can setup access with something like:
>
> ```
> wlans_ath0="wlan0"
> ifconfig_wlan0="authmode shared wepmode on weptxkey 1 wepkey 01234567 DHCP"
> ```
>
> In general shared key authentication is to be avoided because it uses the WEP key material in a highly-constrained manner making it even easier to crack the key. If WEP must be used (e.g., for compatibility with legacy devices) it is better to use WEP with `open` authentication. More information regarding WEP can be found in the Section 31.3.3.1.4.

31.3.3.1.2.3 Getting an IP Address with DHCP

Once you have selected an access point and set the authentication parameters, you will have to get an IP address to communicate. Most of time you will obtain your wireless IP address via DHCP. To achieve that, simply edit `/etc/rc.conf` and add `DHCP` to the configuration for your device as shown in various examples above:

```
wlans_ath0="wlan0"
ifconfig_wlan0="DHCP"
```

At this point, you are ready to bring up the wireless interface:

```
# /etc/rc.d/netif start
```

Once the interface is running, use `ifconfig` to see the status of the interface `ath0`:

```
# 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 0xffffff00 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
```

The `status: associated` means you are connected to the wireless network (to the `dlinkap` network in our case). The `bssid 00:13:46:49:41:76` part is the MAC address of your access point; the `authmode OPEN` part informs you that the communication is not encrypted.

31.3.3.1.2.4 Static IP Address

In the case you cannot obtain an IP address from a DHCP server, you can set a fixed IP address. Replace the `DHCP` keyword shown above with the address information. Be sure to retain any other parameters you have set up for selecting an access point:

```
wlans_ath0="wlan0"
ifconfig_wlan0="inet 192.168.1.100 netmask 255.255.255.0 ssid your_ssid_here"
```

31.3.3.1.3 WPA

WPA (Wi-Fi Protected Access) is a security protocol used together with 802.11 networks to address the lack of proper authentication and the weakness of WEP. WPA leverages the 802.1X authentication protocol and uses one of several ciphers instead of WEP for data integrity. The only cipher required by WPA is TKIP (Temporary Key Integrity Protocol) which is a cipher that extends the basic RC4 cipher used by WEP by adding integrity checking, tamper detection, and measures for responding to any detected intrusions. TKIP is designed to work on legacy hardware with only software modification; it represents a compromise that improves security but is still not entirely immune to attack. WPA also specifies the AES-CCMP cipher as an alternative to TKIP and that is preferred when possible; for this specification the term WPA2 (or RSN) is commonly used.

WPA defines authentication and encryption protocols. Authentication is most commonly done using one of two techniques: by 802.1X and a backend authentication service such as RADIUS, or by a minimal handshake between the station and the access point using a pre-shared secret. The former is commonly termed WPA Enterprise with the latter known as WPA Personal. Since most people will not set up a RADIUS backend server for wireless network, WPA-PSK is by far the most commonly encountered configuration for WPA.

The control of the wireless connection and the authentication (key negotiation or authentication with a server) is done with the wpa_supplicant(8) utility. This program requires a configuration file, `/etc/wpa_supplicant.conf`, to run. More information regarding this file can be found in the wpa_supplicant.conf(5) manual page.

31.3.3.1.3.1 WPA-PSK

WPA-PSK also known as WPA-Personal is based on a pre-shared key (PSK) generated from a given password and that will be used as the master key in the wireless network. This means every wireless user will share the same key. WPA-PSK is intended for small networks where the use of an authentication server is not possible or desired.

> **Warning:** Always use strong passwords that are sufficiently long and made from a rich alphabet so they will not be guessed and/or attacked.

The first step is the configuration of the `/etc/wpa_supplicant.conf` file with the SSID and the pre-shared key of your network:

```
network={
  ssid="freebsdap"
  psk="freebsdmall"
}
```

Then, in `/etc/rc.conf`, we indicate that the wireless device configuration will be done with WPA and the IP address will be obtained with DHCP:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Then, we can bring up the interface:

```
# /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 0xffffff00 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
```

Or you can try to configure it manually using the same `/etc/wpa_supplicant.conf` above, and run:

```
# 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=]
```

The next operation is the launch of the `dhclient` command to get the IP address from the DHCP server:

```
# 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
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 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
```

Note: If the `/etc/rc.conf` is set up with the line `ifconfig_wlan0="DHCP"` then it is no need to run the `dhclient` command manually, `dhclient` will be launched after `wpa_supplicant` plumbs the keys.

In the case where the use of DHCP is not possible, you can set a static IP address after `wpa_supplicant` has authenticated the station:

```
# ifconfig wlan0 inet 192.168.0.100 netmask 255.255.255.0
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.100 netmask 0xffffff00 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
```

When DHCP is not used, you also have to manually set up the default gateway and the nameserver:

```
# route add default your_default_router
```

```
# echo "nameserver your_DNS_server" >> /etc/resolv.conf
```

31.3.3.1.3.2 WPA with EAP-TLS

The second way to use WPA is with an 802.1X backend authentication server, in this case WPA is called WPA-Enterprise to make difference with the less secure WPA-Personal with its pre-shared key. The authentication in WPA-Enterprise is based on EAP (Extensible Authentication Protocol).

EAP does not come with an encryption method, it was decided to embed EAP inside an encrypted tunnel. Many types of EAP authentication methods have been designed, the most common methods are EAP-TLS, EAP-TTLS and EAP-PEAP.

EAP-TLS (EAP with Transport Layer Security) is a very well-supported authentication protocol in the wireless world since it was the first EAP method to be certified by the Wi-Fi alliance (http://www.wi-fi.org/). EAP-TLS will require three certificates to run: the CA certificate (installed on all machines), the server certificate for your authentication server, and one client certificate for each wireless client. In this EAP method, both authentication server and wireless client authenticate each other in presenting their respective certificates, and they verify that these certificates were signed by your organization's certificate authority (CA).

As previously, the configuration is done via `/etc/wpa_supplicant.conf`:

```
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 This field indicates the network name (SSID).

❷ Here, we use RSN (IEEE 802.11i) protocol, i.e., WPA2.

❸ The `key_mgmt` line refers to the key management protocol we use. In our case it is WPA using EAP authentication: `WPA-EAP`.

❹ In this field, we mention the EAP method for our connection.

❺ The `identity` field contains the identity string for EAP.

❻ The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificate.

❼ The `client_cert` line gives the pathname to the client certificate file. This certificate is unique to each wireless client of the network.

❽ The `private_key` field is the pathname to the client certificate private key file.

❾ The `private_key_passwd` field contains the passphrase for the private key.

Then add the following lines to `/etc/rc.conf`:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

The next step is to bring up the interface with the help of the `rc.d` facility:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 15
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 0xffffff00 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
```

As previously shown, it is also possible to bring up the interface manually with both `wpa_supplicant` and `ifconfig` commands.

31.3.3.1.3.3 WPA with EAP-TTLS

With EAP-TLS both the authentication server and the client need a certificate, with EAP-TTLS (EAP-Tunneled Transport Layer Security) a client certificate is optional. This method is close to what some secure web sites do , where the web server can create a secure SSL tunnel even if the visitors do not have client-side certificates. EAP-TTLS will use the encrypted TLS tunnel for safe transport of the authentication data.

The configuration is done via the `/etc/wpa_supplicant.conf` file:

```
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 In this field, we mention the EAP method for our connection.

❷ The `identity` field contains the identity string for EAP authentication inside the encrypted TLS tunnel.

❸ The `password` field contains the passphrase for the EAP authentication.

❹ The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificat.

❺ In this field, we mention the authentication method used in the encrypted TLS tunnel. In our case, EAP with MD5-Challenge has been used. The “inner authentication” phase is often called “phase2”.

You also have to add the following lines to `/etc/rc.conf`:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

The next step is to bring up the interface:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 15
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 21
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 0xffffff00 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 with EAP-PEAP

PEAP (Protected EAP) has been designed as an alternative to EAP-TTLS. There are two types of PEAP methods, the most common one is PEAPv0/EAP-MSCHAPv2. In the rest of this document, we will use the PEAP term to refer to that EAP method. PEAP is the most used EAP standard after EAP-TLS, in other words if you have a network with mixed OSes, PEAP should be the most supported standard after EAP-TLS.

PEAP is similar to EAP-TTLS: it uses a server-side certificate to authenticate clients by creating an encrypted TLS tunnel between the client and the authentication server, which protects the ensuing exchange of authentication information. In term of security the difference between EAP-TTLS and PEAP is that PEAP authentication broadcasts the username in clear, only the password is sent in the encrypted TLS tunnel. EAP-TTLS will use the TLS tunnel for both username and password.

We have to edit the `/etc/wpa_supplicant.conf` file and add the EAP-PEAP related settings:

```
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 In this field, we mention the EAP method for our connection.

❷ The `identity` field contains the identity string for EAP authentication inside the encrypted TLS tunnel.

❸ The `password` field contains the passphrase for the EAP authentication.

❹ The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificat.

❺ This field contains the parameters for the first phase of the authentication (the TLS tunnel). According to the authentication server used, you will have to specify a specific label for the authentication. Most of time, the label will be “client EAP encryption” which is set by using `peaplabel=0`. More information can be found in the wpa_supplicant.conf(5) manual page.

❻ In this field, we mention the authentication protocol used in the encrypted TLS tunnel. In the case of PEAP, it is `auth=MSCHAPV2`.

The following must be added to `/etc/rc.conf`:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Then, we can bring up the interface:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 15
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 21
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 0xffffff00 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

WEP (Wired Equivalent Privacy) is part of the original 802.11 standard. There is no authentication mechanism, only a weak form of access control, and it is easily to be cracked.

WEP can be set up with `ifconfig`:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 inet 192.168.1.100 netmask 255.255.255.0 \
```

```
ssid my_net wepmode on weptxkey 3 wepkey 3:0x3456789012
```

- The `weptxkey` means which WEP key will be used in the transmission. Here we used the third key. This must match the setting in the access point. If you do not have any idea of what is the key used by the access point, you should try to use `1` (i.e., the first key) for this value.
- The `wepkey` means setting the selected WEP key. It should in the format *`index:key`*, if the index is not given, key `1` is set. That is to say we need to set the index if we use keys other than the first key.

> **Note:** You must replace the `0x3456789012` with the key configured for use on the access point.

You are encouraged to read ifconfig(8) manual page for further information.

The `wpa_supplicant` facility also can be used to configure your wireless interface with WEP. The example above can be set up by adding the following lines to `/etc/wpa_supplicant.conf`:

```
network={
  ssid="my_net"
  key_mgmt=NONE
  wep_key3=3456789012
  wep_tx_keyidx=3
}
```

Then:

```
# 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 Ad-hoc Mode

IBSS mode, also called ad-hoc mode, is designed for point to point connections. For example, to establish an ad-hoc network between the machine `A` and the machine `B` we will just need to choose two IP addresses and a SSID.

On the box `A`:

```
# 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 0xffffff00 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
```

The adhoc parameter indicates the interface is running in the IBSS mode.

On B, we should be able to detect A:

```
# ifconfig wlan0 create wlandev ath0 wlanmode adhoc
# ifconfig wlan0 up scan
  SSID/MESH ID    BSSID              CHAN RATE   S:N     INT CAPS
  freebsdap       02:11:95:c3:0d:ac    2   54M -64:-96  100 IS   WME
```

The I in the output confirms the machine A is in ad-hoc mode. We just have to configure B with a different IP address:

```
# 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 0xffffff00 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
```

Both A and B are now ready to exchange informations.

31.3.5 FreeBSD Host Access Points

FreeBSD can act as an Access Point (AP) which eliminates the need to buy a hardware AP or run an ad-hoc network. This can be particularly useful when your FreeBSD machine is acting as a gateway to another network (e.g., the Internet).

31.3.5.1 Basic Settings

Before configuring your FreeBSD machine as an AP, the kernel must be configured with the appropriate wireless networking support for your wireless card. You also have to add the support for the security protocols you intend to use. For more details, see Section 31.3.2.

> **Note:** The use of the NDIS driver wrapper and the Windows drivers do not allow currently the AP operation. Only native FreeBSD wireless drivers support AP mode.

Once the wireless networking support is loaded, you can check if your wireless device supports the host-based access point mode (also know as hostap mode):

```
# 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>
```

This output displays the card capabilities; the HOSTAP word confirms this wireless card can act as an Access Point. Various supported ciphers are also mentioned: WEP, TKIP, AES, etc., these informations are important to know what security protocols could be set on the Access Point.

The wireless device can only be put into hostap mode during the creation of the network pseudo-device, so a previously created device must be destroyed first:

```
# ifconfig wlan0 destroy
```

then regenerated with the correct option before setting the other parameters:

```
# 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
```

Use again ifconfig to see the status of the wlan0 interface:

```
# 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 0xffffff00 broadcast 192.168.0.255
   media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
   status: running
   ssid freebsdap channel 1 (2412 Mhz 11g) 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
```

The hostap parameter indicates the interface is running in the host-based access point mode.

The interface configuration can be done automatically at boot time by adding the following lines to /etc/rc.conf:

```
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 Host-based Access Point without Authentication or Encryption

Although it is not recommended to run an AP without any authentication or encryption, this is a simple way to check if your AP is working. This configuration is also important for debugging client issues.

Once the AP configured as previously shown, it is possible from another wireless machine to initiate a scan to find the AP:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 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
```

The client machine found the Access Point and can be associated with it:

```
# 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 0xffffff00 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 Host-based Access Point

This section will focus on setting up FreeBSD Access Point using the WPA security protocol. More details regarding WPA and the configuration of WPA-based wireless clients can be found in the Section 31.3.3.1.3.

The **hostapd** daemon is used to deal with client authentication and keys management on the WPA enabled Access Point.

In the following, all the configuration operations will be performed on the FreeBSD machine acting as AP. Once the AP is correctly working, **hostapd** should be automatically enabled at boot with the following line in `/etc/rc.conf`:

```
hostapd_enable="YES"
```

Before trying to configure **hostapd**, be sure you have done the basic settings introduced in the Section 31.3.5.1.

31.3.5.3.1 WPA-PSK

WPA-PSK is intended for small networks where the use of an backend authentication server is not possible or desired.

The configuration is done in the `/etc/hostapd.conf` file:

```
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 This field indicates the wireless interface used for the Access Point.

❷ This field sets the level of verbosity during the execution of **hostapd**. A value of `1` represents the minimal level.

❸ The `ctrl_interface` field gives the pathname of the directory used by **hostapd** to stores its domain socket files for the communication with external programs such as hostapd_cli(8). The default value is used here.

❹ The `ctrl_interface_group` line sets the group (here, it is the `wheel` group) allowed to access to the control interface files.

❺ This field sets the network name.

❻ The `wpa` field enables WPA and specifies which WPA authentication protocol will be required. A value of `1` configures the AP for WPA-PSK.

❼ The `wpa_passphrase` field contains the ASCII passphrase for the WPA authentication.

Warning: Always use strong passwords that are sufficiently long and made from a rich alphabet so they will not be guessed and/or attacked.

❽ The `wpa_key_mgmt` line refers to the key management protocol we use. In our case it is WPA-PSK.

❾ The `wpa_pairwise` field indicates the set of accepted encryption algorithms by the Access Point. Here both TKIP (WPA) and CCMP (WPA2) ciphers are accepted. CCMP cipher is an alternative to TKIP and that is strongly preferred when possible; TKIP should be used solely for stations incapable of doing CCMP.

The next step is to start **hostapd**:

```
# /etc/rc.d/hostapd forcestart

# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2290
   inet 192.168.0.1 netmask 0xffffff00 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 dtimper
```

The Access Point is running, the clients can now be associated with it, see Section 31.3.3.1.3 for more details. It is possible to see the stations associated with the AP using the `ifconfig wlan0 list sta` command.

31.3.5.4 WEP Host-based Access Point

It is not recommended to use WEP for setting up an Access Point since there is no authentication mechanism and it is easily to be cracked. Some legacy wireless cards only support WEP as security protocol, these cards will only allow to set up AP without authentication or encryption or using the WEP protocol.

The wireless device can now be put into hostap mode and configured with the correct SSID and IP address:

```
# 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
```

- The `weptxkey` means which WEP key will be used in the transmission. Here we used the third key (note that the key numbering starts with `1`). This parameter must be specified to really encrypt the data.
- The `wepkey` means setting the selected WEP key. It should in the format *`index:key`*, if the index is not given, key `1` is set. That is to say we need to set the index if we use keys other than the first key.

Use again `ifconfig` to see the status of the `wlan0` interface:

```
# 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 0xffffff00 broadcast 192.168.0.255
```

```
media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
status: running
ssid freebsdap channel 4 (2427 Mhz 11g) 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
```

From another wireless machine, it is possible to initiate a scan to find the AP:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 up scan
SSID            BSSID              CHAN RATE  S:N   INT CAPS
freebsdap       00:11:95:c3:0d:ac    1   54M 22:1   100 EPS
```

The client machine found the Access Point and can be associated with it using the correct parameters (key, etc.), see Section 31.3.3.1.4 for more details.

31.3.6 Using both wired and wireless connection

Wired connection provides better performance and reliability, while wireless connection provides flexibility and mobility, users of laptop computers usually want to combine these together and roam seamlessly between the two.

On FreeBSD, it is possible to combine two or even more network interfaces together in a “failover” fashion, that is, to use the most preferred and available connection from a group of network interfaces, and have the operating system to switch automatically when the link state changes.

We will cover link aggregation and failover in Section 31.6 where an example for using both wired and wireless connection is also provided at Example 31-3.

31.3.7 Troubleshooting

If you are having trouble with wireless networking, there are a number of steps you can take to help troubleshoot the problem.

- If you do not see the access point listed when scanning be sure you have not configured your wireless device to a limited set of channels.
- If you cannot associate to an access point verify the configuration of your station matches the one of the access point. This includes the authentication scheme and any security protocols. Simplify your configuration as much as possible. If you are using a security protocol such as WPA or WEP configure the access point for open authentication and no security to see if you can get traffic to pass.
- Once you can associate to the access point diagnose any security configuration using simple tools like ping(8).

  The `wpa_supplicant` has much debugging support; try running it manually with the `-dd` option and look at the system logs.
- There are also many lower-level debugging tools. You can enable debugging messages in the 802.11 protocol support layer using the `wlandebug` program found in `/usr/src/tools/tools/net80211`. For example:

  ```
  # wlandebug -i ath0 +scan+auth+debug+assoc
    net.wlan.0.debug: 0 => 0xc80000<assoc,auth,scan>
  ```

can be used to enable console messages related to scanning for access points and doing the 802.11 protocol handshakes required to arrange communication.

There are also many useful statistics maintained by the 802.11 layer; the `wlanstats` tool will dump these informations. These statistics should identify all errors identified by the 802.11 layer. Beware however that some errors are identified in the device drivers that lie below the 802.11 layer so they may not show up. To diagnose device-specific problems you need to refer to the drivers' documentation.

If the above information does not help to clarify the problem, please submit a problem report and include output from the above tools.

31.4 Bluetooth

31.4.1 Introduction

Bluetooth is a wireless technology for creating personal networks operating in the 2.4 GHz unlicensed band, with a range of 10 meters. Networks are usually formed ad-hoc from portable devices such as cellular phones, handhelds and laptops. Unlike the other popular wireless technology, Wi-Fi, Bluetooth offers higher level service profiles, e.g. FTP-like file servers, file pushing, voice transport, serial line emulation, and more.

The Bluetooth stack in FreeBSD is implemented using the Netgraph framework (see netgraph(4)). A broad variety of Bluetooth USB dongles is supported by the ng_ubt(4) driver. The Broadcom BCM2033 chip based Bluetooth devices are supported via the ubtbcmfw(4) and ng_ubt(4) drivers. The 3Com Bluetooth PC Card 3CRWB60-A is supported by the ng_bt3c(4) driver. Serial and UART based Bluetooth devices are supported via sio(4), ng_h4(4) and hcseriald(8). This section describes the use of the USB Bluetooth dongle.

31.4.2 Plugging in the Device

By default Bluetooth device drivers are available as kernel modules. Before attaching a device, you will need to load the driver into the kernel:

```
# kldload ng_ubt
```

If the Bluetooth device is present in the system during system startup, load the module from `/boot/loader.conf`:

```
ng_ubt_load="YES"
```

Plug in your USB dongle. The output similar to the following will appear on the console (or in syslog):

```
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
```

The `/etc/rc.d/bluetooth` script is used to start and stop the Bluetooth stack. It is a good idea to stop the stack before unplugging the device, but it is not (usually) fatal. When starting the stack, you will receive output similar to the following:

```
# /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)

Host Controller Interface (HCI) provides a command interface to the baseband controller and link manager, and access to hardware status and control registers. This interface provides a uniform method of accessing the Bluetooth baseband capabilities. HCI layer on the Host exchanges data and commands with the HCI firmware on the Bluetooth hardware. The Host Controller Transport Layer (i.e. physical bus) driver provides both HCI layers with the ability to exchange information with each other.

A single Netgraph node of type *hci* is created for a single Bluetooth device. The HCI node is normally connected to the Bluetooth device driver node (downstream) and the L2CAP node (upstream). All HCI operations must be performed on the HCI node and not on the device driver node. Default name for the HCI node is “devicehci”. For more details refer to the ng_hci(4) manual page.

One of the most common tasks is discovery of Bluetooth devices in RF proximity. This operation is called *inquiry*. Inquiry and other HCI related operations are done with the hccontrol(8) utility. The example below shows how to find out which Bluetooth devices are in range. You should receive the list of devices in a few seconds. Note that a remote device will only answer the inquiry if it put into *discoverable* mode.

```
% 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]
```

`BD_ADDR` is unique address of a Bluetooth device, similar to MAC addresses of a network card. This address is needed for further communication with a device. It is possible to assign human readable name to a BD_ADDR. The `/etc/bluetooth/hosts` file contains information regarding the known Bluetooth hosts. The following example shows how to obtain human readable name that was assigned to the remote device:

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav’s T39
```

If you perform an inquiry on a remote Bluetooth device, it will find your computer as “your.host.name (ubt0)”. The name assigned to the local device can be changed at any time.

The Bluetooth system provides a point-to-point connection (only two Bluetooth units involved), or a point-to-multipoint connection. In the point-to-multipoint connection the connection is shared among several Bluetooth devices. The following example shows how to obtain the list of active baseband connections for the local device:

```
% 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 *connection handle* is useful when termination of the baseband connection is required. Note, that it is normally not required to do it by hand. The stack will automatically terminate inactive baseband connections.

```
# hccontrol -n ubt0hci disconnect 41
Connection handle: 41
Reason: Connection terminated by local host [0x16]
```

Refer to `hccontrol help` for a complete listing of available HCI commands. Most of the HCI commands do not require superuser privileges.

31.4.4 Logical Link Control and Adaptation Protocol (L2CAP)

Logical Link Control and Adaptation Protocol (L2CAP) provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability and segmentation and reassembly operation. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

L2CAP is based around the concept of *channels*. Channel is a logical connection on top of baseband connection. Each channel is bound to a single protocol in a many-to-one fashion. Multiple channels can be bound to the same protocol, but a channel cannot be bound to multiple protocols. Each L2CAP packet received on a channel is directed to the appropriate higher level protocol. Multiple channels can share the same baseband connection.

A single Netgraph node of type *l2cap* is created for a single Bluetooth device. The L2CAP node is normally connected to the Bluetooth HCI node (downstream) and Bluetooth sockets nodes (upstream). Default name for the L2CAP node is “devicel2cap”. For more details refer to the ng_l2cap(4) manual page.

A useful command is l2ping(8), which can be used to ping other devices. Some Bluetooth implementations might not return all of the data sent to them, so `0 bytes` in the following example is normal.

```
# 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
```

The l2control(8) utility is used to perform various operations on L2CAP nodes. This example shows how to obtain the list of logical connections (channels) and the list of baseband connections for the local device:

```
% 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 O           0 OPEN
```

Another diagnostic tool is btsockstat(1). It does a job similar to as netstat(1) does, but for Bluetooth network-related data structures. The example below shows the same logical connection as l2control(8) above.

```
% 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 RFCOMM Protocol

The RFCOMM protocol provides emulation of serial ports over the L2CAP protocol. The protocol is based on the ETSI standard TS 07.10. RFCOMM is a simple transport protocol, with additional provisions for emulating the 9 circuits of RS-232 (EIATIA-232-E) serial ports. The RFCOMM protocol supports up to 60 simultaneous connections (RFCOMM channels) between two Bluetooth devices.

For the purposes of RFCOMM, a complete communication path involves two applications running on different devices (the communication endpoints) with a communication segment between them. RFCOMM is intended to cover applications that make use of the serial ports of the devices in which they reside. The communication segment is a Bluetooth link from one device to another (direct connect).

RFCOMM is only concerned with the connection between the devices in the direct connect case, or between the device and a modem in the network case. RFCOMM can support other configurations, such as modules that communicate via Bluetooth wireless technology on one side and provide a wired interface on the other side.

In FreeBSD the RFCOMM protocol is implemented at the Bluetooth sockets layer.

31.4.6 Pairing of Devices

By default, Bluetooth communication is not authenticated, and any device can talk to any other device. A Bluetooth device (for example, cellular phone) may choose to require authentication to provide a particular service (for example, Dial-Up service). Bluetooth authentication is normally done with *PIN codes*. A PIN code is an ASCII string up to 16 characters in length. User is required to enter the same PIN code on both devices. Once user has entered the PIN code, both devices will generate a *link key*. After that the link key can be stored either in the devices themselves or in a persistent storage. Next time both devices will use previously generated link key. The described above procedure is called *pairing*. Note that if the link key is lost by any device then pairing must be repeated.

The hcsecd(8) daemon is responsible for handling of all Bluetooth authentication requests. The default configuration file is `/etc/bluetooth/hcsecd.conf`. An example section for a cellular phone with the PIN code arbitrarily set to “1234” is shown below:

```
device {
        bdaddr  00:80:37:29:19:a4;
        name    "Pav's T39";
        key     nokey;
        pin     "1234";
      }
```

There is no limitation on PIN codes (except length). Some devices (for example Bluetooth headsets) may have a fixed PIN code built in. The `-d` switch forces the hcsecd(8) daemon to stay in the foreground, so it is easy to see what is happening. Set the remote device to receive pairing and initiate the Bluetooth connection to the remote device. The remote device should say that pairing was accepted, and request the PIN code. Enter the same PIN code as you have in `hcsecd.conf`. Now your PC and the remote device are paired. Alternatively, you can initiate pairing on the remote device.

The following line can be added to the `/etc/rc.conf` file to have **hcsecd** started automatically on system start:

```
hcsecd_enable="YES"
```

The following is a sample of the **hcsecd** daemon output:

```
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 do
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
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39', PIN code ex
hcsecd[16484]: Sending PIN_Code_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
```

31.4.7 Service Discovery Protocol (SDP)

The Service Discovery Protocol (SDP) provides the means for client applications to discover the existence of services provided by server applications as well as the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to utilize the service.

SDP involves communication between a SDP server and a SDP client. The server maintains a list of service records that describe the characteristics of services associated with the server. Each service record contains information about a single service. A client may retrieve information from a service record maintained by the SDP server by issuing a SDP request. If the client, or an application associated with the client, decides to use a service, it must open a separate connection to the service provider in order to utilize the service. SDP provides a mechanism for discovering services and their attributes, but it does not provide a mechanism for utilizing those services.

Normally, a SDP client searches for services based on some desired characteristics of the services. However, there are times when it is desirable to discover which types of services are described by an SDP server's service records without any a priori information about the services. This process of looking for any offered services is called *browsing*.

The Bluetooth SDP server sdpd(8) and command line client sdpcontrol(8) are included in the standard FreeBSD installation. The following example shows how to perform a SDP browse query.

```
% 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
```

... and so on. Note that each service has a list of attributes (RFCOMM channel for example). Depending on the service you might need to make a note of some of the attributes. Some Bluetooth implementations do not support service browsing and may return an empty list. In this case it is possible to search for the specific service. The example below shows how to search for the OBEX Object Push (OPUSH) service:

```
% sdpcontrol -a 00:01:03:fc:6e:ec search OPUSH
```

Offering services on FreeBSD to Bluetooth clients is done with the sdpd(8) server. The following line can be added to the `/etc/rc.conf` file:

```
sdpd_enable="YES"
```

Then the **sdpd** daemon can be started with:

```
# /etc/rc.d/sdpd start
```

The local server application that wants to provide Bluetooth service to the remote clients will register service with the local SDP daemon. The example of such application is rfcomm_pppd(8). Once started it will register Bluetooth LAN service with the local SDP daemon.

The list of services registered with the local SDP server can be obtained by issuing SDP browse query via local control channel:

```
# sdpcontrol -l browse
```

31.4.8 Dial-Up Networking (DUN) and Network Access with PPP (LAN) Profiles

The Dial-Up Networking (DUN) profile is mostly used with modems and cellular phones. The scenarios covered by this profile are the following:

- use of a cellular phone or modem by a computer as a wireless modem for connecting to a dial-up Internet access server, or using other dial-up services;

- use of a cellular phone or modem by a computer to receive data calls.

Network Access with PPP (LAN) profile can be used in the following situations:

- LAN access for a single Bluetooth device;
- LAN access for multiple Bluetooth devices;
- PC to PC (using PPP networking over serial cable emulation).

In FreeBSD both profiles are implemented with ppp(8) and rfcomm_pppd(8) - a wrapper that converts RFCOMM Bluetooth connection into something PPP can operate with. Before any profile can be used, a new PPP label in the `/etc/ppp/ppp.conf` must be created. Consult rfcomm_pppd(8) manual page for examples.

In the following example rfcomm_pppd(8) will be used to open RFCOMM connection to remote device with BD_ADDR 00:80:37:29:19:a4 on DUN RFCOMM channel. The actual RFCOMM channel number will be obtained from the remote device via SDP. It is possible to specify RFCOMM channel by hand, and in this case rfcomm_pppd(8) will not perform SDP query. Use sdpcontrol(8) to find out RFCOMM channel on the remote device.

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

In order to provide Network Access with PPP (LAN) service the sdpd(8) server must be running. A new entry for LAN clients must be created in the `/etc/ppp/ppp.conf` file. Consult rfcomm_pppd(8) manual page for examples. Finally, start RFCOMM PPP server on valid RFCOMM channel number. The RFCOMM PPP server will automatically register Bluetooth LAN service with the local SDP daemon. The example below shows how to start RFCOMM PPP server.

```
# rfcomm_pppd -s -C 7 -l rfcomm-server
```

31.4.9 OBEX Object Push (OPUSH) Profile

OBEX is a widely used protocol for simple file transfers between mobile devices. Its main use is in infrared communication, where it is used for generic file transfers between notebooks or PDAs, and for sending business cards or calendar entries between cellular phones and other devices with PIM applications.

The OBEX server and client are implemented as a third-party package **obexapp**, which is available as `comms/obexapp` port.

OBEX client is used to push and/or pull objects from the OBEX server. An object can, for example, be a business card or an appointment. The OBEX client can obtain RFCOMM channel number from the remote device via SDP. This can be done by specifying service name instead of RFCOMM channel number. Supported service names are: IrMC, FTRN and OPUSH. It is possible to specify RFCOMM channel as a number. Below is an example of an OBEX session, where device information object is pulled from the cellular phone, and a new object (business card) is pushed into the phone's directory.

```
% 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)
```

In order to provide OBEX Object Push service, sdpd(8) server must be running. A root folder, where all incoming objects will be stored, must be created. The default path to the root folder is `/var/spool/obex`. Finally, start OBEX server on valid RFCOMM channel number. The OBEX server will automatically register OBEX Object Push service with the local SDP daemon. The example below shows how to start OBEX server.

```
# obexapp -s -C 10
```

31.4.10 Serial Port Profile (SPP)

The Serial Port Profile (SPP) allows Bluetooth devices to perform RS232 (or similar) serial cable emulation. The scenario covered by this profile deals with legacy applications using Bluetooth as a cable replacement, through a virtual serial port abstraction.

The rfcomm_sppd(1) utility implements the Serial Port profile. A pseudo tty is used as a virtual serial port abstraction. The example below shows how to connect to a remote device Serial Port service. Note that you do not have to specify a RFCOMM channel - rfcomm_sppd(1) can obtain it from the remote device via SDP. If you would like to override this, specify a RFCOMM channel on the command line.

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t /dev/ttyp6
rfcomm_sppd[94692]: Starting on /dev/ttyp6...
```

Once connected, the pseudo tty can be used as serial port:

```
# cu -l ttyp6
```

31.4.11 Troubleshooting

31.4.11.1 A remote device cannot connect

Some older Bluetooth devices do not support role switching. By default, when FreeBSD is accepting a new connection, it tries to perform a role switch and become master. Devices, which do not support this will not be able to connect. Note that role switching is performed when a new connection is being established, so it is not possible to ask the remote device if it does support role switching. There is a HCI option to disable role switching on the local side:

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

31.4.11.2 Something is going wrong, can I see what exactly is happening?

Yes, you can. Use the third-party package **hcidump**, which is available as `comms/hcidump` port. The **hcidump** utility is similar to tcpdump(1). It can be used to display the content of the Bluetooth packets on the terminal and to dump the Bluetooth packets to a file.

31.5 Bridging

31.5.1 Introduction

It is sometimes useful to divide one physical network (such as an Ethernet segment) into two separate network segments without having to create IP subnets and use a router to connect the segments together. A device that connects two networks together in this fashion is called a “bridge”. A FreeBSD system with two network interface cards can act as a bridge.

The bridge works by learning the MAC layer addresses (Ethernet addresses) of the devices on each of its network interfaces. It forwards traffic between two networks only when its source and destination are on different networks.

In many respects, a bridge is like an Ethernet switch with very few ports.

31.5.2 Situations Where Bridging Is Appropriate

There are many common situations in which a bridge is used today.

31.5.2.1 Connecting Networks

The basic operation of a bridge is to join two or more network segments together. There are many reasons to use a host based bridge over plain networking equipment such as cabling constraints, firewalling or connecting pseudo networks such as a Virtual Machine interface. A bridge can also connect a wireless interface running in hostap mode to a wired network and act as an access point.

31.5.2.2 Filtering/Traffic Shaping Firewall

A common situation is where firewall functionality is needed without routing or network address translation (NAT).

An example is a small company that is connected via DSL or ISDN to their ISP. They have a 13 globally-accessible IP addresses from their ISP and have 10 PCs on their network. In this situation, using a router-based firewall is difficult because of subnetting issues.

A bridge-based firewall can be configured and dropped into the path just downstream of their DSL/ISDN router without any IP numbering issues.

31.5.2.3 Network Tap

A bridge can join two network segments and be used to inspect all Ethernet frames that pass between them. This can either be from using bpf(4)/tcpdump(1) on the bridge interface or by sending a copy of all frames out an additional interface (span port).

31.5.2.4 Layer 2 VPN

Two Ethernet networks can be joined across an IP link by bridging the networks to an EtherIP tunnel or a tap(4) based solution such as OpenVPN.

31.5.2.5 Layer 2 Redundancy

A network can be connected together with multiple links and use the Spanning Tree Protocol to block redundant paths. For an Ethernet network to function properly only one active path can exist between two devices, Spanning Tree will detect loops and put the redundant links into a blocked state. Should one of the active links fail then the protocol will calculate a different tree and reenable one of the blocked paths to restore connectivity to all points in the network.

31.5.3 Kernel Configuration

This section covers if_bridge(4) bridge implementation, a netgraph bridging driver is also available, for more information see ng_bridge(4) manual page.

The bridge driver is a kernel module and will be automatically loaded by ifconfig(8) when creating a bridge interface. It is possible to compile the bridge in to the kernel by adding `device if_bridge` to your kernel configuration file.

Packet filtering can be used with any firewall package that hooks in via the pfil(9) framework. The firewall can be loaded as a module or compiled into the kernel.

The bridge can be used as a traffic shaper with altq(4) or dummynet(4).

31.5.4 Enabling the Bridge

The bridge is created using interface cloning. To create a bridge use ifconfig(8), if the bridge driver is not present in the kernel then it will be loaded automatically.

```
# 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
```

A bridge interface is created and is automatically assigned a randomly generated Ethernet address. The `maxaddr` and `timeout` parameters control how many MAC addresses the bridge will keep in its forwarding table and how many seconds before each entry is removed after it is last seen. The other parameters control how Spanning Tree operates.

Add the member network interfaces to the bridge. For the bridge to forward packets all member interfaces and the bridge need to be up:

```
# ifconfig bridge0 addm fxp0 addm fxp1 up
# ifconfig fxp0 up
# ifconfig fxp1 up
```

The bridge is now forwarding Ethernet frames between `fxp0` and `fxp1`. The equivalent configuration in `/etc/rc.conf` so the bridge is created at startup is:

```
cloned_interfaces="bridge0"
ifconfig_bridge0="addm fxp0 addm fxp1 up"
```

```
ifconfig_fxp0="up"
ifconfig_fxp1="up"
```

If the bridge host needs an IP address then the correct place to set this is on the bridge interface itself rather than one of the member interfaces. This can be set statically or via DHCP:

```
# ifconfig bridge0 inet 192.168.0.1/24
```

It is also possible to assign an IPv6 address to a bridge interface.

31.5.5 Firewalling

When packet filtering is enabled, bridged packets will pass through the filter inbound on the originating interface, on the bridge interface and outbound on the appropriate interfaces. Either stage can be disabled. When direction of the packet flow is important it is best to firewall on the member interfaces rather than the bridge itself.

The bridge has several configurable settings for passing non-IP and ARP packets, and layer2 firewalling with IPFW. See if_bridge(4) for more information.

31.5.6 Spanning Tree

The bridge driver implements the Rapid Spanning Tree Protocol (RSTP or 802.1w) with backwards compatibility with the legacy Spanning Tree Protocol (STP). Spanning Tree is used to detect and remove loops in a network topology. RSTP provides faster Spanning Tree convergence than legacy STP, the protocol will exchange information with neighbouring switches to quickly transition to forwarding without creating loops. FreeBSD supports RTSP and STP as operating modes, with RTSP being the default mode.

Spanning Tree can be enabled on member interfaces using the `stp` command. For a bridge with `fxp0` and `fxp1` as the current interfaces, enable STP with the following:

```
# 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
```

This bridge has a spanning tree ID of `00:01:02:4b:d4:50` and a priority of `32768`. As the `root id` is the same it indicates that this is the root bridge for the tree.

Another bridge on the network also has spanning tree enabled:

```
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
```

The line `root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4` shows that the root bridge is `00:01:02:4b:d4:50` as above and has a path cost of `400000` from this bridge, the path to the root bridge is via `port 4` which is `fxp0`.

31.5.7 Advanced Bridging

31.5.7.1 Reconstruct Traffic Flows

The bridge supports monitor mode, where the packets are discarded after bpf(4) processing, and are not processed or forwarded further. This can be used to multiplex the input of two or more interfaces into a single bpf(4) stream. This is useful for reconstructing the traffic for network taps that transmit the RX/TX signals out through two separate interfaces.

To read the input from four network interfaces as one stream:

```
# ifconfig bridge0 addm fxp0 addm fxp1 addm fxp2 addm fxp3 monitor up
# tcpdump -i bridge0
```

31.5.7.2 Span Ports

A copy of every Ethernet frame received by the bridge will be transmitted out a designated span port. The number of span ports configured on a bridge is unlimited, if an interface is designated as a span port then it may not also be used as a regular bridge port. This is most useful for snooping a bridged network passively on another host connected to one of the span ports of the bridge.

To send a copy of all frames out the interface named `fxp4`:

```
# ifconfig bridge0 span fxp4
```

31.5.7.3 Private Interfaces

A private interface does not forward any traffic to any other port that is also a private interface. The traffic is blocked unconditionally so no Ethernet frames will be forwarded, including ARP. If traffic needs to be selectively blocked then a firewall should be used instead.

31.5.7.4 Sticky Interfaces

If a bridge member interface is marked as sticky then dynamically learned address entries are treated at static once entered into the forwarding cache. Sticky entries are never aged out of the cache or replaced, even if the address is

seen on a different interface. This gives the benefit of static address entries without the need to pre-populate the forwarding table, clients learnt on a particular segment of the bridge can not roam to another segment.

Another example of using sticky addresses would be to combine the bridge with VLANs to create a router where customer networks are isolated without wasting IP address space. Consider that `CustomerA` is on `vlan100` and `CustomerB` is on `vlan101`. The bridge has the address `192.168.0.1` and is also an internet router.

```
# ifconfig bridge0 addm vlan100 sticky vlan100 addm vlan101 sticky vlan101
# ifconfig bridge0 inet 192.168.0.1/24
```

Both clients see `192.168.0.1` as their default gateway and since the bridge cache is sticky they can not spoof the MAC address of the other customer to intercept their traffic.

Any communication between the VLANs can be blocked using private interfaces (or a firewall):

```
# ifconfig bridge0 private vlan100 private vlan101
```

The customers are completely isolated from each other, the full `/24` address range can be allocated without subnetting.

31.5.7.5 Address limits

The number of unique source MAC addresses behind an interface can be limited. Once the limit is reached packets with unknown source addresses are dropped until an existing host cache entry expires or is removed.

The following example sets the maximum number of Ethernet devices for `CustomerA` on `vlan100` to 10.

```
# ifconfig bridge0 ifmaxaddr vlan100 10
```

31.5.7.6 SNMP Monitoring

The bridge interface and STP parameters can be monitored via the SNMP daemon which is included in the FreeBSD base system. The exported bridge MIBs conform to the IETF standards so any SNMP client or monitoring package can be used to retrieve the data.

On the bridge machine uncomment the `begemotSnmpdModulePath."bridge" = "/usr/lib/snmp_bridge.so"` line from `/etc/snmp.config` and start the **bsnmpd** daemon. Other configuration such as community names and access lists may need to be modified. See bsnmpd(1) and snmp_bridge(3) for more information.

The following examples use the **Net-SNMP** software (`net-mgmt/net-snmp`) to query a bridge, the `net-mgmt/bsnmptools` port can also be used. From the SNMP client host add to `$HOME/.snmp/snmp.conf` the following lines to import the bridge MIB definitions in to **Net-SNMP**:

```
mibdirs +/usr/share/snmp/mibs
mibs +BRIDGE-MIB:RSTP-MIB:BEGEMOT-MIB:BEGEMOT-BRIDGE-MIB
```

To monitor a single bridge via the IETF BRIDGE-MIB (RFC4188) do

```
% 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)
```

The `dot1dStpTopChanges.0` value is two which means that the STP bridge topology has changed twice, a topology change means that one or more links in the network have changed or failed and a new tree has been calculated. The `dot1dStpTimeSinceTopologyChange.0` value will show when this happened.

To monitor multiple bridge interfaces one may use the private BEGEMOT-BRIDGE-MIB:

```
% 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
```

To change the bridge interface being monitored via the `mib-2.dot1dBridge` subtree do:

```
% snmpset -v 2c -c private bridge1.example.com
BEGEMOT-BRIDGE-MIB::begemotBridgeDefaultBridgeIf.0 s bridge2
```

31.6 Link Aggregation and Failover

31.6.1 Introduction

The lagg(4) interface allows aggregation of multiple network interfaces as one virtual interface for the purpose of providing fault-tolerance and high-speed links.

31.6.2 Operating Modes

Failover

Sends and receives traffic only through the master port. If the master port becomes unavailable, the next active port is used. The first interface added is the master port; any interfaces added after that are used as failover devices.

Cisco® Fast EtherChannel®

Cisco Fast EtherChannel (FEC), is a static setup and does not negotiate aggregation with the peer or exchange frames to monitor the link. If the switch supports LACP then that should be used instead.

FEC balances outgoing traffic across the active ports based on hashed protocol header information and accepts incoming traffic from any active port. The hash includes the Ethernet source and destination address, and, if available, the VLAN tag, and the IPv4/IPv6 source and destination address.

LACP

The IEEE 802.3ad Link Aggregation Control Protocol (LACP) and the Marker Protocol. LACP will negotiate a set of aggregable links with the peer in to one or more Link Aggregated Groups (LAG). Each LAG is composed of ports of the same speed, set to full-duplex operation. The traffic will be balanced across the ports in the LAG with the greatest total speed, in most cases there will only be one LAG which contains all ports. In the event of changes in physical connectivity, Link Aggregation will quickly converge to a new configuration.

LACP balances outgoing traffic across the active ports based on hashed protocol header information and accepts incoming traffic from any active port. The hash includes the Ethernet source and destination address, and, if available, the VLAN tag, and the IPv4/IPv6 source and destination address.

Loadbalance

This is an alias of *FEC* mode.

Round-robin

Distributes outgoing traffic using a round-robin scheduler through all active ports and accepts incoming traffic from any active port. This mode violates Ethernet Frame ordering and should be used with caution.

31.6.3 Examples

Example 31-1. LACP aggregation with a Cisco® Switch

This example connects two interfaces on a FreeBSD machine to the switch as a single load balanced and fault tolerant link. More interfaces can be added to increase throughput and fault tolerance. Since frame ordering is mandatory on Ethernet links then any traffic between two stations always flows over the same physical link limiting the maximum speed to that of one interface. The transmit algorithm attempts to use as much information as it can to distinguish different traffic flows and balance across the available interfaces.

On the Cisco switch add the *`FastEthernet0/1`* and *`FastEthernet0/2`* interfaces to the channel-group *`1`*:

```
interface FastEthernet0/1
 channel-group 1 mode active
 channel-protocol lacp
```

```
!
interface FastEthernet0/2
 channel-group 1 mode active
 channel-protocol lacp
```

On the FreeBSD machine create the lagg(4) interface using *fxp0* and *fxp1*:

```
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto lacp laggport fxp0 laggport fxp1
```

View the interface status by running:

```
# ifconfig lagg0
```

Ports marked as *ACTIVE* are part of the active aggregation group that has been negotiated with the remote switch and traffic will be transmitted and received. Use the verbose output of ifconfig(8) to view the LAG identifiers.

```
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 lacp
        laggport: fxp1 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
        laggport: fxp0 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
```

To see the port status on the switch, use **`show lacp neighbor`**:

```
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:

                  LACP port                        Oper    Port     Port
Port      Flags   Priority  Dev ID         Age     Key     Number   State
Fa0/1     SA      32768     0005.5d71.8db8  29s    0x146   0x3      0x3D
Fa0/2     SA      32768     0005.5d71.8db8  29s    0x146   0x4      0x3D
```

For more detail use the **`show lacp neighbor detail`** command.

Example 31-2. Failover mode

Failover mode can be used to switch over to a secondary interface if the link is lost on the master interface. Create and configure the *lagg0* interface, with *fxp0* as the master interface and *fxp1* as the secondary interface:

```
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport fxp0 laggport fxp1
```

The interface will look something like this, the major differences will be the MAC address and the device names:

```
# 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>
```

Traffic will be transmitted and received on *fxp0*. If the link is lost on *fxp0* then *fxp1* will become the active link. If the link is restored on the master interface then it will once again become the active link.

Example 31-3. Failover mode between wired and wireless interfaces

For laptop users, it is usually desirable to make wireless as a secondary interface, which is to be used when the wired connection is not available. With lagg(4), it is possible to use one IP address, prefer the wired connection for both performance and security reasons, while maintaining the ability to transfer data over the wireless connection.

In this setup, we will need to override the underlying wireless interface's MAC address to match the lagg(4)'s, which is inherited from the master interface being used, the wired interface.

In this setup, we will treat the wired interface, *bge0*, as the master, and the wireless interface, *wlan0*, as the failover interface. The *wlan0* was created from *iwn0* which we will set up with the wired connection's MAC address. The first step would be to obtain the MAC address from the wired interface:

```
# 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
```

You can replace the *bge0* to match your reality, and will get a different `ether` line which is the MAC address of your wired interface. Now, we change the underlying wireless interface, *iwn0*:

```
# ifconfig iwn0 ether 00:21:70:da:ae:37
```

Bring up the wireless interface but don't set up any IP address on it:

```
# ifconfig wlan0 create wlandev iwn0 ssid my_router up
```

Create the lagg(4) interface with *bge0* as master, and failover to *wlan0* if necessary:

```
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport bge0 laggport wlan0
```

The interface will look something like this, the major differences will be the MAC address and the device names:

```
# 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>
```

To avoid having to do this after every reboot, one can add something like the following lines to the `/etc/rc.conf` file:

```
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 Diskless Operation

A FreeBSD machine can boot over the network and operate without a local disk, using file systems mounted from an NFS server. No system modification is necessary, beyond standard configuration files. Such a system is relatively easy to set up because all the necessary elements are readily available:

- There are at least two possible methods to load the kernel over the network:
  - PXE: The Intel Preboot eXecution Environment system is a form of smart boot ROM built into some networking cards or motherboards. See pxeboot(8) for more details.
  - The **Etherboot** port (`net/etherboot`) produces ROM-able code to boot kernels over the network. The code can be either burnt into a boot PROM on a network card, or loaded from a local floppy (or hard) disk drive, or from a running MS-DOS system. Many network cards are supported.

- A sample script (`/usr/share/examples/diskless/clone_root`) eases the creation and maintenance of the workstation's root file system on the server. The script will probably require a little customization but it will get you started very quickly.
- Standard system startup files exist in `/etc` to detect and support a diskless system startup.
- Swapping, if needed, can be done either to an NFS file or to a local disk.

There are many ways to set up diskless workstations. Many elements are involved, and most can be customized to suit local taste. The following will describe variations on the setup of a complete system, emphasizing simplicity and compatibility with the standard FreeBSD startup scripts. The system described has the following characteristics:

- The diskless workstations use a shared read-only `/` file system, and a shared read-only `/usr`.

  The root file system is a copy of a standard FreeBSD root (typically the server's), with some configuration files overridden by ones specific to diskless operation or, possibly, to the workstation they belong to.

  The parts of the root which have to be writable are overlaid with md(4) file systems. Any changes will be lost when the system reboots.
- The kernel is transferred and loaded either with **Etherboot** or PXE as some situations may mandate the use of either method.

Caution: As described, this system is insecure. It should live in a protected area of a network, and be untrusted by other hosts.

All the information in this section has been tested using FreeBSD 5.2.1-RELEASE.

31.7.1 Background Information

Setting up diskless workstations is both relatively straightforward and prone to errors. These are sometimes difficult to diagnose for a number of reasons. For example:

- Compile time options may determine different behaviors at runtime.
- Error messages are often cryptic or totally absent.

In this context, having some knowledge of the background mechanisms involved is very useful to solve the problems that may arise.

Several operations need to be performed for a successful bootstrap:

- The machine needs to obtain initial parameters such as its IP address, executable filename, server name, root path. This is done using the DHCP or BOOTP protocols. DHCP is a compatible extension of BOOTP, and uses the same port numbers and basic packet format.

  It is possible to configure a system to use only BOOTP. The bootpd(8) server program is included in the base FreeBSD system.

  However, DHCP has a number of advantages over BOOTP (nicer configuration files, possibility of using PXE, plus many others not directly related to diskless operation), and we will describe mainly a DHCP configuration, with equivalent examples using bootpd(8) when possible. The sample configuration will use the **ISC DHCP** software package (release 3.0.1.r12 was installed on the test server).
- The machine needs to transfer one or several programs to local memory. Either TFTP or NFS are used. The choice between TFTP and NFS is a compile time option in several places. A common source of error is to specify filenames for the wrong protocol: TFTP typically transfers all files from a single directory on the server, and would expect filenames relative to this directory. NFS needs absolute file paths.
- The possible intermediate bootstrap programs and the kernel need to be initialized and executed. There are several important variations in this area:
  - PXE will load pxeboot(8), which is a modified version of the FreeBSD third stage loader. The loader(8) will obtain most parameters necessary to system startup, and leave them in the kernel environment before transferring control. It is possible to use a `GENERIC` kernel in this case.
  - **Etherboot**, will directly load the kernel, with less preparation. You will need to build a kernel with specific options.

  PXE and **Etherboot** work equally well; however, because kernels normally let the loader(8) do more work for them, PXE is the preferred method.

  If your BIOS and network cards support PXE, you should probably use it.
- Finally, the machine needs to access its file systems. NFS is used in all cases.

See also diskless(8) manual page.

31.7.2 Setup Instructions

31.7.2.1 Configuration Using ISC DHCP

The **ISC DHCP** server can answer both BOOTP and DHCP requests.

ISC DHCP 3.1 is not part of the base system. You will first need to install the `net/isc-dhcp31-server` port or the corresponding package.

Once **ISC DHCP** is installed, it needs a configuration file to run (normally named `/usr/local/etc/dhcpd.conf`). Here follows a commented example, where host `margaux` uses **Etherboot** and host `corbieres` uses PXE:

```
default-lease-time 600;
max-lease-time 7200;
authoritative;

option domain-name "example.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.example.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.example.com;
    next-server 192.168.4.4;
    filename "pxeboot";
    option root-path "192.168.4.4:/data/misc/diskless";
  }
}
```

❶ This option tells **dhcpd** to send the value in the `host` declarations as the hostname for the diskless host. An alternate way would be to add an `option host-name` *`margaux`* inside the `host` declarations.

❷ The `next-server` directive designates the TFTP or NFS server to use for loading loader or kernel file (the default is to use the same host as the DHCP server).

❸ The `filename` directive defines the file that **Etherboot** or PXE will load for the next execution step. It must be specified according to the transfer method used. **Etherboot** can be compiled to use NFS or TFTP. The FreeBSD port configures NFS by default. PXE uses TFTP, which is why a relative filename is used here (this may depend on the TFTP server configuration, but would be fairly typical). Also, PXE loads `pxeboot`, not the kernel. There

are other interesting possibilities, like loading `pxeboot` from a FreeBSD CD-ROM `/boot` directory (as pxeboot(8) can load a `GENERIC` kernel, this makes it possible to use PXE to boot from a remote CD-ROM).

❹ The `root-path` option defines the path to the root file system, in usual NFS notation. When using PXE, it is possible to leave off the host's IP as long as you do not enable the kernel option BOOTP. The NFS server will then be the same as the TFTP one.

31.7.2.2 Configuration Using BOOTP

Here follows an equivalent **bootpd** configuration (reduced to one client). This would be found in `/etc/bootptab`.

Please note that **Etherboot** must be compiled with the non-default option `NO_DHCP_SUPPORT` in order to use BOOTP, and that PXE *needs* DHCP. The only obvious advantage of **bootpd** is that it exists in the base system.

```
.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 Preparing a Boot Program with Etherboot

Etherboot's Web site (http://etherboot.sourceforge.net) contains extensive documentation (http://etherboot.sourceforge.net/doc/html/userman/t1.html) mainly intended for Linux systems, but nonetheless containing useful information. The following will just outline how you would use **Etherboot** on a FreeBSD system.

You must first install the `net/etherboot` package or port.

You can change the **Etherboot** configuration (i.e. to use TFTP instead of NFS) by editing the `Config` file in the **Etherboot** source directory.

For our setup, we shall use a boot floppy. For other methods (PROM, or MS-DOS program), please refer to the **Etherboot** documentation.

To make a boot floppy, insert a floppy in the drive on the machine where you installed **Etherboot**, then change your current directory to the `src` directory in the **Etherboot** tree and type:

```
# gmake bin32/devicetype.fd0
```

`devicetype` depends on the type of the Ethernet card in the diskless workstation. Refer to the `NIC` file in the same directory to determine the right *`devicetype`*.

31.7.2.4 Booting with PXE

By default, the pxeboot(8) loader loads the kernel via NFS. It can be compiled to use TFTP instead by specifying the `LOADER_TFTP_SUPPORT` option in `/etc/make.conf`. See the comments in `/usr/share/examples/etc/make.conf` for instructions.

There are two other `make.conf` options which may be useful for setting up a serial console diskless machine: `BOOT_PXELDR_PROBE_KEYBOARD`, and `BOOT_PXELDR_ALWAYS_SERIAL`.

To use PXE when the machine starts, you will usually need to select the `Boot from network` option in your BIOS setup, or type a function key during the PC initialization.

31.7.2.5 Configuring the TFTP and NFS Servers

If you are using PXE or **Etherboot** configured to use TFTP, you need to enable **tftpd** on the file server:

1. Create a directory from which **tftpd** will serve the files, e.g. `/tftpboot`.
2. Add this line to your `/etc/inetd.conf`:

   ```
   tftp dgram udp wait root /usr/libexec/tftpd tftpd -l -s /tftpboot
   ```

   Note: It appears that at least some PXE versions want the TCP version of TFTP. In this case, add a second line, replacing `dgram udp` with `stream tcp`.

3. Tell **inetd** to reread its configuration file. The `inetd_enable="YES"` must be in the `/etc/rc.conf` file for this command to execute correctly:

   ```
   # /etc/rc.d/inetd restart
   ```

You can place the `tftpboot` directory anywhere on the server. Make sure that the location is set in both `inetd.conf` and `dhcpd.conf`.

In all cases, you also need to enable NFS and export the appropriate file system on the NFS server.

1. Add this to `/etc/rc.conf`:

   ```
   nfs_server_enable="YES"
   ```

2. Export the file system where the diskless root directory is located by adding the following to `/etc/exports` (adjust the volume mount point and replace *`margaux corbieres`* with the names of the diskless workstations):

   ```
   /data/misc -alldirs -ro margaux corbieres
   ```

3. Tell **mountd** to reread its configuration file. If you actually needed to enable NFS in `/etc/rc.conf` at the first step, you probably want to reboot instead.

   ```
   # /etc/rc.d/mountd restart
   ```

31.7.2.6 Building a Diskless Kernel

If using **Etherboot**, you need to create a kernel configuration file for the diskless client with the following options (in addition to the usual ones):

```
options     BOOTP          # Use BOOTP to obtain IP address/hostname
options     BOOTP_NFSROOT  # NFS mount root file system using BOOTP info
```

You may also want to use BOOTP_NFSV3, BOOT_COMPAT and BOOTP_WIRED_TO (refer to NOTES).

These option names are historical and slightly misleading as they actually enable indifferent use of DHCP and BOOTP inside the kernel (it is also possible to force strict BOOTP or DHCP use).

Build the kernel (see Chapter 8), and copy it to the place specified in dhcpd.conf.

> **Note:** When using PXE, building a kernel with the above options is not strictly necessary (though suggested). Enabling them will cause more DHCP requests to be issued during kernel startup, with a small risk of inconsistency between the new values and those retrieved by pxeboot(8) in some special cases. The advantage of using them is that the host name will be set as a side effect. Otherwise you will need to set the host name by another method, for example in a client-specific rc.conf file.

> **Note:** In order to be loadable with **Etherboot**, a kernel needs to have the device hints compiled in. You would typically set the following option in the configuration file (see the NOTES configuration comments file):
>
> ```
> hints  "GENERIC.hints"
> ```

31.7.2.7 Preparing the Root Filesystem

You need to create a root file system for the diskless workstations, in the location listed as root-path in dhcpd.conf.

31.7.2.7.1 Using `make world` to populate root

This method is quick and will install a complete virgin system (not only the root file system) into DESTDIR. All you have to do is simply execute the following script:

```
#!/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
```

Once done, you may need to customize your /etc/rc.conf and /etc/fstab placed into DESTDIR according to your needs.

31.7.2.8 Configuring Swap

If needed, a swap file located on the server can be accessed via NFS.

31.7.2.8.1 NFS Swap

The kernel does not support enabling NFS swap at boot time. Swap must be enabled by the startup scripts, by mounting a writable file system and creating and enabling a swap file. To create a swap file of appropriate size, you can do like this:

```
# dd if=/dev/zero of=/path/to/swapfile bs=1k count=1 oseek=100000
```

To enable it you have to add the following line to your `rc.conf`:

```
swapfile=/path/to/swapfile
```

31.7.2.9 Miscellaneous Issues

31.7.2.9.1 Running with a Read-only `/usr`

If the diskless workstation is configured to run X, you will have to adjust the **XDM** configuration file, which puts the error log on `/usr` by default.

31.7.2.9.2 Using a Non-FreeBSD Server

When the server for the root file system is not running FreeBSD, you will have to create the root file system on a FreeBSD machine, then copy it to its destination, using `tar` or `cpio`.

In this situation, there are sometimes problems with the special files in `/dev`, due to differing major/minor integer sizes. A solution to this problem is to export a directory from the non-FreeBSD server, mount this directory onto a FreeBSD machine, and use devfs(5) to allocate device nodes transparently for the user.

31.8 ISDN

A good resource for information on ISDN technology and hardware is Dan Kegel's ISDN Page (http://www.alumni.caltech.edu/~dank/isdn/).

A quick simple road map to ISDN follows:

- If you live in Europe you might want to investigate the ISDN card section.
- If you are planning to use ISDN primarily to connect to the Internet with an Internet Provider on a dial-up non-dedicated basis, you might look into Terminal Adapters. This will give you the most flexibility, with the fewest problems, if you change providers.
- If you are connecting two LANs together, or connecting to the Internet with a dedicated ISDN connection, you might consider the stand alone router/bridge option.

Cost is a significant factor in determining what solution you will choose. The following options are listed from least expensive to most expensive.

31.8.1 ISDN Cards

FreeBSD's ISDN implementation supports only the DSS1/Q.931 (or Euro-ISDN) standard using passive cards. Some active cards are supported where the firmware also supports other signaling protocols; this also includes the first supported Primary Rate (PRI) ISDN card.

The **isdn4bsd** software allows you to connect to other ISDN routers using either IP over raw HDLC or by using synchronous PPP: either by using kernel PPP with `isppp`, a modified sppp(4) driver, or by using userland ppp(8). By using userland ppp(8), channel bonding of two or more ISDN B-channels is possible. A telephone answering machine application is also available as well as many utilities such as a software 300 Baud modem.

Some growing number of PC ISDN cards are supported under FreeBSD and the reports show that it is successfully used all over Europe and in many other parts of the world.

The passive ISDN cards supported are mostly the ones with the Infineon (formerly Siemens) ISAC/HSCX/IPAC ISDN chipsets, but also ISDN cards with chips from Cologne Chip (ISA bus only), PCI cards with Winbond W6692 chips, some cards with the Tiger300/320/ISAC chipset combinations and some vendor specific chipset based cards such as the AVM Fritz!Card PCI V.1.0 and the AVM Fritz!Card PnP.

Currently the active supported ISDN cards are the AVM B1 (ISA and PCI) BRI cards and the AVM T1 PCI PRI cards.

For documentation on **isdn4bsd**, have a look at the homepage of isdn4bsd (http://www.freebsd-support.de/i4b/) which also has pointers to hints, erratas and much more documentation such as the isdn4bsd handbook (http://people.FreeBSD.org/~hm/).

In case you are interested in adding support for a different ISDN protocol, a currently unsupported ISDN PC card or otherwise enhancing **isdn4bsd**, please get in touch with Hellmuth Michaelis <`hm@FreeBSD.org`>.

For questions regarding the installation, configuration and troubleshooting **isdn4bsd**, a freebsd-isdn (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn) mailing list is available.

31.8.2 ISDN Terminal Adapters

Terminal adapters (TA), are to ISDN what modems are to regular phone lines.

Most TA's use the standard Hayes modem AT command set, and can be used as a drop in replacement for a modem.

A TA will operate basically the same as a modem except connection and throughput speeds will be much faster than your old modem. You will need to configure PPP exactly the same as for a modem setup. Make sure you set your serial speed as high as possible.

The main advantage of using a TA to connect to an Internet Provider is that you can do Dynamic PPP. As IP address space becomes more and more scarce, most providers are not willing to provide you with a static IP anymore. Most stand-alone routers are not able to accommodate dynamic IP allocation.

TA's completely rely on the PPP daemon that you are running for their features and stability of connection. This allows you to upgrade easily from using a modem to ISDN on a FreeBSD machine, if you already have PPP set up. However, at the same time any problems you experienced with the PPP program and are going to persist.

If you want maximum stability, use the kernel PPP option, not the userland PPP.

The following TA's are known to work with FreeBSD:

- Motorola BitSurfer and Bitsurfer Pro

- Adtran

Most other TA's will probably work as well, TA vendors try to make sure their product can accept most of the standard modem AT command set.

The real problem with external TA's is that, like modems, you need a good serial card in your computer.

You should read the FreeBSD Serial Hardware (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/serial-uart/index.html) tutorial for a detailed understanding of serial devices, and the differences between asynchronous and synchronous serial ports.

A TA running off a standard PC serial port (asynchronous) limits you to 115.2 Kbs, even though you have a 128 Kbs connection. To fully utilize the 128 Kbs that ISDN is capable of, you must move the TA to a synchronous serial card.

Do not be fooled into buying an internal TA and thinking you have avoided the synchronous/asynchronous issue. Internal TA's simply have a standard PC serial port chip built into them. All this will do is save you having to buy another serial cable and find another empty electrical socket.

A synchronous card with a TA is at least as fast as a stand-alone router, and with a simple 386 FreeBSD box driving it, probably more flexible.

The choice of synchronous card/TA v.s. stand-alone router is largely a religious issue. There has been some discussion of this in the mailing lists. We suggest you search the archives (http://www.FreeBSD.org/search/index.html) for the complete discussion.

31.8.3 Stand-alone ISDN Bridges/Routers

ISDN bridges or routers are not at all specific to FreeBSD or any other operating system. For a more complete description of routing and bridging technology, please refer to a networking reference book.

In the context of this section, the terms router and bridge will be used interchangeably.

As the cost of low end ISDN routers/bridges comes down, it will likely become a more and more popular choice. An ISDN router is a small box that plugs directly into your local Ethernet network, and manages its own connection to the other bridge/router. It has built in software to communicate via PPP and other popular protocols.

A router will allow you much faster throughput than a standard TA, since it will be using a full synchronous ISDN connection.

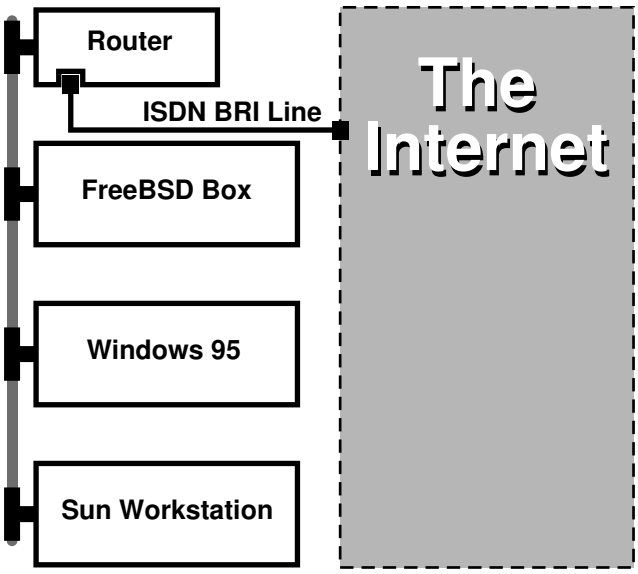
The main problem with ISDN routers and bridges is that interoperability between manufacturers can still be a problem. If you are planning to connect to an Internet provider, you should discuss your needs with them.

If you are planning to connect two LAN segments together, such as your home LAN to the office LAN, this is the simplest lowest maintenance solution. Since you are buying the equipment for both sides of the connection you can be assured that the link will work.

For example to connect a home computer or branch office network to a head office network the following setup could be used:

Example 31-4. Branch Office or Home Network

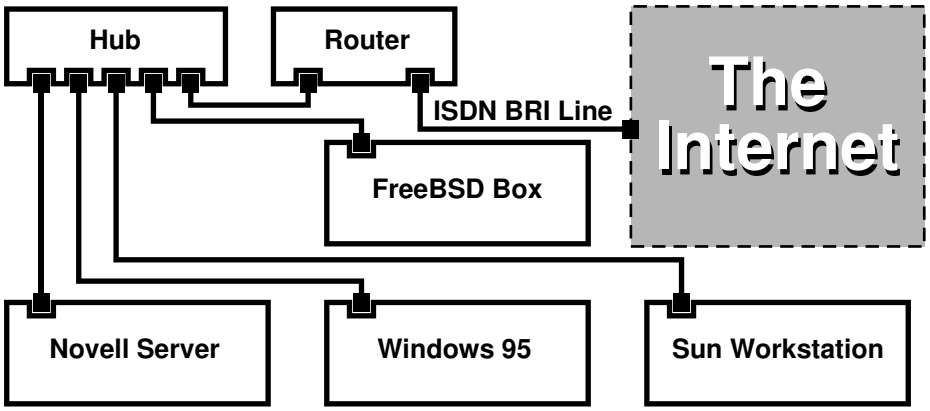
Network uses a bus based topology with 10 base 2 Ethernet ("thinnet"). Connect router to network cable with AUI/10BT transceiver, if necessary.



If your home/branch office is only one computer you can use a twisted pair crossover cable to connect to the stand-alone router directly.

Example 31-5. Head Office or Other LAN

Network uses a star topology with 10 base T Ethernet (“Twisted Pair”).



One large advantage of most routers/bridges is that they allow you to have 2 *separate independent* PPP connections to 2 separate sites at the *same* time. This is not supported on most TA’s, except for specific (usually expensive) models that have two serial ports. Do not confuse this with channel bonding, MPP, etc.

This can be a very useful feature if, for example, you have an dedicated ISDN connection at your office and would like to tap into it, but do not want to get another ISDN line at work. A router at the office location can manage a dedicated B channel connection (64 Kbps) to the Internet and use the other B channel for a separate data connection.

The second B channel can be used for dial-in, dial-out or dynamically bonding (MPP, etc.) with the first B channel for more bandwidth.

An Ethernet bridge will also allow you to transmit more than just IP traffic. You can also send IPX/SPX or whatever other protocols you use.

31.9 Network Address Translation

31.9.1 Overview

FreeBSD's Network Address Translation daemon, commonly known as natd(8) is a daemon that accepts incoming raw IP packets, changes the source to the local machine and re-injects these packets back into the outgoing IP packet stream. natd(8) does this by changing the source IP address and port such that when data is received back, it is able to determine the original location of the data and forward it back to its original requester.

The most common use of NAT is to perform what is commonly known as Internet Connection Sharing.

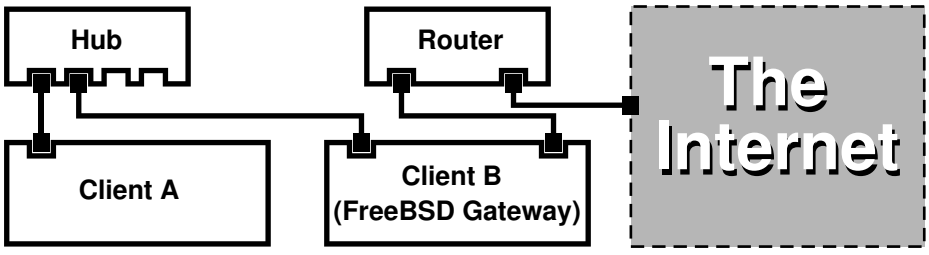
31.9.2 Setup

Due to the diminishing IP space in IPv4, and the increased number of users on high-speed consumer lines such as cable or DSL, people are increasingly in need of an Internet Connection Sharing solution. The ability to connect several computers online through one connection and IP address makes natd(8) a reasonable choice.

Most commonly, a user has a machine connected to a cable or DSL line with one IP address and wishes to use this one connected computer to provide Internet access to several more over a LAN.

To do this, the FreeBSD machine on the Internet must act as a gateway. This gateway machine must have two NICs—one for connecting to the Internet router, the other connecting to a LAN. All the machines on the LAN are connected through a hub or switch.

> **Note:** There are many ways to get a LAN connected to the Internet through a FreeBSD gateway. This example will only cover a gateway with at least two NICs.



A setup like this is commonly used to share an Internet connection. One of the LAN machines is connected to the Internet. The rest of the machines access the Internet through that "gateway" machine.

31.9.3 Boot Loader Configuration

The kernel features for network address translation with natd(8) are not enabled in the `GENERIC` kernel, but they can be preloaded at boot time, by adding a couple of options to `/boot/loader.conf`:

```
ipfw_load="YES"
ipdivert_load="YES"
```

Additionally, the `net.inet.ip.fw.default_to_accept` tunable option may be set to `1`:

```
net.inet.ip.fw.default_to_accept="1"
```

> **Note:** It is a very good idea to set this option during the first attempts to setup a firewall and NAT gateway. This way the default policy of ipfw(8) will be `allow ip from any to any` instead of the less permissive `deny ip from any to any`, and it will be slightly more difficult to get locked out of the system right after a reboot.

31.9.4 Kernel Configuration

When modules are not an option or if it is preferrable to build all the required features into the running kernel, the following options must be in the kernel configuration file:

```
options IPFIREWALL
options IPDIVERT
```

Additionally, at choice, the following may also be suitable:

```
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_VERBOSE
```

31.9.5 System Startup Configuration

To enable firewall and NAT support at boot time, the following must be in `/etc/rc.conf`:

```
gateway_enable="YES" ❶
firewall_enable="YES" ❷
firewall_type="OPEN" ❸
natd_enable="YES"
natd_interface="fxp0" ❹
natd_flags="" ❺
```

❶ Sets up the machine to act as a gateway. Running `sysctl net.inet.ip.forwarding=1` would have the same effect.

❷ Enables the firewall rules in `/etc/rc.firewall` at boot.

❸ This specifies a predefined firewall ruleset that allows anything in. See `/etc/rc.firewall` for additional types.

❹ Indicates which interface to forward packets through (the interface connected to the Internet).

❺ Any additional configuration options passed to natd(8) on boot.

Having the previous options defined in `/etc/rc.conf` would run `natd -interface fxp0` at boot. This can also be run manually.

> **Note:** It is also possible to use a configuration file for natd(8) when there are too many options to pass. In this case, the configuration file must be defined by adding the following line to `/etc/rc.conf`:
>
> ```
> natd_flags="-f /etc/natd.conf"
> ```
>
> The `/etc/natd.conf` file will contain a list of configuration options, one per line. For example the next section case would use the following file:
>
> ```
> redirect_port tcp 192.168.0.2:6667 6667
> redirect_port tcp 192.168.0.3:80 80
> ```
>
> For more information about the configuration file, consult the natd(8) manual page about the `-f` option.

Each machine and interface behind the LAN should be assigned IP address numbers in the private network space as defined by RFC 1918 (ftp://ftp.isi.edu/in-notes/rfc1918.txt) and have a default gateway of the **natd** machine's internal IP address.

For example, client `A` and `B` behind the LAN have IP addresses of `192.168.0.2` and `192.168.0.3`, while the natd machine's LAN interface has an IP address of `192.168.0.1`. Client `A` and `B`'s default gateway must be set to that of the **natd** machine, `192.168.0.1`. The **natd** machine's external, or Internet interface does not require any special modification for natd(8) to work.

31.9.6 Port Redirection

The drawback with natd(8) is that the LAN clients are not accessible from the Internet. Clients on the LAN can make outgoing connections to the world but cannot receive incoming ones. This presents a problem if trying to run Internet services on one of the LAN client machines. A simple way around this is to redirect selected Internet ports on the **natd** machine to a LAN client.

For example, an IRC server runs on client `A`, and a web server runs on client `B`. For this to work properly, connections received on ports 6667 (IRC) and 80 (web) must be redirected to the respective machines.

The `-redirect_port` must be passed to natd(8) with the proper options. The syntax is as follows:

```
 -redirect_port proto targetIP:targetPORT[-targetPORT]
            [aliasIP:]aliasPORT[-aliasPORT]
            [remoteIP[:remotePORT[-remotePORT]]]
```

In the above example, the argument should be:

```
-redirect_port tcp 192.168.0.2:6667 6667
-redirect_port tcp 192.168.0.3:80 80
```

This will redirect the proper *tcp* ports to the LAN client machines.

The `-redirect_port` argument can be used to indicate port ranges over individual ports. For example, *`tcp 192.168.0.2:2000-3000 2000-3000`* would redirect all connections received on ports 2000 to 3000 to ports 2000 to 3000 on client `A`.

These options can be used when directly running natd(8), placed within the `natd_flags=""` option in `/etc/rc.conf`, or passed via a configuration file.

For further configuration options, consult natd(8)

31.9.7 Address Redirection

Address redirection is useful if several IP addresses are available, yet they must be on one machine. With this, natd(8) can assign each LAN client its own external IP address. natd(8) then rewrites outgoing packets from the LAN clients with the proper external IP address and redirects all traffic incoming on that particular IP address back to the specific LAN client. This is also known as static NAT. For example, the IP addresses `128.1.1.1`, `128.1.1.2`, and `128.1.1.3` belong to the **natd** gateway machine. `128.1.1.1` can be used as the **natd** gateway machine's external IP address, while `128.1.1.2` and `128.1.1.3` are forwarded back to LAN clients `A` and `B`.

The `-redirect_address` syntax is as follows:

```
-redirect_address localIP publicIP
```

localIP	The internal IP address of the LAN client.
publicIP	The external IP address corresponding to the LAN client.

In the example, this argument would read:

```
-redirect_address 192.168.0.2 128.1.1.2
-redirect_address 192.168.0.3 128.1.1.3
```

Like `-redirect_port`, these arguments are also placed within the `natd_flags=""` option of `/etc/rc.conf`, or passed via a configuration file. With address redirection, there is no need for port redirection since all data received on a particular IP address is redirected.

The external IP addresses on the **natd** machine must be active and aliased to the external interface. Look at rc.conf(5) to do so.

31.10 Parallel Line IP (PLIP)

PLIP lets us run TCP/IP between parallel ports. It is useful on machines without network cards, or to install on laptops. In this section, we will discuss:

- Creating a parallel (laplink) cable.
- Connecting two computers with PLIP.

31.10.1 Creating a Parallel Cable

You can purchase a parallel cable at most computer supply stores. If you cannot do that, or you just want to know how it is done, the following table shows how to make one out of a normal parallel printer cable.

Table 31-1. Wiring a Parallel Cable for Networking

A-name	A-End	B-End	Descr.	Post/Bit
DATA0 -ERROR	2 15	15 2	Data	0/0x01 1/0x08
DATA1 +SLCT	3 13	13 3	Data	0/0x02 1/0x10
DATA2 +PE	4 12	12 4	Data	0/0x04 1/0x20
DATA3 -ACK	5 10	10 5	Strobe	0/0x08 1/0x40
DATA4 BUSY	6 11	11 6	Data	0/0x10 1/0x80
GND	18-25	18-25	GND	-

31.10.2 Setting Up PLIP

First, you have to get a laplink cable. Then, confirm that both computers have a kernel with lpt(4) driver support:

```
# grep lp /var/run/dmesg.boot
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
```

The parallel port must be an interrupt driven port, you should have lines similar to the following in your in the `/boot/device.hints` file:

```
hint.ppc.0.at="isa"
hint.ppc.0.irq="7"
```

Then check if the kernel configuration file has a `device plip` line or if the `plip.ko` kernel module is loaded. In both cases the parallel networking interface should appear when you use the ifconfig(8) command to display it:

```
# ifconfig plip0
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
```

Plug the laplink cable into the parallel interface on both computers.

Configure the network interface parameters on both sites as `root`. For example, if you want to connect the host `host1` with another machine `host2`:

```
                host1 <-----> host2
IP Address    10.0.0.1      10.0.0.2
```

Configure the interface on `host1` by doing:

```
# ifconfig plip0 10.0.0.1 10.0.0.2
```

Configure the interface on `host2` by doing:

```
# ifconfig plip0 10.0.0.2 10.0.0.1
```

You now should have a working connection. Please read the manual pages lp(4) and lpt(4) for more details.

You should also add both hosts to `/etc/hosts`:

```
127.0.0.1               localhost.my.domain localhost
```

```
10.0.0.1                host1.my.domain host1
10.0.0.2                host2.my.domain host2
```

To confirm the connection works, go to each host and ping the other. For example, on `host1`:

```
# 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
host2              host1            UH          0       0       plip0
# ping -c 4 host2
PING host2 (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

--- host2 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 IPv6

IPv6 (also known as IPng “IP next generation”) is the new version of the well known IP protocol (also known as IPv4). Like the other current *BSD systems, FreeBSD includes the KAME IPv6 reference implementation. So your FreeBSD system comes with all you will need to experiment with IPv6. This section focuses on getting IPv6 configured and running.

In the early 1990s, people became aware of the rapidly diminishing address space of IPv4. Given the expansion rate of the Internet there were two major concerns:

- Running out of addresses. Today this is not so much of a concern anymore since RFC1918 private address space (`10.0.0.0/8`, `172.16.0.0/12`, and `192.168.0.0/16`) and Network Address Translation (NAT) are being employed.
- Router table entries were getting too large. This is still a concern today.

IPv6 deals with these and many other issues:

- 128 bit address space. In other words theoretically there are 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses available. This means there are approximately 6.67 * 10^27 IPv6 addresses per square meter on our planet.
- Routers will only store network aggregation addresses in their routing tables thus reducing the average space of a routing table to 8192 entries.

There are also lots of other useful features of IPv6 such as:

- Address autoconfiguration (RFC2462 (http://www.ietf.org/rfc/rfc2462.txt))
- Anycast addresses (“one-out-of many”)
- Mandatory multicast addresses
- IPsec (IP security)
- Simplified header structure
- Mobile IP
- IPv6-to-IPv4 transition mechanisms

For more information see:

- IPv6 overview at playground.sun.com (http://playground.sun.com/pub/ipng/html/ipng-main.html)
- KAME.net (http://www.kame.net)

31.11.1 Background on IPv6 Addresses

There are different types of IPv6 addresses: Unicast, Anycast and Multicast.

Unicast addresses are the well known addresses. A packet sent to a unicast address arrives exactly at the interface belonging to the address.

Anycast addresses are syntactically indistinguishable from unicast addresses but they address a group of interfaces. The packet destined for an anycast address will arrive at the nearest (in router metric) interface. Anycast addresses may only be used by routers.

Multicast addresses identify a group of interfaces. A packet destined for a multicast address will arrive at all interfaces belonging to the multicast group.

Note: The IPv4 broadcast address (usually `xxx.xxx.xxx.255`) is expressed by multicast addresses in IPv6.

Table 31-2. Reserved IPv6 addresses

IPv6 address	Prefixlength (Bits)	Description	Notes
`::`	128 bits	unspecified	cf. `0.0.0.0` in IPv4
`::1`	128 bits	loopback address	cf. `127.0.0.1` in IPv4
`::00:xx:xx:xx:xx`	96 bits	embedded IPv4	The lower 32 bits are the IPv4 address. Also called “IPv4 compatible IPv6 address”
`::ff:xx:xx:xx:xx`	96 bits	IPv4 mapped IPv6 address	The lower 32 bits are the IPv4 address. For hosts which do not support IPv6.
`fe80::-feb::`	10 bits	link-local	cf. loopback address in IPv4
`fec0::-fef::`	10 bits	site-local	

IPv6 address	Prefixlength (Bits)	Description	Notes
`ff::`	8 bits	multicast	
`001` (base 2)	3 bits	global unicast	All global unicast addresses are assigned from this pool. The first 3 bits are “001”.

31.11.2 Reading IPv6 Addresses

The canonical form is represented as: `x:x:x:x:x:x:x:x`, each “x” being a 16 Bit hex value. For example `FEBC:A574:382B:23C1:AA49:4592:4EFE:9982`

Often an address will have long substrings of all zeros therefore one such substring per address can be abbreviated by “::”. Also up to three leading “0”s per hexquad can be omitted. For example `fe80::1` corresponds to the canonical form `fe80:0000:0000:0000:0000:0000:0000:0001`.

A third form is to write the last 32 Bit part in the well known (decimal) IPv4 style with dots “.” as separators. For example `2002::10.0.0.1` corresponds to the (hexadecimal) canonical representation `2002:0000:0000:0000:0000:0000:0a00:0001` which in turn is equivalent to writing `2002::a00:1`.

By now the reader should be able to understand the following:

```
# ifconfig

rl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
         inet 10.0.0.10 netmask 0xffffff00 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
```

`fe80::200:21ff:fe03:8e1%rl0` is an auto configured link-local address. It is generated from the MAC address as part of the auto configuration.

For further information on the structure of IPv6 addresses see RFC3513 (http://www.ietf.org/rfc/rfc3513.txt).

31.11.3 Getting Connected

Currently there are four ways to connect to other IPv6 hosts and networks:

- Contact your Internet Service Provider to see if they offer IPv6 yet.
- SixXS (http://www.sixxs.net) offers tunnels with end-points all around the globe.
- Tunnel via 6-to-4 (RFC3068 (http://www.ietf.org/rfc/rfc3068.txt))
- Use the `net/freenet6` port if you are on a dial-up connection.

31.11.4 DNS in the IPv6 World

There used to be two types of DNS records for IPv6. The IETF has declared A6 records obsolete. AAAA records are the standard now.

Using AAAA records is straightforward. Assign your hostname to the new IPv6 address you just received by adding:

```
MYHOSTNAME            AAAA     MYIPv6ADDR
```

To your primary zone DNS file. In case you do not serve your own DNS zones ask your DNS provider. Current versions of **bind** (version 8.3 and 9) and `dns/djbdns` (with the IPv6 patch) support AAAA records.

31.11.5 Applying the needed changes to `/etc/rc.conf`

31.11.5.1 IPv6 Client Settings

These settings will help you configure a machine that will be on your LAN and act as a client, not a router. To have rtsol(8) autoconfigure your interface on boot all you need to add is:

```
ipv6_enable="YES"
```

To statically assign an IP address such as `2001:471:1f11:251:290:27ff:fee0:2093`, to your `fxp0` interface, add:

```
ipv6_ifconfig_fxp0="2001:471:1f11:251:290:27ff:fee0:2093"
```

To assign a default router of `2001:471:1f11:251::1` add the following to `/etc/rc.conf`:

```
ipv6_defaultrouter="2001:471:1f11:251::1"
```

31.11.5.2 IPv6 Router/Gateway Settings

This will help you take the directions that your tunnel provider has given you and convert it into settings that will persist through reboots. To restore your tunnel on startup use something like the following in `/etc/rc.conf`:

List the Generic Tunneling interfaces that will be configured, for example `gif0`:

```
gif_interfaces="gif0"
```

To configure the interface with a local endpoint of *`MY_IPv4_ADDR`* to a remote endpoint of *`REMOTE_IPv4_ADDR`*:

```
gifconfig_gif0="MY_IPv4_ADDR REMOTE_IPv4_ADDR"
```

To apply the IPv6 address you have been assigned for use as your IPv6 tunnel endpoint, add:

```
ipv6_ifconfig_gif0="MY_ASSIGNED_IPv6_TUNNEL_ENDPOINT_ADDR"
```

Then all you have to do is set the default route for IPv6. This is the other side of the IPv6 tunnel:

```
ipv6_defaultrouter="MY_IPv6_REMOTE_TUNNEL_ENDPOINT_ADDR"
```

31.11.5.3 IPv6 Tunnel Settings

If the server is to route IPv6 between the rest of your network and the world, the following `/etc/rc.conf` setting will also be needed:

```
ipv6_gateway_enable="YES"
```

31.11.6 Router Advertisement and Host Auto Configuration

This section will help you setup rtadvd(8) to advertise the IPv6 default route.

To enable rtadvd(8) you will need the following in your `/etc/rc.conf`:

```
rtadvd_enable="YES"
```

It is important that you specify the interface on which to do IPv6 router solicitation. For example to tell rtadvd(8) to use `fxp0`:

```
rtadvd_interfaces="fxp0"
```

Now we must create the configuration file, `/etc/rtadvd.conf`. Here is an example:

```
fxp0:\
 :addrs#1:addr="2001:471:1f11:246::":prefixlen#64:tc=ether:
```

Replace `fxp0` with the interface you are going to be using.

Next, replace `2001:471:1f11:246::` with the prefix of your allocation.

If you are dedicated a `/64` subnet you will not need to change anything else. Otherwise, you will need to change the `prefixlen#` to the correct value.

31.12 Asynchronous Transfer Mode (ATM)

31.12.1 Configuring classical IP over ATM (PVCs)

Classical IP over ATM (CLIP) is the simplest method to use Asynchronous Transfer Mode (ATM) with IP. It can be used with switched connections (SVCs) and with permanent connections (PVCs). This section describes how to set up a network based on PVCs.

31.12.1.1 Fully meshed configurations

The first method to set up a CLIP with PVCs is to connect each machine to each other machine in the network via a dedicated PVC. While this is simple to configure it tends to become impractical for a larger number of machines. The example supposes that we have four machines in the network, each connected to the ATM network with an ATM adapter card. The first step is the planning of the IP addresses and the ATM connections between the machines. We use the following:

Host	IP Address
hostA	192.168.173.1
hostB	192.168.173.2
hostC	192.168.173.3
hostD	192.168.173.4

To build a fully meshed net we need one ATM connection between each pair of machines:

Machines	VPI.VCI couple
hostA - hostB	0.100
hostA - hostC	0.101
hostA - hostD	0.102
hostB - hostC	0.103
hostB - hostD	0.104
hostC - hostD	0.105

The VPI and VCI values at each end of the connection may of course differ, but for simplicity we assume that they are the same. Next we need to configure the ATM interfaces on each host:

```
hostA# ifconfig hatm0 192.168.173.1 up
hostB# ifconfig hatm0 192.168.173.2 up
hostC# ifconfig hatm0 192.168.173.3 up
hostD# ifconfig hatm0 192.168.173.4 up
```

assuming that the ATM interface is `hatm0` on all hosts. Now the PVCs need to be configured on `hostA` (we assume that they are already configured on the ATM switches, you need to consult the manual for the switch on how to do this).

```
hostA# atmconfig natm add 192.168.173.2 hatm0 0 100 llc/snap ubr
hostA# atmconfig natm add 192.168.173.3 hatm0 0 101 llc/snap ubr
hostA# atmconfig natm add 192.168.173.4 hatm0 0 102 llc/snap ubr

hostB# atmconfig natm add 192.168.173.1 hatm0 0 100 llc/snap ubr
hostB# atmconfig natm add 192.168.173.3 hatm0 0 103 llc/snap ubr
hostB# atmconfig natm add 192.168.173.4 hatm0 0 104 llc/snap ubr

hostC# atmconfig natm add 192.168.173.1 hatm0 0 101 llc/snap ubr
hostC# atmconfig natm add 192.168.173.2 hatm0 0 103 llc/snap ubr
hostC# atmconfig natm add 192.168.173.4 hatm0 0 105 llc/snap ubr

hostD# atmconfig natm add 192.168.173.1 hatm0 0 102 llc/snap ubr
hostD# atmconfig natm add 192.168.173.2 hatm0 0 104 llc/snap ubr
hostD# atmconfig natm add 192.168.173.3 hatm0 0 105 llc/snap ubr
```

Of course other traffic contracts than UBR can be used given the ATM adapter supports those. In this case the name of the traffic contract is followed by the parameters of the traffic. Help for the atmconfig(8) tool can be obtained with:

```
# atmconfig help natm add
```

or in the atmconfig(8) manual page.

The same configuration can also be done via `/etc/rc.conf`. For `hostA` this would look like:

```
network_interfaces="lo0 hatm0"
ifconfig_hatm0="inet 192.168.173.1 up"
natm_static_routes="hostB hostC hostD"
route_hostB="192.168.173.2 hatm0 0 100 llc/snap ubr"
route_hostC="192.168.173.3 hatm0 0 101 llc/snap ubr"
route_hostD="192.168.173.4 hatm0 0 102 llc/snap ubr"
```

The current state of all CLIP routes can be obtained with:

```
hostA# atmconfig natm show
```

31.13 Common Address Redundancy Protocol (CARP)

The Common Address Redundancy Protocol, or CARP allows multiple hosts to share the same IP address. In some configurations, this may be used for availability or load balancing. Hosts may use separate IP addresses as well, as in the example provided here.

To enable support for CARP, the FreeBSD kernel must be rebuilt with the following option:

```
device carp
```

CARP functionality should now be available and may be tuned via several `sysctl` OIDs:

OID	Description
`net.inet.carp.allow`	Accept incoming CARP packets. Enabled by default.
`net.inet.carp.preempt`	This option downs all of the CARP interfaces on the host when one of them goes down. Disabled by default
`net.inet.carp.log`	A value of `0` disables any logging. A Value of `1` enables logging of bad CARP packets. Values greater than `1` enables logging of state changes for the CARP interfaces. The default value is `1`.
`net.inet.carp.arpbalance`	Balance local network traffic using ARP. Disabled by default.
`net.inet.carp.suppress_preempt`	A read only OID showing the status of preemption suppression. Preemption can be suppressed if link on an interface is down. A value of `0`, means that preemption is not suppressed. Every problem increments this OID.

The CARP devices themselves may be created via the `ifconfig` command:

```
# ifconfig carp0 create
```

In a real environment, these interfaces will need unique identification numbers known as a VHID. This VHID or Virtual Host Identification will be used to distinguish the host on the network.

31.13.1 Using CARP For Server Availability (CARP)

One use of CARP, as noted above, is for server availability. This example will provide failover support for three hosts, all with unique IP addresses and providing the same web content. These machines will act in conjunction with a Round Robin DNS configuration. The failover machine will have two additional CARP interfaces, one for each of the content server's IPs. When a failure occurs, the failover server should pick up the failed machine's IP address. This means the failure should go completely unnoticed to the user. The failover server requires identical content and services as the other content servers it is expected to pick up load for.

The two machines should be configured identically other than their issued hostnames and VHIDs. This example calls these machines `hosta.example.org` and `hostb.example.org` respectively. First, the required lines for a CARP configuration have to be added to `rc.conf`. For `hosta.example.org`, the `rc.conf` file should contain the following lines:

```
hostname="hosta.example.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"
```

On `hostb.example.org` the following lines should be in `rc.conf`:

```
hostname="hostb.example.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"
```

> **Note:** It is very important that the passwords, specified by the `pass` option to `ifconfig`, are identical. The `carp` devices will only listen to and accept advertisements from machines with the correct password. The VHID must also be different for each machine.

The third machine, `provider.example.org`, should be prepared so that it may handle failover from either host. This machine will require two `carp` devices, one to handle each host. The appropriate `rc.conf` configuration lines will be similar to the following:

```
hostname="provider.example.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"
```

Having the two `carp` devices will allow `provider.example.org` to notice and pick up the IP address of either machine should it stop responding.

> **Note:** The default FreeBSD kernel *may* have preemption enabled. If so, `provider.example.org` may not relinquish the IP address back to the original content server. In this case, an administrator may have to manually force the IP back to the master. The following command should be issued on `provider.example.org`:
>
> ```
> # ifconfig carp0 down && ifconfig carp0 up
> ```
>
> This should be done on the `carp` interface which corresponds to the correct host.

At this point, CARP should be completely enabled and available for testing. For testing, either networking has to be restarted or the machines need to be rebooted.

More information is always available in the carp(4) manual page.

V. Appendices

Appendix A.
Obtaining FreeBSD

A.1 CDROM and DVD Publishers

A.1.1 Retail Boxed Products

FreeBSD is available as a boxed product (FreeBSD CDs, additional software, and printed documentation) from several retailers:

- CompUSA
  WWW: http://www.compusa.com/
- Frys Electronics
  WWW: http://www.frys.com/

A.1.2 CD and DVD Sets

FreeBSD CD and DVD sets are available from many online retailers:

- FreeBSD Mall, Inc.
  700 Harvest Park Ste F
  Brentwood, CA 94513
  USA
  Phone: +1 925 240-6652
  Fax: +1 925 674-0821
  Email: <info@freebsdmall.com>
  WWW: http://www.freebsdmall.com/
- Dr. Hinner EDV
  St. Augustinus-Str. 10
  D-81825 München
  Germany
  Phone: (089) 428 419
  WWW: http://www.hinner.de/linux/freebsd.html
- Ikarios
  22-24 rue Voltaire
  92000 Nanterre
  France
  WWW: http://ikarios.com/form/#freebsd
- JMC Software

Ireland
Phone: 353 1 6291282
WWW: http://www.thelinuxmall.com

- The Linux Emporium
  Hilliard House, Lester Way
  Wallingford
  OX10 9TA
  United Kingdom
  Phone: +44 1491 837010
  Fax: +44 1491 837016
  WWW: http://www.linuxemporium.co.uk/products/bsd/

- Linux+ DVD Magazine
  Lewartowskiego 6
  Warsaw
  00-190
  Poland
  Phone: +48 22 860 18 18
  Email: <editors@lpmagazine.org>
  WWW: http://www.lpmagazine.org/

- Linux System Labs Australia
  21 Ray Drive
  Balwyn North
  VIC - 3104
  Australia
  Phone: +61 3 9857 5918
  Fax: +61 3 9857 8974
  WWW: http://www.lsl.com.au

- LinuxCenter.Ru
  Galernaya Street, 55
  Saint-Petersburg
  190000
  Russia
  Phone: +7-812-3125208
  Email: <info@linuxcenter.ru>
  WWW: http://linuxcenter.ru/shop/freebsd

A.1.3 Distributors

If you are a reseller and want to carry FreeBSD CDROM products, please contact a distributor:

- Cylogistics
  809B Cuesta Dr., #2149
  Mountain View, CA 94040
  USA
  Phone: +1 650 694-4949

Fax: +1 650 694-4953
Email: <sales@cylogistics.com>
WWW: http://www.cylogistics.com/

- Ingram Micro
  1600 E. St. Andrew Place
  Santa Ana, CA 92705-4926
  USA
  Phone: 1 (800) 456-8000
  WWW: http://www.ingrammicro.com/
- Kudzu, LLC
  7375 Washington Ave. S.
  Edina, MN 55439
  USA
  Phone: +1 952 947-0822
  Fax: +1 952 947-0876
  Email: <sales@kudzuenterprises.com>
- LinuxCenter.Kz
  Ust-Kamenogorsk
  Kazakhstan
  Phone: +7-705-501-6001
  Email: <info@linuxcenter.kz>
  WWW: http://linuxcenter.kz/page.php?page=fr
- LinuxCenter.Ru
  Galernaya Street, 55
  Saint-Petersburg
  190000
  Russia
  Phone: +7-812-3125208
  Email: <info@linuxcenter.ru>
  WWW: http://linuxcenter.ru/freebsd
- Navarre Corp
  7400 49th Ave South
  New Hope, MN 55428
  USA
  Phone: +1 763 535-8333
  Fax: +1 763 535-0341
  WWW: http://www.navarre.com/

A.2 FTP Sites

The official sources for FreeBSD are available via anonymous FTP from a worldwide set of mirror sites. The site ftp://ftp.FreeBSD.org/pub/FreeBSD/ is well connected and allows a large number of connections to it, but you are probably better off finding a “closer” mirror site (especially if you decide to set up some sort of mirror site).

The FreeBSD mirror sites database (http://mirrorlist.FreeBSD.org/) is more accurate than the mirror listing in the Handbook, as it gets its information from the DNS rather than relying on static lists of hosts.

Additionally, FreeBSD is available via anonymous FTP from the following mirror sites. If you choose to obtain FreeBSD via anonymous FTP, please try to use a site near you. The mirror sites listed as “Primary Mirror Sites” typically have the entire FreeBSD archive (all the currently available versions for each of the architectures) but you will probably have faster download times from a site that is in your country or region. The regional sites carry the most recent versions for the most popular architecture(s) but might not carry the entire FreeBSD archive. All sites provide access via anonymous FTP but some sites also provide access via other methods. The access methods available for each site are provided in parentheses after the hostname.

Central Servers, Primary Mirror Sites, Argentina, Armenia, Australia, Austria, Brazil, Bulgaria, Canada, China, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hong Kong, Hungary, Iceland, Indonesia, Ireland, Israel, Italy, Japan, Korea, Latvia, Lithuania, Netherlands, Norway, Poland, Portugal, Romania, Russia, Saudi Arabia, Singapore, Slovak Republic, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Turkey, Ukraine, United Kingdom, USA.

(as of 2010/11/13 13:50:55 UTC)

Central Servers

- ftp://ftp.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp.FreeBSD.org/pub/FreeBSD/))

Primary Mirror Sites

In case of problems, please contact the hostmaster <mirror-admin@FreeBSD.org> for this domain.

- 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://ftp8.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp9.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://ftp12.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/))

Argentina

In case of problems, please contact the hostmaster <hostmaster@ar.FreeBSD.org> for this domain.

- ftp://ftp.ar.FreeBSD.org/pub/FreeBSD/ (ftp)

Armenia

In case of problems, please contact the hostmaster <hostmaster@am.FreeBSD.org> for this domain.

- ftp://ftp1.am.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp1.am.FreeBSD.org/pub/FreeBSD/) / rsync)

Australia

In case of problems, please contact the hostmaster <hostmaster@au.FreeBSD.org> for this domain.

- ftp://ftp.au.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.au.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.au.FreeBSD.org/pub/FreeBSD/ (ftp)

Austria

In case of problems, please contact the hostmaster <hostmaster@at.FreeBSD.org> for this domain.

- 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/))
- ftp://ftp2.at.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp2.at.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp2.at.FreeBSD.org/pub/FreeBSD/) / rsync / rsyncv6)

Brazil

In case of problems, please contact the hostmaster <hostmaster@br.FreeBSD.org> for this domain.

- ftp://ftp.br.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.br.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp2.br.FreeBSD.org/pub/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)
- ftp5.br.FreeBSD.org
- ftp://ftp6.br.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.br.FreeBSD.org/pub/FreeBSD/ (ftp)

Bulgaria

In case of problems, please contact the hostmaster <hostmaster@bg.FreeBSD.org> for this domain.

- ftp://ftp.bg.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp2.bg.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)

Canada

In case of problems, please contact the hostmaster <hostmaster@ca.FreeBSD.org> for this domain.

- ftp://ftp.ca.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.ca.FreeBSD.org/ (ftp)
- ftp://ftp3.ca.FreeBSD.org/pub/FreeBSD/ (ftp)

China

In case of problems, please contact the hostmaster <hostmaster@cn.FreeBSD.org> for this domain.

- ftp://ftp.cn.FreeBSD.org/pub/FreeBSD/ (ftp)

Czech Republic

In case of problems, please contact the hostmaster <hostmaster@cz.FreeBSD.org> for this domain.

- ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/) / http (http://ftp.cz.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp.cz.FreeBSD.org/pub/FreeBSD/) / rsync / rsyncv6)
- ftp://ftp2.cz.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.cz.FreeBSD.org/pub/FreeBSD/))

Denmark

In case of problems, please contact the hostmaster <hostmaster@dk.FreeBSD.org> for this domain.

- ftp://ftp.dk.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp.dk.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp.dk.FreeBSD.org/pub/FreeBSD/))

Estonia

In case of problems, please contact the hostmaster <hostmaster@ee.FreeBSD.org> for this domain.

- ftp://ftp.ee.FreeBSD.org/pub/FreeBSD/ (ftp)

Finland

In case of problems, please contact the hostmaster <hostmaster@fi.FreeBSD.org> for this domain.

- ftp://ftp.fi.FreeBSD.org/pub/FreeBSD/ (ftp)

France

In case of problems, please contact the hostmaster <hostmaster@fr.FreeBSD.org> for this domain.

- ftp://ftp.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp1.fr.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp1.fr.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp2.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.fr.FreeBSD.org/pub/FreeBSD/ (ftp)

Germany

In case of problems, please contact the hostmaster <de-bsd-hubs@de.FreeBSD.org> for this domain.

- 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://ftp3.de.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.de.FreeBSD.org/FreeBSD/ (ftp / http (http://ftp4.de.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp5.de.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.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)

Greece

In case of problems, please contact the hostmaster <hostmaster@gr.FreeBSD.org> for this domain.

- ftp://ftp.gr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.gr.FreeBSD.org/pub/FreeBSD/ (ftp)

Hong Kong

- ftp://ftp.hk.FreeBSD.org/pub/FreeBSD/ (ftp)

Hungary

In case of problems, please contact the hostmaster <hostmaster@hu.FreeBSD.org> for this domain.

- ftp://ftp.hu.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.hu.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp2.hu.FreeBSD.org/pub/FreeBSD/ (ftp)

Iceland

In case of problems, please contact the hostmaster <hostmaster@is.FreeBSD.org> for this domain.

- ftp://ftp.is.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)

Indonesia

In case of problems, please contact the hostmaster <hostmaster@id.FreeBSD.org> for this domain.

- ftp://ftp.id.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.id.FreeBSD.org/) / rsync)

Ireland

In case of problems, please contact the hostmaster <hostmaster@ie.FreeBSD.org> for this domain.

- ftp://ftp.ie.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.ie.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.ie.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp3.ie.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp3.ie.FreeBSD.org/pub/FreeBSD/) / rsync)

Israel

In case of problems, please contact the hostmaster <hostmaster@il.FreeBSD.org> for this domain.

- ftp://ftp.il.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6)

Italy

In case of problems, please contact the hostmaster <hostmaster@it.FreeBSD.org> for this domain.

- ftp://ftp.it.FreeBSD.org/pub/FreeBSD/ (ftp)

Japan

In case of problems, please contact the hostmaster <hostmaster@jp.FreeBSD.org> for this domain.

- 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)

Korea

In case of problems, please contact the hostmaster <`hostmaster@kr.FreeBSD.org`> for this domain.

- 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/))

Latvia

In case of problems, please contact the hostmaster <`hostmaster@lv.FreeBSD.org`> for this domain.

- ftp://ftp.lv.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.lv.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp2.lv.FreeBSD.org/pub/FreeBSD/ (ftp)

Lithuania

In case of problems, please contact the hostmaster <`hostmaster@lt.FreeBSD.org`> for this domain.

- ftp://ftp.lt.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.lt.FreeBSD.org/pub/FreeBSD/))

Netherlands

In case of problems, please contact the hostmaster <`hostmaster@nl.FreeBSD.org`> for this domain.

- 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)

Norway

In case of problems, please contact the hostmaster <`hostmaster@no.FreeBSD.org`> for this domain.

- ftp://ftp.no.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp3.no.FreeBSD.org/pub/FreeBSD/ (ftp)

Poland

In case of problems, please contact the hostmaster <`hostmaster@pl.FreeBSD.org`> for this domain.

- 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)

Portugal

In case of problems, please contact the hostmaster <`hostmaster@pt.FreeBSD.org`> for this domain.

- ftp://ftp.pt.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.pt.FreeBSD.org/pub/freebsd/ (ftp)
- ftp://ftp4.pt.FreeBSD.org/pub/ISO/FreeBSD/ (ftp)

Romania

In case of problems, please contact the hostmaster <`hostmaster@ro.FreeBSD.org`> for this domain.

- ftp://ftp.ro.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp1.ro.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp1.ro.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp1.ro.FreeBSD.org/pub/FreeBSD/))

Russia

In case of problems, please contact the hostmaster <`hostmaster@ru.FreeBSD.org`> for this domain.

- 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://ftp3.ru.FreeBSD.org/pub/FreeBSD/ (ftp)
- 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)

Saudi Arabia

In case of problems, please contact the hostmaster <`ftpadmin@isu.net.sa`> for this domain.

- ftp://ftp.isu.net.sa/pub/ftp.freebsd.org/ (ftp)

Singapore

In case of problems, please contact the hostmaster <`hostmaster@sg.FreeBSD.org`> for this domain.

- ftp://ftp.sg.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.sg.FreeBSD.org/pub/FreeBSD/) / rsync)

Slovak Republic

In case of problems, please contact the hostmaster <hostmaster@sk.FreeBSD.org> for this domain.

- ftp://ftp.sk.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp.sk.FreeBSD.org/pub/FreeBSD/) / http (http://ftp.sk.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp.sk.FreeBSD.org/pub/FreeBSD/) / rsync / rsyncv6)
- ftp://ftp2.sk.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp2.sk.FreeBSD.org/pub/FreeBSD/) / http (http://ftp2.sk.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp2.sk.FreeBSD.org/pub/FreeBSD/))

Slovenia

In case of problems, please contact the hostmaster <hostmaster@si.FreeBSD.org> for this domain.

- ftp://ftp.si.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.si.FreeBSD.org/pub/FreeBSD/ (ftp)

South Africa

In case of problems, please contact the hostmaster <hostmaster@za.FreeBSD.org> for this domain.

- ftp://ftp.za.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.za.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.za.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.za.FreeBSD.org/pub/FreeBSD/ (ftp)

Spain

In case of problems, please contact the hostmaster <hostmaster@es.FreeBSD.org> for this domain.

- ftp://ftp.es.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.es.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp2.es.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.es.FreeBSD.org/pub/FreeBSD/ (ftp)

Sweden

In case of problems, please contact the hostmaster <hostmaster@se.FreeBSD.org> for this domain.

- ftp://ftp.se.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.se.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.se.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/) / rsync / rsyncv6)
- ftp://ftp5.se.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp5.se.FreeBSD.org/) / rsync)

Switzerland

In case of problems, please contact the hostmaster <hostmaster@ch.FreeBSD.org> for this domain.

- ftp://ftp.ch.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.ch.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp2.ch.FreeBSD.org/mirror/FreeBSD/ (ftp / ftpv6 (ftp://ftp2.ch.FreeBSD.org/mirror/FreeBSD/) / http (http://ftp2.ch.FreeBSD.org/ftp/mirror/FreeBSD/) / httpv6 (http://ftp2.ch.FreeBSD.org/ftp/mirror/FreeBSD/))

Taiwan

In case of problems, please contact the hostmaster <hostmaster@tw.FreeBSD.org> for this domain.

- 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)

Turkey

- ftp://ftp.tr.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.tr.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp2.tr.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)

Ukraine

- ftp://ftp.ua.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.ua.FreeBSD.org/pub/FreeBSD/))

- ftp://ftp2.ua.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.ua.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp7.ua.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.ua.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp8.ua.FreeBSD.org/FreeBSD/))
- ftp://ftp11.ua.FreeBSD.org/pub/FreeBSD/ (ftp)

United Kingdom

In case of problems, please contact the hostmaster <`hostmaster@uk.FreeBSD.org`> for this domain.

- ftp://ftp.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.uk.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.uk.FreeBSD.org/) / rsync)
- ftp://ftp3.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.uk.FreeBSD.org/pub/FreeBSD/ (ftp)

USA

In case of problems, please contact the hostmaster <`hostmaster@us.FreeBSD.org`> for this domain.

- 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/))
- ftp://ftp5.us.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp6.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.us.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp7.us.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp8.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp9.us.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp9.us.FreeBSD.org/pub/os/FreeBSD/))
- ftp://ftp10.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp11.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp12.us.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- 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)

A.3 BitTorrent

The ISO images for the basic release CDs are available via BitTorrent. A collection of torrent files to download the images is available at http://torrents.freebsd.org:8080 (http://torrents.freebsd.org:8080/)

The BitTorrent client software is available from the `net-p2p/py-bittorrent` port, or a precompiled package.

After downloading the ISO image with BitTorrent, you may burn it to CD or DVD media as described in Section 18.6.3, burncd.

A.4 Anonymous CVS

A.4.1 Introduction

Anonymous CVS (or, as it is otherwise known, *anoncvs*) is a feature provided by the CVS utilities bundled with FreeBSD for synchronizing with a remote CVS repository. Among other things, it allows users of FreeBSD to perform, with no special privileges, read-only CVS operations against one of the FreeBSD project's official anoncvs servers. To use it, one simply sets the `CVSROOT` environment variable to point at the appropriate anoncvs server, provides the well-known password "anoncvs" with the `cvs login` command, and then uses the cvs(1) command to access it like any local repository.

> **Note:** The `cvs login` command, stores the passwords that are used for authenticating to the CVS server in a file called `.cvspass` in your `HOME` directory. If this file does not exist, you might get an error when trying to use `cvs login` for the first time. Just make an empty `.cvspass` file, and retry to login.

While it can also be said that the CVSup and *anoncvs* services both perform essentially the same function, there are various trade-offs which can influence the user's choice of synchronization methods. In a nutshell, **CVSup** is much more efficient in its usage of network resources and is by far the most technically sophisticated of the two, but at a price. To use **CVSup**, a special client must first be installed and configured before any bits can be grabbed, and then only in the fairly large chunks which **CVSup** calls *collections*.

Anoncvs, by contrast, can be used to examine anything from an individual file to a specific program (like `ls` or `grep`) by referencing the CVS module name. Of course, **anoncvs** is also only good for read-only operations on the CVS repository, so if it is your intention to support local development in one repository shared with the FreeBSD project bits then **CVSup** is really your only option.

A.4.2 Using Anonymous CVS

Configuring cvs(1) to use an Anonymous CVS repository is a simple matter of setting the `CVSROOT` environment variable to point to one of the FreeBSD project's *anoncvs* servers. At the time of this writing, the following servers are available:

- *France*: :pserver:anoncvs@anoncvs.fr.FreeBSD.org:/home/ncvs (For pserver mode, use `cvs login` and enter the password "anoncvs" when prompted. For ssh, no password is required.)
- *Taiwan*: :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs (For pserver mode, use `cvs login` and enter any password when prompted. For ssh, no password is required.)

```
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
```

- *USA*: anoncvs@anoncvs1.FreeBSD.org:/home/ncvs (For ssh, use ssh version 2 and no password is required.)

  ```
  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
  ```

Since CVS allows one to "check out" virtually any version of the FreeBSD sources that ever existed (or, in some cases, will exist), you need to be familiar with the revision (`-r`) flag to cvs(1) and what some of the permissible values for it in the FreeBSD Project repository are.

There are two kinds of tags, revision tags and branch tags. A revision tag refers to a specific revision. Its meaning stays the same from day to day. A branch tag, on the other hand, refers to the latest revision on a given line of development, at any given time. Because a branch tag does not refer to a specific revision, it may mean something different tomorrow than it means today.

Section A.7 contains revision tags that users might be interested in. Again, none of these are valid for the Ports Collection since the Ports Collection does not have multiple branches of development.

When you specify a branch tag, you normally receive the latest versions of the files on that line of development. If you wish to receive some past version, you can do so by specifying a date with the `-D date` flag. See the cvs(1) manual page for more details.

A.4.3 Examples

While it really is recommended that you read the manual page for cvs(1) thoroughly before doing anything, here are some quick examples which essentially show how to use Anonymous CVS:

Example A-1. Checking Out Something from -CURRENT (ls(1)):

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
At the prompt, enter any word for "password".
% cvs co ls
```

Example A-2. Using SSH to check out the `src/` tree:

```
% 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.
```

Example A-3. Checking Out the Version of ls(1) in the 8-STABLE Branch:

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
At the prompt, enter any word for "password".
% cvs co -rRELENG_8 ls
```

Example A-4. Creating a List of Changes (as Unified Diffs) to ls(1)

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
At the prompt, enter any word for "password".
% cvs rdiff -u -rRELENG_8_0_0_RELEASE -rRELENG_8_1_0_RELEASE ls
```

Example A-5. Finding Out What Other Module Names Can Be Used:

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
At the prompt, enter any word for "password".
% cvs co modules
% more modules/modules
```

A.4.4 Other Resources

The following additional resources may be helpful in learning CVS:

- CVS Tutorial (http://users.csc.calpoly.edu/~gfisher/classes/308/handouts/cvs-basics.html) from California Polytechnic State University.
- CVS Home (http://ximbiot.com/cvs/wiki/), the CVS development and support community.
- CVSweb (http://www.FreeBSD.org/cgi/cvsweb.cgi) is the FreeBSD Project web interface for CVS.

A.5 Using CTM

CTM is a method for keeping a remote directory tree in sync with a central one. It has been developed for usage with FreeBSD's source trees, though other people may find it useful for other purposes as time goes by. Little, if any, documentation currently exists at this time on the process of creating deltas, so contact the ctm-users (http://lists.FreeBSD.org/mailman/listinfo/ctm-users) mailing list for more information and if you wish to use **CTM** for other things.

A.5.1 Why Should I Use CTM?

CTM will give you a local copy of the FreeBSD source trees. There are a number of "flavors" of the tree available. Whether you wish to track the entire CVS tree or just one of the branches, **CTM** can provide you the information. If you are an active developer on FreeBSD, but have lousy or non-existent TCP/IP connectivity, or simply wish to have the changes automatically sent to you, **CTM** was made for you. You will need to obtain up to three deltas per day for the most active branches. However, you should consider having them sent by automatic email. The sizes of the updates are always kept as small as possible. This is typically less than 5K, with an occasional (one in ten) being 10-50K and every now and then a large 100K+ or more coming around.

You will also need to make yourself aware of the various caveats related to working directly from the development sources rather than a pre-packaged release. This is particularly true if you choose the “current” sources. It is recommended that you read Staying current with FreeBSD.

A.5.2 What Do I Need to Use CTM?

You will need two things: The **CTM** program, and the initial deltas to feed it (to get up to “current” levels).

The **CTM** program has been part of FreeBSD ever since version 2.0 was released, and lives in `/usr/src/usr.sbin/ctm` if you have a copy of the source available.

The “deltas” you feed **CTM** can be had two ways, FTP or email. If you have general FTP access to the Internet then the following FTP sites support access to **CTM**:

ftp://ftp.FreeBSD.org/pub/FreeBSD/CTM/

or see section mirrors.

FTP the relevant directory and fetch the `README` file, starting from there.

If you wish to get your deltas via email:

Subscribe to one of the **CTM** distribution lists. ctm-cvs-cur (http://lists.FreeBSD.org/mailman/listinfo/ctm-cvs-cur) supports the entire CVS tree. ctm-src-cur (http://lists.FreeBSD.org/mailman/listinfo/ctm-src-cur) supports the head of the development branch. ctm-src-7 (http://lists.FreeBSD.org/mailman/listinfo/ctm-src-7) supports the 7.X release branch, etc.. (If you do not know how to subscribe yourself to a list, click on the list name above or go to http://lists.FreeBSD.org/mailman/listinfo and click on the list that you wish to subscribe to. The list page should contain all of the necessary subscription instructions.)

When you begin receiving your **CTM** updates in the mail, you may use the `ctm_rmail` program to unpack and apply them. You can actually use the `ctm_rmail` program directly from a entry in `/etc/aliases` if you want to have the process run in a fully automated fashion. Check the `ctm_rmail` manual page for more details.

> **Note:** No matter what method you use to get the **CTM** deltas, you should subscribe to the ctm-announce (http://lists.FreeBSD.org/mailman/listinfo/ctm-announce) mailing list. In the future, this will be the only place where announcements concerning the operations of the **CTM** system will be posted. Click on the list name above and follow the instructions to subscribe to the list.

A.5.3 Using CTM for the First Time

Before you can start using **CTM** deltas, you will need to get to a starting point for the deltas produced subsequently to it.

First you should determine what you already have. Everyone can start from an “empty” directory. You must use an initial “Empty” delta to start off your **CTM** supported tree. At some point it is intended that one of these “started” deltas be distributed on the CD for your convenience, however, this does not currently happen.

Since the trees are many tens of megabytes, you should prefer to start from something already at hand. If you have a -RELEASE CD, you can copy or extract an initial source from it. This will save a significant transfer of data.

You can recognize these “starter” deltas by the `X` appended to the number (`src-cur.3210XEmpty.gz` for instance). The designation following the `X` corresponds to the origin of your initial “seed”. `Empty` is an empty directory. As a

rule a base transition from `Empty` is produced every 100 deltas. By the way, they are large! 70 to 80 Megabytes of `gzip`'d data is common for the `XEmpty` deltas.

Once you have picked a base delta to start from, you will also need all deltas with higher numbers following it.

A.5.4 Using CTM in Your Daily Life

To apply the deltas, simply say:

```
# cd /where/ever/you/want/the/stuff
# ctm -v -v /where/you/store/your/deltas/src-xxx.*
```

CTM understands deltas which have been put through `gzip`, so you do not need to `gunzip` them first, this saves disk space.

Unless it feels very secure about the entire process, **CTM** will not touch your tree. To verify a delta you can also use the `-c` flag and **CTM** will not actually touch your tree; it will merely verify the integrity of the delta and see if it would apply cleanly to your current tree.

There are other options to **CTM** as well, see the manual pages or look in the sources for more information.

That is really all there is to it. Every time you get a new delta, just run it through **CTM** to keep your sources up to date.

Do not remove the deltas if they are hard to download again. You just might want to keep them around in case something bad happens. Even if you only have floppy disks, consider using `fdwrite` to make a copy.

A.5.5 Keeping Your Local Changes

As a developer one would like to experiment with and change files in the source tree. **CTM** supports local modifications in a limited way: before checking for the presence of a file `foo`, it first looks for `foo.ctm`. If this file exists, **CTM** will operate on it instead of `foo`.

This behavior gives us a simple way to maintain local changes: simply copy the files you plan to modify to the corresponding file names with a `.ctm` suffix. Then you can freely hack the code, while **CTM** keeps the `.ctm` file up-to-date.

A.5.6 Other Interesting CTM Options

A.5.6.1 Finding Out Exactly What Would Be Touched by an Update

You can determine the list of changes that **CTM** will make on your source repository using the `-l` option to **CTM**.

This is useful if you would like to keep logs of the changes, pre- or post- process the modified files in any manner, or just are feeling a tad paranoid.

A.5.6.2 Making Backups Before Updating

Sometimes you may want to backup all the files that would be changed by a **CTM** update.

Specifying the `-B backup-file` option causes **CTM** to backup all files that would be touched by a given **CTM** delta to `backup-file`.

A.5.6.3 Restricting the Files Touched by an Update

Sometimes you would be interested in restricting the scope of a given **CTM** update, or may be interested in extracting just a few files from a sequence of deltas.

You can control the list of files that **CTM** would operate on by specifying filtering regular expressions using the `-e` and `-x` options.

For example, to extract an up-to-date copy of `lib/libc/Makefile` from your collection of saved **CTM** deltas, run the commands:

```
# cd /where/ever/you/want/to/extract/it/
# ctm -e '^lib/libc/Makefile' ~ctm/src-xxx.*
```

For every file specified in a **CTM** delta, the `-e` and `-x` options are applied in the order given on the command line. The file is processed by **CTM** only if it is marked as eligible after all the `-e` and `-x` options are applied to it.

A.5.7 Future Plans for CTM

Tons of them:

- Use some kind of authentication into the **CTM** system, so as to allow detection of spoofed **CTM** updates.
- Clean up the options to **CTM**, they became confusing and counter intuitive.

A.5.8 Miscellaneous Stuff

There is a sequence of deltas for the `ports` collection too, but interest has not been all that high yet.

A.5.9 CTM Mirrors

CTM/FreeBSD is available via anonymous FTP from the following mirror sites. If you choose to obtain **CTM** via anonymous FTP, please try to use a site near you.

In case of problems, please contact the ctm-users (http://lists.FreeBSD.org/mailman/listinfo/ctm-users) mailing list.

California, Bay Area, official source

- ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CTM/

South Africa, backup server for old deltas

- ftp://ftp.za.FreeBSD.org/pub/FreeBSD/CTM/

Taiwan/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/

If you did not find a mirror near to you or the mirror is incomplete, try to use a search engine such as alltheweb (http://www.alltheweb.com/).

A.6 Using CVSup

A.6.1 Introduction

CVSup is a software package for distributing and updating source trees from a master CVS repository on a remote server host. The FreeBSD sources are maintained in a CVS repository on a central development machine in California. With **CVSup**, FreeBSD users can easily keep their own source trees up to date.

CVSup uses the so-called *pull* model of updating. Under the pull model, each client asks the server for updates, if and when they are wanted. The server waits passively for update requests from its clients. Thus all updates are instigated by the client. The server never sends unsolicited updates. Users must either run the **CVSup** client manually to get an update, or they must set up a `cron` job to run it automatically on a regular basis.

The term **CVSup**, capitalized just so, refers to the entire software package. Its main components are the client `cvsup` which runs on each user's machine, and the server `cvsupd` which runs at each of the FreeBSD mirror sites.

As you read the FreeBSD documentation and mailing lists, you may see references to **sup**. **Sup** was the predecessor of **CVSup**, and it served a similar purpose. **CVSup** is used much in the same way as sup and, in fact, uses configuration files which are backward-compatible with `sup`'s. **Sup** is no longer used in the FreeBSD project, because **CVSup** is both faster and more flexible.

> **Note:** The **csup** utility is a rewrite of the **CVSup** software in C. Its biggest advantage is, that it is faster and does not depend on the Modula-3 language, thus you do not need to install it as a requirement. Moreover you can use it out-of-the-box, since it is included in the base system. If you decided to use **csup**, just skip the steps on the installation of **CVSup** and substitute the references of **CVSup** with **csup** while following the remainder of this article.

A.6.2 Installation

The easiest way to install **CVSup** is to use the precompiled `net/cvsup` package from the FreeBSD packages collection. If you prefer to build **CVSup** from source, you can use the `net/cvsup` port instead. But be forewarned: the `net/cvsup` port depends on the Modula-3 system, which takes a substantial amount of time and disk space to download and build.

> **Note:** If you are going to be using **CVSup** on a machine which will not have **Xorg** installed, such as a server, be sure to use the port which does not include the **CVSup** GUI, `net/cvsup-without-gui`.

A.6.3 CVSup Configuration

CVSup's operation is controlled by a configuration file called the `supfile`. There are some sample `supfiles` in the directory `/usr/share/examples/cvsup/`.

The information in a `supfile` answers the following questions for **CVSup**:

- Which files do you want to receive?
- Which versions of them do you want?
- Where do you want to get them from?
- Where do you want to put them on your own machine?
- Where do you want to put your status files?

In the following sections, we will construct a typical `supfile` by answering each of these questions in turn. First, we describe the overall structure of a `supfile`.

A `supfile` is a text file. Comments begin with # and extend to the end of the line. Lines that are blank and lines that contain only comments are ignored.

Each remaining line describes a set of files that the user wishes to receive. The line begins with the name of a "collection", a logical grouping of files defined by the server. The name of the collection tells the server which files you want. After the collection name come zero or more fields, separated by white space. These fields answer the questions listed above. There are two types of fields: flag fields and value fields. A flag field consists of a keyword standing alone, e.g., `delete` or `compress`. A value field also begins with a keyword, but the keyword is followed without intervening white space by `=` and a second word. For example, `release=cvs` is a value field.

A `supfile` typically specifies more than one collection to receive. One way to structure a `supfile` is to specify all of the relevant fields explicitly for each collection. However, that tends to make the `supfile` lines quite long, and it is inconvenient because most fields are the same for all of the collections in a `supfile`. **CVSup** provides a defaulting mechanism to avoid these problems. Lines beginning with the special pseudo-collection name `*default` can be used to set flags and values which will be used as defaults for the subsequent collections in the `supfile`. A default value can be overridden for an individual collection, by specifying a different value with the collection itself. Defaults can also be changed or augmented in mid-supfile by additional `*default` lines.

With this background, we will now proceed to construct a `supfile` for receiving and updating the main source tree of FreeBSD-CURRENT.

- Which files do you want to receive?

The files available via **CVSup** are organized into named groups called “collections”. The collections that are available are described in the following section. In this example, we wish to receive the entire main source tree for the FreeBSD system. There is a single large collection `src-all` which will give us all of that. As a first step toward constructing our `supfile`, we simply list the collections, one per line (in this case, only one line):

```
src-all
```

- Which version(s) of them do you want?

  With **CVSup**, you can receive virtually any version of the sources that ever existed. That is possible because the **cvsupd** server works directly from the CVS repository, which contains all of the versions. You specify which one of them you want using the `tag=` and `date=` value fields.

  > **Warning:** Be very careful to specify any `tag=` fields correctly. Some tags are valid only for certain collections of files. If you specify an incorrect or misspelled tag, **CVSup** will delete files which you probably do not want deleted. In particular, use *only* `tag=.` for the `ports-*` collections.

  The `tag=` field names a symbolic tag in the repository. There are two kinds of tags, revision tags and branch tags. A revision tag refers to a specific revision. Its meaning stays the same from day to day. A branch tag, on the other hand, refers to the latest revision on a given line of development, at any given time. Because a branch tag does not refer to a specific revision, it may mean something different tomorrow than it means today.

  Section A.7 contains branch tags that users might be interested in. When specifying a tag in **CVSup**’s configuration file, it must be preceded with `tag=` (`RELENG_8` will become `tag=RELENG_8`). Keep in mind that only the `tag=.` is relevant for the Ports Collection.

  > **Warning:** Be very careful to type the tag name exactly as shown. **CVSup** cannot distinguish between valid and invalid tags. If you misspell the tag, **CVSup** will behave as though you had specified a valid tag which happens to refer to no files at all. It will delete your existing sources in that case.

  When you specify a branch tag, you normally receive the latest versions of the files on that line of development. If you wish to receive some past version, you can do so by specifying a date with the `date=` value field. The cvsup(1) manual page explains how to do that.

  For our example, we wish to receive FreeBSD-CURRENT. We add this line at the beginning of our `supfile`:

  ```
  *default tag=.
  ```

  There is an important special case that comes into play if you specify neither a `tag=` field nor a `date=` field. In that case, you receive the actual RCS files directly from the server’s CVS repository, rather than receiving a particular version. Developers generally prefer this mode of operation. By maintaining a copy of the repository itself on their systems, they gain the ability to browse the revision histories and examine past versions of files. This gain is achieved at a large cost in terms of disk space, however.

- Where do you want to get them from?

  We use the `host=` field to tell `cvsup` where to obtain its updates. Any of the CVSup mirror sites will do, though you should try to select one that is close to you in cyberspace. In this example we will use a fictional FreeBSD distribution site, `cvsup99.FreeBSD.org`:

  ```
  *default host=cvsup99.FreeBSD.org
  ```

You will need to change the host to one that actually exists before running **CVSup**. On any particular run of `cvsup`, you can override the host setting on the command line, with `-h` *`hostname`*.

- Where do you want to put them on your own machine?

  The `prefix=` field tells `cvsup` where to put the files it receives. In this example, we will put the source files directly into our main source tree, `/usr/src`. The `src` directory is already implicit in the collections we have chosen to receive, so this is the correct specification:

  ```
  *default prefix=/usr
  ```

- Where should `cvsup` maintain its status files?

  The **CVSup** client maintains certain status files in what is called the "base" directory. These files help **CVSup** to work more efficiently, by keeping track of which updates you have already received. We will use the standard base directory, `/var/db`:

  ```
  *default base=/var/db
  ```

  If your base directory does not already exist, now would be a good time to create it. The `cvsup` client will refuse to run if the base directory does not exist.

- Miscellaneous `supfile` settings:

  There is one more line of boiler plate that normally needs to be present in the `supfile`:

  ```
  *default release=cvs delete use-rel-suffix compress
  ```

  `release=cvs` indicates that the server should get its information out of the main FreeBSD CVS repository. This is virtually always the case, but there are other possibilities which are beyond the scope of this discussion.

  `delete` gives **CVSup** permission to delete files. You should always specify this, so that **CVSup** can keep your source tree fully up-to-date. **CVSup** is careful to delete only those files for which it is responsible. Any extra files you happen to have will be left strictly alone.

  `use-rel-suffix` is ... arcane. If you really want to know about it, see the cvsup(1) manual page. Otherwise, just specify it and do not worry about it.

  `compress` enables the use of gzip-style compression on the communication channel. If your network link is T1 speed or faster, you probably should not use compression. Otherwise, it helps substantially.

- Putting it all together:

  Here is the entire `supfile` for our example:

  ```
  *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 The `refuse` File

As mentioned above, **CVSup** uses a *pull method*. Basically, this means that you connect to the **CVSup** server, and it says, "Here is what you can download from me...", and your client responds "OK, I will take this, this, this, and this." In the default configuration, the **CVSup** client will take every file associated with the collection and tag you chose in the configuration file. However, this is not always what you want, especially if you are synching the `doc`,

`ports`, or `www` trees — most people cannot read four or five languages, and therefore they do not need to download the language-specific files. If you are **CVSup**ing the Ports Collection, you can get around this by specifying each collection individually (e.g., *ports-astrology*, *ports-biology*, etc instead of simply saying *ports-all*). However, since the `doc` and `www` trees do not have language-specific collections, you must use one of **CVSup**'s many nifty features: the `refuse` file.

The `refuse` file essentially tells **CVSup** that it should not take every single file from a collection; in other words, it tells the client to *refuse* certain files from the server. The `refuse` file can be found (or, if you do not yet have one, should be placed) in *`base`*`/sup/`. *`base`* is defined in your `supfile`; our defined *`base`* is `/var/db`, which means that by default the `refuse` file is `/var/db/sup/refuse`.

The `refuse` file has a very simple format; it simply contains the names of files or directories that you do not wish to download. For example, if you cannot speak any languages other than English and some German, and you do not feel the need to read the German translation of documentation, you can put the following in your `refuse` file:

```
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_*
```

and so forth for the other languages (you can find the full list by browsing the FreeBSD CVS repository (http://www.FreeBSD.org/cgi/cvsweb.cgi/)).

With this very useful feature, those users who are on slow links or pay by the minute for their Internet connection will be able to save valuable time as they will no longer need to download files that they will never use. For more information on `refuse` files and other neat features of **CVSup**, please view its manual page.

A.6.4 Running CVSup

You are now ready to try an update. The command line for doing this is quite simple:

```
# cvsup supfile
```

where *`supfile`* is of course the name of the `supfile` you have just created. Assuming you are running under X11, `cvsup` will display a GUI window with some buttons to do the usual things. Press the go button, and watch it run.

Since you are updating your actual `/usr/src` tree in this example, you will need to run the program as `root` so that `cvsup` has the permissions it needs to update your files. Having just created your configuration file, and having never

used this program before, that might understandably make you nervous. There is an easy way to do a trial run without touching your precious files. Just create an empty directory somewhere convenient, and name it as an extra argument on the command line:

```
# mkdir /var/tmp/dest
# cvsup supfile /var/tmp/dest
```

The directory you specify will be used as the destination directory for all file updates. **CVSup** will examine your usual files in `/usr/src`, but it will not modify or delete any of them. Any file updates will instead land in `/var/tmp/dest/usr/src`. **CVSup** will also leave its base directory status files untouched when run this way. The new versions of those files will be written into the specified directory. As long as you have read access to `/usr/src`, you do not even need to be `root` to perform this kind of trial run.

If you are not running X11 or if you just do not like GUIs, you should add a couple of options to the command line when you run `cvsup`:

```
# cvsup -g -L 2 supfile
```

The `-g` tells **CVSup** not to use its GUI. This is automatic if you are not running X11, but otherwise you have to specify it.

The `-L 2` tells **CVSup** to print out the details of all the file updates it is doing. There are three levels of verbosity, from `-L 0` to `-L 2`. The default is 0, which means total silence except for error messages.

There are plenty of other options available. For a brief list of them, type `cvsup -H`. For more detailed descriptions, see the manual page.

Once you are satisfied with the way updates are working, you can arrange for regular runs of **CVSup** using cron(8). Obviously, you should not let **CVSup** use its GUI when running it from cron(8).

A.6.5 CVSup File Collections

The file collections available via **CVSup** are organized hierarchically. There are a few large collections, and they are divided into smaller sub-collections. Receiving a large collection is equivalent to receiving each of its sub-collections. The hierarchical relationships among collections are reflected by the use of indentation in the list below.

The most commonly used collections are `src-all`, and `ports-all`. The other collections are used only by small groups of people for specialized purposes, and some mirror sites may not carry all of them.

`cvs-all release=cvs`

The main FreeBSD CVS repository, including the cryptography code.

`distrib release=cvs`

Files related to the distribution and mirroring of FreeBSD.

`doc-all release=cvs`

Sources for the FreeBSD Handbook and other documentation. This does not include files for the FreeBSD web site.

`ports-all release=cvs`

The FreeBSD Ports Collection.

Important: If you do not want to update the whole of `ports-all` (the whole ports tree), but use one of the subcollections listed below, make sure that you *always* update the `ports-base` subcollection! Whenever something changes in the ports build infrastructure represented by `ports-base`, it is virtually certain that those changes will be used by "real" ports real soon. Thus, if you only update the "real" ports and they use some of the new features, there is a very high chance that their build will fail with some mysterious error message. The *very first* thing to do in this case is to make sure that your `ports-base` subcollection is up to date.

Important: If you are going to be building your own local copy of `ports/INDEX`, you *must* accept `ports-all` (the whole ports tree). Building `ports/INDEX` with a partial tree is not supported. See the FAQ (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/applications.html#MAKE-INDEX).

`ports-accessibility release=cvs`

Software to help disabled users.

`ports-arabic release=cvs`

Arabic language support.

`ports-archivers release=cvs`

Archiving tools.

`ports-astro release=cvs`

Astronomical ports.

`ports-audio release=cvs`

Sound support.

`ports-base release=cvs`

The Ports Collection build infrastructure - various files located in the `Mk/` and `Tools/` subdirectories of `/usr/ports`.

Note: Please see the important warning above: you should *always* update this subcollection, whenever you update any part of the FreeBSD Ports Collection!

`ports-benchmarks release=cvs`

Benchmarks.

`ports-biology release=cvs`

Biology.

`ports-cad release=cvs`

Computer aided design tools.

`ports-chinese release=cvs`

Chinese language support.

`ports-comms release=cvs`

Communication software.

`ports-converters release=cvs`

character code converters.

`ports-databases release=cvs`

Databases.

`ports-deskutils release=cvs`

Things that used to be on the desktop before computers were invented.

`ports-devel release=cvs`

Development utilities.

`ports-dns release=cvs`

DNS related software.

`ports-editors release=cvs`

Editors.

`ports-emulators release=cvs`

Emulators for other operating systems.

`ports-finance release=cvs`

Monetary, financial and related applications.

`ports-ftp release=cvs`

FTP client and server utilities.

`ports-games release=cvs`

Games.

`ports-german release=cvs`

German language support.

`ports-graphics release=cvs`

Graphics utilities.

`ports-hebrew release=cvs`

Hebrew language support.

`ports-hungarian release=cvs`

Hungarian language support.

`ports-irc release=cvs`

Internet Relay Chat utilities.

`ports-japanese release=cvs`

Japanese language support.

`ports-java release=cvs`

Java utilities.

`ports-korean release=cvs`

Korean language support.

`ports-lang release=cvs`

Programming languages.

`ports-mail release=cvs`

Mail software.

`ports-math release=cvs`

Numerical computation software.

`ports-mbone release=cvs`

MBone applications.

`ports-misc release=cvs`

Miscellaneous utilities.

`ports-multimedia release=cvs`

Multimedia software.

`ports-net release=cvs`

Networking software.

`ports-net-im release=cvs`

Instant messaging software.

`ports-net-mgmt release=cvs`

Network management software.

`ports-net-p2p release=cvs`

Peer to peer networking.

`ports-news release=cvs`

USENET news software.

`ports-palm release=cvs`

Software support for Palm™ series.

`ports-polish release=cvs`

Polish language support.

`ports-ports-mgmt release=cvs`

Utilities to manage ports and packages.

`ports-portuguese release=cvs`

Portuguese language support.

`ports-print release=cvs`

Printing software.

`ports-russian release=cvs`

Russian language support.

`ports-science release=cvs`

Science.

`ports-security release=cvs`

Security utilities.

`ports-shells release=cvs`

Command line shells.

`ports-sysutils release=cvs`

System utilities.

`ports-textproc release=cvs`

text processing utilities (does not include desktop publishing).

`ports-ukrainian release=cvs`

Ukrainian language support.

`ports-vietnamese release=cvs`

Vietnamese language support.

`ports-www release=cvs`

Software related to the World Wide Web.

`ports-x11 release=cvs`

Ports to support the X window system.

`ports-x11-clocks release=cvs`

X11 clocks.

`ports-x11-drivers release=cvs`

X11 drivers.

`ports-x11-fm release=cvs`

X11 file managers.

`ports-x11-fonts release=cvs`

X11 fonts and font utilities.

`ports-x11-toolkits release=cvs`

X11 toolkits.

`ports-x11-servers release=cvs`

X11 servers.

`ports-x11-themes release=cvs`

X11 themes.

`ports-x11-wm release=cvs`

X11 window managers.

`projects-all release=cvs`

Sources for the FreeBSD projects repository.

`src-all release=cvs`

The main FreeBSD sources, including the cryptography code.

`src-base release=cvs`

Miscellaneous files at the top of `/usr/src`.

`src-bin release=cvs`

User utilities that may be needed in single-user mode (`/usr/src/bin`).

`src-cddl release=cvs`

Utilities and libraries covered by the CDDL license (`/usr/src/cddl`).

`src-contrib release=cvs`

Utilities and libraries from outside the FreeBSD project, used relatively unmodified (`/usr/src/contrib`).

`src-crypto release=cvs`

Cryptography utilities and libraries from outside the FreeBSD project, used relatively unmodified (`/usr/src/crypto`).

`src-eBones release=cvs`

Kerberos and DES (`/usr/src/eBones`). Not used in current releases of FreeBSD.

`src-etc release=cvs`

System configuration files (`/usr/src/etc`).

`src-games release=cvs`

Games (`/usr/src/games`).

`src-gnu release=cvs`

Utilities covered by the GNU Public License (`/usr/src/gnu`).

`src-include release=cvs`

Header files (`/usr/src/include`).

`src-kerberos5 release=cvs`

Kerberos5 security package (`/usr/src/kerberos5`).

`src-kerberosIV release=cvs`

KerberosIV security package (`/usr/src/kerberosIV`).

`src-lib release=cvs`

Libraries (`/usr/src/lib`).

`src-libexec release=cvs`

System programs normally executed by other programs (`/usr/src/libexec`).

`src-release release=cvs`

Files required to produce a FreeBSD release (`/usr/src/release`).

`src-rescue release=cvs`

Statically linked programs for emergency recovery; see rescue(8) (`/usr/src/rescue`).

`src-sbin release=cvs`

System utilities for single-user mode (`/usr/src/sbin`).

`src-secure release=cvs`

Cryptographic libraries and commands (`/usr/src/secure`).

`src-share release=cvs`

Files that can be shared across multiple systems (`/usr/src/share`).

`src-sys release=cvs`

The kernel (`/usr/src/sys`).

`src-sys-crypto release=cvs`

Kernel cryptography code (`/usr/src/sys/crypto`).

`src-tools release=cvs`

Various tools for the maintenance of FreeBSD (`/usr/src/tools`).

`src-usrbin release=cvs`

User utilities (`/usr/src/usr.bin`).

`src-usrsbin release=cvs`

System utilities (`/usr/src/usr.sbin`).

`www release=cvs`

The sources for the FreeBSD WWW site.

`distrib release=self`

The **CVSup** server's own configuration files. Used by **CVSup** mirror sites.

`gnats release=current`

The GNATS bug-tracking database.

`mail-archive release=current`

FreeBSD mailing list archive.

`www release=current`

The pre-processed FreeBSD WWW site files (not the source files). Used by WWW mirror sites.

A.6.6 For More Information

For the **CVSup** FAQ and other information about **CVSup**, see The CVSup Home Page (http://www.cvsup.org).

Most FreeBSD-related discussion of **CVSup** takes place on the FreeBSD technical discussions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers). New versions of the software are announced there, as well as on the FreeBSD announcements mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce).

For questions or bug reports about **CVSup** take a look at the CVSup FAQ (http://www.cvsup.org/faq.html#bugreports).

A.6.7 CVSup Sites

CVSup servers for FreeBSD are running at the following sites:

Central Servers, Primary Mirror Sites, Argentina, Armenia, Australia, Austria, Brazil, Bulgaria, Canada, China, Costa Rica, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Indonesia, Ireland, Israel, Italy, Japan, Korea, Kuwait, Kyrgyzstan, Latvia, Lithuania, Netherlands, Norway, Philippines, Poland, Portugal, Romania, Russia, San Marino, Singapore, Slovak Republic, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Thailand, Turkey, Ukraine, United Kingdom, USA.

(as of 2010/11/13 13:50:55 UTC)

Central Servers

- cvsup.FreeBSD.org

Primary Mirror Sites

- cvsup1.FreeBSD.org
- cvsup2.FreeBSD.org
- cvsup3.FreeBSD.org
- cvsup4.FreeBSD.org
- cvsup5.FreeBSD.org
- cvsup6.FreeBSD.org
- cvsup7.FreeBSD.org
- cvsup8.FreeBSD.org
- cvsup9.FreeBSD.org
- cvsup10.FreeBSD.org
- cvsup11.FreeBSD.org
- cvsup12.FreeBSD.org
- cvsup13.FreeBSD.org

- cvsup14.FreeBSD.org
- cvsup15.FreeBSD.org
- cvsup16.FreeBSD.org
- cvsup18.FreeBSD.org

Argentina

- cvsup.ar.FreeBSD.org

Armenia

- cvsup1.am.FreeBSD.org

Australia

- cvsup.au.FreeBSD.org

Austria

- cvsup.at.FreeBSD.org
- cvsup2.at.FreeBSD.org

Brazil

- cvsup.br.FreeBSD.org
- cvsup2.br.FreeBSD.org
- cvsup3.br.FreeBSD.org
- cvsup4.br.FreeBSD.org
- cvsup5.br.FreeBSD.org

Bulgaria

- cvsup.bg.FreeBSD.org

Canada

- cvsup1.ca.FreeBSD.org

China

- cvsup.cn.FreeBSD.org
- cvsup2.cn.FreeBSD.org

Costa Rica

- cvsup1.cr.FreeBSD.org

Czech Republic

- cvsup.cz.FreeBSD.org

Denmark

- cvsup.dk.FreeBSD.org
- cvsup2.dk.FreeBSD.org

Estonia

- cvsup.ee.FreeBSD.org

Finland

- cvsup.fi.FreeBSD.org
- cvsup2.fi.FreeBSD.org

France

- cvsup.fr.FreeBSD.org
- cvsup1.fr.FreeBSD.org
- cvsup2.fr.FreeBSD.org
- cvsup3.fr.FreeBSD.org
- cvsup4.fr.FreeBSD.org
- cvsup5.fr.FreeBSD.org
- cvsup8.fr.FreeBSD.org

Germany

- cvsup.de.FreeBSD.org
- cvsup2.de.FreeBSD.org
- cvsup3.de.FreeBSD.org
- cvsup4.de.FreeBSD.org
- cvsup5.de.FreeBSD.org
- cvsup6.de.FreeBSD.org
- cvsup7.de.FreeBSD.org
- cvsup8.de.FreeBSD.org

Greece

- cvsup.gr.FreeBSD.org
- cvsup2.gr.FreeBSD.org

Hungary

- cvsup.hu.FreeBSD.org

Iceland

- cvsup.is.FreeBSD.org

Indonesia

- cvsup.id.FreeBSD.org

Ireland

- cvsup.ie.FreeBSD.org
- cvsup2.ie.FreeBSD.org

Israel

- cvsup.il.FreeBSD.org

Italy

- cvsup.it.FreeBSD.org

Japan

- 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
- cvsup2.kr.FreeBSD.org
- cvsup3.kr.FreeBSD.org

Kuwait

- cvsup1.kw.FreeBSD.org

Kyrgyzstan

- cvsup.kg.FreeBSD.org

Latvia

- cvsup.lv.FreeBSD.org
- cvsup2.lv.FreeBSD.org

Lithuania

- cvsup.lt.FreeBSD.org
- cvsup2.lt.FreeBSD.org
- cvsup3.lt.FreeBSD.org

Netherlands

- cvsup.nl.FreeBSD.org
- cvsup2.nl.FreeBSD.org
- cvsup3.nl.FreeBSD.org

Norway

- cvsup.no.FreeBSD.org

Philippines

- cvsup1.ph.FreeBSD.org

Poland

- cvsup.pl.FreeBSD.org
- cvsup2.pl.FreeBSD.org
- cvsup3.pl.FreeBSD.org

Portugal

- cvsup.pt.FreeBSD.org
- cvsup2.pt.FreeBSD.org
- cvsup3.pt.FreeBSD.org

Romania

- cvsup.ro.FreeBSD.org
- cvsup1.ro.FreeBSD.org
- cvsup2.ro.FreeBSD.org
- cvsup3.ro.FreeBSD.org

Russia

- cvsup.ru.FreeBSD.org
- cvsup2.ru.FreeBSD.org
- cvsup3.ru.FreeBSD.org
- cvsup4.ru.FreeBSD.org
- cvsup5.ru.FreeBSD.org
- cvsup6.ru.FreeBSD.org
- cvsup7.ru.FreeBSD.org

San Marino

- cvsup.sm.FreeBSD.org

Singapore

- cvsup.sg.FreeBSD.org

Slovak Republic

- cvsup.sk.FreeBSD.org

Slovenia

- cvsup.si.FreeBSD.org
- cvsup2.si.FreeBSD.org

South Africa

- cvsup.za.FreeBSD.org
- cvsup2.za.FreeBSD.org

Spain

- cvsup.es.FreeBSD.org
- cvsup2.es.FreeBSD.org
- cvsup3.es.FreeBSD.org

Sweden

- cvsup.se.FreeBSD.org
- cvsup2.se.FreeBSD.org

Switzerland

- cvsup.ch.FreeBSD.org

Taiwan

- cvsup.tw.FreeBSD.org
- cvsup3.tw.FreeBSD.org
- cvsup4.tw.FreeBSD.org
- cvsup5.tw.FreeBSD.org
- cvsup6.tw.FreeBSD.org
- cvsup7.tw.FreeBSD.org
- cvsup8.tw.FreeBSD.org
- cvsup9.tw.FreeBSD.org
- cvsup10.tw.FreeBSD.org
- cvsup11.tw.FreeBSD.org
- cvsup12.tw.FreeBSD.org
- cvsup13.tw.FreeBSD.org
- cvsup14.tw.FreeBSD.org

Thailand

- cvsup.th.FreeBSD.org

Turkey

- cvsup.tr.FreeBSD.org
- cvsup2.tr.FreeBSD.org

Ukraine

- cvsup2.ua.FreeBSD.org
- cvsup3.ua.FreeBSD.org
- cvsup5.ua.FreeBSD.org
- cvsup6.ua.FreeBSD.org
- cvsup7.ua.FreeBSD.org

United Kingdom

- cvsup.uk.FreeBSD.org
- cvsup2.uk.FreeBSD.org
- cvsup3.uk.FreeBSD.org
- cvsup4.uk.FreeBSD.org

USA

- cvsup1.us.FreeBSD.org
- cvsup2.us.FreeBSD.org
- cvsup3.us.FreeBSD.org
- cvsup4.us.FreeBSD.org
- cvsup5.us.FreeBSD.org
- cvsup6.us.FreeBSD.org
- cvsup7.us.FreeBSD.org
- cvsup8.us.FreeBSD.org
- cvsup9.us.FreeBSD.org
- cvsup10.us.FreeBSD.org
- cvsup11.us.FreeBSD.org
- cvsup12.us.FreeBSD.org
- cvsup13.us.FreeBSD.org
- cvsup14.us.FreeBSD.org
- cvsup15.us.FreeBSD.org
- cvsup16.us.FreeBSD.org
- cvsup18.us.FreeBSD.org

A.7 CVS Tags

When obtaining or updating sources using **cvs** or **CVSup**, a revision tag must be specified. A revision tag refers to either a particular line of FreeBSD development, or a specific point in time. The first type are called “branch tags”, and the second type are called “release tags”.

A.7.1 Branch Tags

All of these, with the exception of `HEAD` (which is always a valid tag), only apply to the `src/` tree. The `ports/`, `doc/`, and `www/` trees are not branched.

HEAD

Symbolic name for the main line, or FreeBSD-CURRENT. Also the default when no revision is specified.

In **CVSup**, this tag is represented by a `.` (not punctuation, but a literal `.` character).

> **Note:** In CVS, this is the default when no revision tag is specified. It is usually *not* a good idea to checkout or update to CURRENT sources on a STABLE machine, unless that is your intent.

RELENG_8

The line of development for FreeBSD-8.X, also known as FreeBSD 8-STABLE

RELENG_8_2

The release branch for FreeBSD-8.2, used only for security advisories and other critical fixes.

RELENG_8_1

The release branch for FreeBSD-8.1, used only for security advisories and other critical fixes.

RELENG_8_0

The release branch for FreeBSD-8.0, used only for security advisories and other critical fixes.

RELENG_7

The line of development for FreeBSD-7.X, also known as FreeBSD 7-STABLE

RELENG_7_4

The release branch for FreeBSD-7.4, used only for security advisories and other critical fixes.

RELENG_7_3

The release branch for FreeBSD-7.3, used only for security advisories and other critical fixes.

RELENG_7_2

The release branch for FreeBSD-7.2, used only for security advisories and other critical fixes.

RELENG_7_1

The release branch for FreeBSD-7.1, used only for security advisories and other critical fixes.

RELENG_7_0

The release branch for FreeBSD-7.0, used only for security advisories and other critical fixes.

RELENG_6

The line of development for FreeBSD-6.X, also known as FreeBSD 6-STABLE

RELENG_6_4

The release branch for FreeBSD-6.4, used only for security advisories and other critical fixes.

RELENG_6_3

The release branch for FreeBSD-6.3, used only for security advisories and other critical fixes.

RELENG_6_2

The release branch for FreeBSD-6.2, used only for security advisories and other critical fixes.

RELENG_6_1

The release branch for FreeBSD-6.1, used only for security advisories and other critical fixes.

RELENG_6_0

The release branch for FreeBSD-6.0, used only for security advisories and other critical fixes.

RELENG_5

The line of development for FreeBSD-5.X, also known as FreeBSD 5-STABLE.

RELENG_5_5

The release branch for FreeBSD-5.5, used only for security advisories and other critical fixes.

RELENG_5_4

The release branch for FreeBSD-5.4, used only for security advisories and other critical fixes.

RELENG_5_3

The release branch for FreeBSD-5.3, used only for security advisories and other critical fixes.

RELENG_5_2

The release branch for FreeBSD-5.2 and FreeBSD-5.2.1, used only for security advisories and other critical fixes.

RELENG_5_1

The release branch for FreeBSD-5.1, used only for security advisories and other critical fixes.

RELENG_5_0

The release branch for FreeBSD-5.0, used only for security advisories and other critical fixes.

RELENG_4

The line of development for FreeBSD-4.X, also known as FreeBSD 4-STABLE.

RELENG_4_11

The release branch for FreeBSD-4.11, used only for security advisories and other critical fixes.

RELENG_4_10

The release branch for FreeBSD-4.10, used only for security advisories and other critical fixes.

RELENG_4_9

The release branch for FreeBSD-4.9, used only for security advisories and other critical fixes.

RELENG_4_8

The release branch for FreeBSD-4.8, used only for security advisories and other critical fixes.

RELENG_4_7

The release branch for FreeBSD-4.7, used only for security advisories and other critical fixes.

RELENG_4_6

The release branch for FreeBSD-4.6 and FreeBSD-4.6.2, used only for security advisories and other critical fixes.

RELENG_4_5

The release branch for FreeBSD-4.5, used only for security advisories and other critical fixes.

RELENG_4_4

The release branch for FreeBSD-4.4, used only for security advisories and other critical fixes.

RELENG_4_3

The release branch for FreeBSD-4.3, used only for security advisories and other critical fixes.

RELENG_3

The line of development for FreeBSD-3.X, also known as 3.X-STABLE.

RELENG_2_2

The line of development for FreeBSD-2.2.X, also known as 2.2-STABLE. This branch is mostly obsolete.

A.7.2 Release Tags

These tags refer to a specific point in time when a particular version of FreeBSD was released. The release engineering process is documented in more detail by the Release Engineering Information (http://www.FreeBSD.org/releng/) and Release Process (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/releng/release-proc.html) documents. The `src` tree uses tag names that start with `RELENG_` tags. The `ports` and `doc` trees use tags whose names begin with `RELEASE` tags. Finally, the `www` tree is not tagged with any special name for releases.

RELENG_8_2_0_RELEASE

FreeBSD 8.2

RELENG_8_1_0_RELEASE

FreeBSD 8.1

RELENG_8_0_0_RELEASE
: FreeBSD 8.0

RELENG_7_4_0_RELEASE
: FreeBSD 7.4

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 Sites

AFS servers for FreeBSD are running at the following sites:

Sweden

The path to the files are: `/afs/stacken.kth.se/ftp/pub/FreeBSD/`

```
stacken.kth.se          # Stacken Computer Club, KTH, Sweden
130.237.234.43          #hot.stacken.kth.se
130.237.237.230         #fishburger.stacken.kth.se
130.237.234.3           #milko.stacken.kth.se
```

Maintainer <`ftp@stacken.kth.se`>

A.9 rsync Sites

The following sites make FreeBSD available through the rsync protocol. The **rsync** utility works in much the same way as the rcp(1) command, but has more options and uses the rsync remote-update protocol which transfers only the differences between two sets of files, thus greatly speeding up the synchronization over the network. This is most useful if you are a mirror site for the FreeBSD FTP server, or the CVS repository. The **rsync** suite is available for many operating systems, on FreeBSD, see the `net/rsync` port or use the package.

Czech Republic

rsync://ftp.cz.FreeBSD.org/

Available collections:

- ftp: A partial mirror of the FreeBSD FTP server.
- FreeBSD: A full mirror of the FreeBSD FTP server.

Netherlands

rsync://ftp.nl.FreeBSD.org/

Available collections:

- FreeBSD: A full mirror of the FreeBSD FTP server.

Russia

rsync://ftp.mtu.ru/

Available collections:

- FreeBSD: A full mirror of the FreeBSD FTP server.
- FreeBSD-gnats: The GNATS bug-tracking database.
- FreeBSD-Archive: The mirror of FreeBSD Archive FTP server.

Taiwan

rsync://ftp.tw.FreeBSD.org/

rsync://ftp2.tw.FreeBSD.org/

rsync://ftp6.tw.FreeBSD.org/

Available collections:

- FreeBSD: A full mirror of the FreeBSD FTP server.

United Kingdom

rsync://rsync.mirrorservice.org/

Available collections:

- sites/ftp.freebsd.org: A full mirror of the FreeBSD FTP server.

United States of America

rsync://ftp-master.FreeBSD.org/

This server may only be used by FreeBSD primary mirror sites.

Available collections:

- FreeBSD: The master archive of the FreeBSD FTP server.
- acl: The FreeBSD master ACL list.

rsync://ftp13.FreeBSD.org/

Available collections:

- FreeBSD: A full mirror of the FreeBSD FTP server.

Appendix B.
Bibliography

While the manual pages provide the definitive reference for individual pieces of the FreeBSD operating system, they are notorious for not illustrating how to put the pieces together to make the whole operating system run smoothly. For this, there is no substitute for a good book on UNIX system administration and a good users' manual.

B.1 Books & Magazines Specific to FreeBSD

International books & Magazines:

- Using FreeBSD (http://jdli.tw.FreeBSD.org/publication/book/freebsd2/index.htm) (in Traditional Chinese), published by Drmaster (http://www.drmaster.com.tw/), 1997. ISBN 9-578-39435-7.
- FreeBSD Unleashed (Simplified Chinese translation), published by China Machine Press (http://www.hzbook.com/). ISBN 7-111-10201-0.
- FreeBSD From Scratch First Edition (in Simplified Chinese), published by China Machine Press. ISBN 7-111-07482-3.
- FreeBSD From Scratch Second Edition (in Simplified Chinese), published by China Machine Press. ISBN 7-111-10286-X.
- FreeBSD Handbook Second Edition (Simplified Chinese translation), published by Posts & Telecom Press (http://www.ptpress.com.cn/). ISBN 7-115-10541-3.
- FreeBSD 3.x Internet (in Simplified Chinese), published by Tsinghua University Press (http://www.tup.tsinghua.edu.cn/). ISBN 7-900625-66-6.
- FreeBSD & Windows (in Simplified Chinese), published by China Railway Publishing House (http://www.tdpress.com/). ISBN 7-113-03845-X
- FreeBSD Internet Services HOWTO (in Simplified Chinese), published by China Railway Publishing House. ISBN 7-113-03423-3
- FreeBSD for PC 98'ers (in Japanese), published by SHUWA System Co, LTD. ISBN 4-87966-468-5 C3055 P2900E.
- FreeBSD (in Japanese), published by CUTT. ISBN 4-906391-22-2 C3055 P2400E.
- Complete Introduction to FreeBSD (http://www.shoeisha.com/book/Detail.asp?bid=650) (in Japanese), published by 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) (in Japanese), published by ASCII (http://www.ascii.co.jp/). ISBN 4-7561-1733-3 P3000E.
- FreeBSD Handbook (Japanese translation), published by ASCII (http://www.ascii.co.jp/). ISBN 4-7561-1580-2 P3800E.

- FreeBSD mit Methode (in German), published by 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) (in German), published by 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) (in German), published by 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/) (in German), published by Verlag Modere 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) (in Japanese), published by 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/)* (in Indonesia Language), published by Elex Media Komputindo (http://www.elexmedia.co.id/).
- Absolute BSD: The Ultimate Guide to FreeBSD (Traditional Chinese translation), published by GrandTech Press (http://www.grandtech.com.tw/), 2003. ISBN 986-7944-92-5.
- The FreeBSD 6.0 Book (http://www.twbsd.org/cht/book/) (in Traditional Chinese), published by Drmaster, 2006. ISBN 9-575-27878-X.

English language books & Magazines:

- Absolute FreeBSD, 2nd Edition: The Complete Guide to FreeBSD (http://www.absoluteFreeBSD.com/), published by No Starch Press (http://www.nostarch.com/), 2007. ISBN: 978-1-59327-151-0
- The Complete FreeBSD (http://www.freebsdmall.com/cgi-bin/fm/bsdcomp), published by O'Reilly (http://www.oreilly.com/), 2003. ISBN: 0596005164
- The FreeBSD Corporate Networker's Guide (http://www.freebsd-corp-net-guide.com/), published by Addison-Wesley (http://www.awl.com/aw/), 2000. ISBN: 0201704811
- FreeBSD: An Open-Source Operating System for Your Personal Computer (http://andrsn.stanford.edu/FreeBSD/introbook/), published by The Bit Tree Press, 2001. ISBN: 0971204500
- Teach Yourself FreeBSD in 24 Hours, published by Sams (http://www.samspublishing.com/), 2002. ISBN: 0672324245
- FreeBSD 6 Unleashed, published by Sams (http://www.samspublishing.com/), 2006. ISBN: 0672328755
- FreeBSD: The Complete Reference, published by McGrawHill (http://books.mcgraw-hill.com), 2003. ISBN: 0072224096
- BSD Magazine (http://www.bsdmag.org), published by Software Press Sp. z o.o. SK. ISSN 1898-9144

B.2 Users' Guides

- 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 State University (http://www.osu.edu/) has written a UNIX Introductory Course (http://8help.osu.edu/wks/unix_course/index.html) which is available online in HTML and PostScript format.

  An Italian translation (http://www.FreeBSD.org/doc/it_IT.ISO8859-15/books/unix-introduction/index.html) of this document is available as part of the FreeBSD Italian Documentation Project.
- Jpman Project, Japan FreeBSD Users Group (http://www.jp.FreeBSD.org/). FreeBSD User's Reference Manual (http://www.pc.mycom.co.jp/FreeBSD/urm.html) (Japanese translation). Mainichi Communications Inc. (http://www.pc.mycom.co.jp/), 1998. ISBN4-8399-0088-4 P3800E.
- Edinburgh University (http://www.ed.ac.uk/) has written an Online Guide (http://unixhelp.ed.ac.uk/) for newcomers to the UNIX environment.

B.3 Administrators' Guides

- Albitz, Paul and Liu, Cricket. *DNS and BIND*, 4th Ed. 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, et al. *Sendmail*, 2nd Ed. O'Reilly & Associates, Inc., 1997. ISBN 1-56592-222-0
- Frisch, Æleen. *Essential System Administration*, 2nd Ed. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-127-5
- Hunt, Craig. *TCP/IP Network Administration*, 2nd Ed. O'Reilly & Associates, Inc., 1997. ISBN 1-56592-322-7
- Nemeth, Evi. *UNIX System Administration Handbook*. 3rd Ed. 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) (Japanese translation). 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/) 2nd Ed. (in French), Eyrolles, 2004. ISBN 2-212-11463-X

B.4 Programmers' Guides

- 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*. 4th ed. Prentice Hall, 1995. ISBN 0-13-326224-3
- Kernighan, Brian and Dennis M. Ritchie. *The C Programming Language*. 2nd Ed. 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*. 2nd Ed. Reading, Mass. : Addison-Wesley, 2005. ISBN 0-201-43307-9
- Stevens, W. Richard. *UNIX Network Programming*. 2nd Ed, PTR Prentice Hall, 1998. ISBN 0-13-490012-X
- Wells, Bill. "Writing Serial Drivers for UNIX". *Dr. Dobb's Journal*. 19(15), December 1994. pp68-71, 97-99.

B.5 Operating System Internals

- Andleigh, Prabhat K. *UNIX System Architecture*. Prentice-Hall, Inc., 1990. ISBN 0-13-949843-5
- Jolitz, William. "Porting UNIX to the 386". *Dr. Dobb's Journal*. January 1991-July 1992.
- Leffler, Samuel J., Marshall Kirk McKusick, Michael J Karels and 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, and John Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-54979-4

  (Chapter 2 of this book is available online (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/design-44bsd/book.html) as part of the FreeBSD Documentation Project, and chapter 9 here (http://www.netapp.com/tech_library/nfsbook.html).)
- 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, Volume 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, Volume 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. and W. Richard Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63354-X

B.6 Security Reference

- Cheswick, William R. and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63357-4
- Garfinkel, Simson and Gene Spafford. *Practical UNIX & Internet Security*. 2nd Ed. 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 Hardware Reference

- Anderson, Don and Tom Shanley. *Pentium Processor System Architecture*. 2nd Ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40992-5
- Ferraro, Richard F. *Programmer's Guide to the EGA, VGA, and Super VGA Cards*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-62490-7
- Intel Corporation publishes documentation on their CPUs, chipsets and standards on their developer web site (http://developer.intel.com/), usually as PDF files.
- Shanley, Tom. *80486 System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40994-1
- Shanley, Tom. *ISA System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40996-8
- Shanley, Tom. *PCI System Architecture*. 4th ed. Reading, Mass. : Addison-Wesley, 1999. ISBN 0-201-30974-2
- Van Gilluwe, Frank. *The Undocumented PC,* 2nd Ed. Reading, Mass: Addison-Wesley Pub. Co., 1996. ISBN 0-201-47950-8
- Messmer, Hans-Peter. *The Indispensable PC Hardware Book*, 4th Ed. Reading, Mass: Addison-Wesley Pub. Co., 2002. ISBN 0-201-59616-4

B.8 UNIX History

- Lion, John *Lion's Commentary on UNIX, 6th Ed. With Source Code*. ITP Media Group, 1996. ISBN 1573980137
- Raymond, Eric S. *The New Hacker's Dictionary, 3rd edition*. MIT Press, 1996. ISBN 0-262-68092-0. Also known as the Jargon File (http://www.catb.org/~esr/jargon/html/index.html)
- 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. Out of print, but available online (http://www.simson.net/ref/ugh.pdf).
- Don Libes, Sandy Ressler *Life with UNIX* — special edition. 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 or `/usr/share/misc/bsd-family-tree` on a FreeBSD machine.
- *Networked Computer Science Technical Reports Library*. http://www.ncstrl.org/
- *Old BSD releases from the Computer Systems Research group (CSRG)*. http://www.mckusick.com/csrg/: The 4CD set covers all BSD versions from 1BSD to 4.4BSD and 4.4BSD-Lite2 (but not 2.11BSD, unfortunately). The last disk also holds the final sources plus the SCCS files.

B.9 Magazines and Journals

- *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* (in German) Computer- und Literaturverlag GmbH, ISSN 1436-7033

Appendix C.
Resources on the Internet

The rapid pace of FreeBSD progress makes print media impractical as a means of following the latest developments. Electronic resources are the best, if not often the only, way to stay informed of the latest advances. Since FreeBSD is a volunteer effort, the user community itself also generally serves as a "technical support department" of sorts, with electronic mail, web forums, and USENET news being the most effective way of reaching that community.

The most important points of contact with the FreeBSD user community are outlined below. If you are aware of other resources not mentioned here, please send them to the FreeBSD documentation project mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc) so that they may also be included.

C.1 Mailing Lists

The mailing lists are the most direct way of addressing questions or opening a technical discussion to a concentrated FreeBSD audience. There are a wide variety of lists on a number of different FreeBSD topics. Addressing your questions to the most appropriate mailing list will invariably assure a faster and more accurate response.

The charters for the various lists are given at the bottom of this document. *Please read the charter before joining or sending mail to any list*. Most of our list subscribers now receive many hundreds of FreeBSD related messages every day, and by setting down charters and rules for proper use we are striving to keep the signal-to-noise ratio of the lists high. To do less would see the mailing lists ultimately fail as an effective communications medium for the project.

> **Note:** *If you wish to test your ability to send to FreeBSD lists, send a test message to freebsd-test (http://lists.FreeBSD.org/mailman/listinfo/freebsd-test).* Please do not send test messages to any other list.

When in doubt about what list to post a question to, see How to get best results from the FreeBSD-questions mailing list (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/freebsd-questions).

Before posting to any list, please learn about how to best use the mailing lists, such as how to help avoid frequently-repeated discussions, by reading the Mailing List Frequently Asked Questions (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/mailing-list-faq) (FAQ) document.

Archives are kept for all of the mailing lists and can be searched using the FreeBSD World Wide Web server (http://www.FreeBSD.org/search/index.html). The keyword searchable archive offers an excellent way of finding answers to frequently asked questions and should be consulted before posting a question. Note that this also means that messages sent to FreeBSD mailing lists are archived in perpetuity. When protecting privacy is a concern, consider using a disposable secondary email address and posting only public information.

C.1.1 List Summary

General lists: The following are general lists which anyone is free (and encouraged) to join:

List	Purpose
freebsd-advocacy (http://lists.FreeBSD.org/mailman/listinfo/freebsd-advocacy)	FreeBSD Evangelism
freebsd-announce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce)	Important events and project milestones
freebsd-arch (http://lists.FreeBSD.org/mailman/listinfo/freebsd-arch)	Architecture and design discussions
freebsd-bugbusters (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugbusters)	Discussions pertaining to the maintenance of the FreeBSD problem report database and related tools
freebsd-bugs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugs)	Bug reports
freebsd-chat (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat)	Non-technical items related to the FreeBSD community
freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current)	Discussion concerning the use of FreeBSD-CURRENT
freebsd-isp (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp)	Issues for Internet Service Providers using FreeBSD
freebsd-jobs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-jobs)	FreeBSD employment and consulting opportunities
freebsd-policy (http://lists.FreeBSD.org/mailman/listinfo/freebsd-policy)	FreeBSD Core team policy decisions. Low volume, and read-only
freebsd-questions (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions)	User questions and technical support
freebsd-security-notifications (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications)	Security notifications
freebsd-stable (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable)	Discussion concerning the use of FreeBSD-STABLE
freebsd-test (http://lists.FreeBSD.org/mailman/listinfo/freebsd-test)	Where to send your test messages instead of one of the actual lists

Technical lists: The following lists are for technical discussion. You should read the charter for each list carefully before joining or sending mail to one as there are firm guidelines for their use and content.

List	Purpose
freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi)	ACPI and power management development
freebsd-afs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-afs)	Porting AFS to FreeBSD
freebsd-aic7xxx (http://lists.FreeBSD.org/mailman/listinfo/aic7xxx)	Developing drivers for the Adaptec AIC 7xxx
freebsd-alpha (http://lists.FreeBSD.org/mailman/listinfo/freebsd-alpha)	Porting FreeBSD to the Alpha
freebsd-amd64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-amd64)	Porting FreeBSD to AMD64 systems
freebsd-apache (http://lists.FreeBSD.org/mailman/listinfo/freebsd-apache)	Discussion about **Apache** related ports
freebsd-arm (http://lists.FreeBSD.org/mailman/listinfo/freebsd-arm)	Porting FreeBSD to ARM® processors
freebsd-atm (http://lists.FreeBSD.org/mailman/listinfo/freebsd-atm)	Using ATM networking with FreeBSD
freebsd-audit (http://lists.FreeBSD.org/mailman/listinfo/freebsd-audit)	Source code audit project
freebsd-binup (http://lists.FreeBSD.org/mailman/listinfo/freebsd-binup)	Design and development of the binary update system
freebsd-bluetooth (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bluetooth)	Using Bluetooth technology in FreeBSD
freebsd-cluster (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cluster)	Using FreeBSD in a clustered environment
freebsd-cvsweb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cvsweb)	CVSweb maintenance

List	Purpose
freebsd-database (http://lists.FreeBSD.org/mailman/listinfo/freebsd-database)	Discussing database use and development under FreeBSD
freebsd-doc (http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc)	Creating FreeBSD related documents
freebsd-drivers (http://lists.FreeBSD.org/mailman/listinfo/freebsd-drivers)	Writing device drivers for FreeBSD
freebsd-eclipse (http://lists.FreeBSD.org/mailman/listinfo/freebsd-eclipse)	FreeBSD users of Eclipse IDE, tools, rich client applications and ports.
freebsd-embedded (http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded)	Using FreeBSD in embedded applications
freebsd-eol (http://lists.FreeBSD.org/mailman/listinfo/freebsd-eol)	Peer support of FreeBSD-related software that is no longer supported by the FreeBSD project.
freebsd-emulation (http://lists.FreeBSD.org/mailman/listinfo/freebsd-emulation)	Emulation of other systems such as Linux/MS-DOS/Windows
freebsd-firewire (http://lists.FreeBSD.org/mailman/listinfo/freebsd-firewire)	FreeBSD FireWire® (iLink, IEEE 1394) technical discussion
freebsd-fs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-fs)	File systems
freebsd-gecko (http://lists.FreeBSD.org/mailman/listinfo/freebsd-gecko)	**Gecko Rendering Engine** issues
freebsd-geom (http://lists.FreeBSD.org/mailman/listinfo/freebsd-geom)	GEOM-specific discussions and implementations
freebsd-gnome (http://lists.FreeBSD.org/mailman/listinfo/freebsd-gnome)	Porting **GNOME** and **GNOME** applications

List	Purpose
freebsd-hackers (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers)	General technical discussion
freebsd-hardware (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hardware)	General discussion of hardware for running FreeBSD
freebsd-i18n (http://lists.FreeBSD.org/mailman/listinfo/freebsd-i18n)	FreeBSD Internationalization
freebsd-ia32 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia32)	FreeBSD on the IA-32 (Intel x86) platform
freebsd-ia64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia64)	Porting FreeBSD to Intel's upcoming IA64 systems
freebsd-ipfw (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ipfw)	Technical discussion concerning the redesign of the IP firewall code
freebsd-isdn (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn)	ISDN developers
freebsd-jail (http://lists.FreeBSD.org/mailman/listinfo/freebsd-jail)	Discussion about the jail(8) facility
freebsd-java (http://lists.FreeBSD.org/mailman/listinfo/freebsd-java)	Java developers and people porting JDKs to FreeBSD
freebsd-kde (https://mail.kde.org/mailman/listinfo/kde-freebsd)	Porting **KDE** and **KDE** applications
freebsd-lfs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-lfs)	Porting LFS to FreeBSD
freebsd-libh (http://lists.FreeBSD.org/mailman/listinfo/freebsd-libh)	The second generation installation and package system
freebsd-mips (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mips)	Porting FreeBSD to MIPS®
freebsd-mobile (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mobile)	Discussions about mobile computing
freebsd-mono (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mono)	Mono and C# applications on FreeBSD
freebsd-mozilla (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mozilla)	Porting **Mozilla** to FreeBSD

List	Purpose
freebsd-multimedia (http://lists.FreeBSD.org/mailman/listinfo/freebsd-multimedia)	Multimedia applications
freebsd-new-bus (http://lists.FreeBSD.org/mailman/listinfo/freebsd-new-bus)	Technical discussions about bus architecture
freebsd-net (http://lists.FreeBSD.org/mailman/listinfo/freebsd-net)	Networking discussion and TCP/IP source code
freebsd-openoffice (http://lists.FreeBSD.org/mailman/listinfo/freebsd-openoffice)	Porting **OpenOffice.org** and **StarOffice** to FreeBSD
freebsd-performance (http://lists.FreeBSD.org/mailman/listinfo/freebsd-performance)	Performance tuning questions for high performance/load installations
freebsd-perl (http://lists.FreeBSD.org/mailman/listinfo/freebsd-perl)	Maintenance of a number of Perl-related ports
freebsd-pf (http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf)	Discussion and questions about the packet filter firewall system
freebsd-platforms (http://lists.FreeBSD.org/mailman/listinfo/freebsd-platforms)	Concerning ports to non Intel architecture platforms
freebsd-ports (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports)	Discussion of the Ports Collection
freebsd-ports-bugs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs)	Discussion of the ports bugs/PRs
freebsd-ppc (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ppc)	Porting FreeBSD to the PowerPC®
freebsd-proliant (http://lists.FreeBSD.org/mailman/listinfo/freebsd-proliant)	Technical discussion of FreeBSD on HP ProLiant server platforms
freebsd-python (http://lists.FreeBSD.org/mailman/listinfo/freebsd-python)	FreeBSD-specific Python issues
freebsd-qa (http://lists.FreeBSD.org/mailman/listinfo/freebsd-qa)	Discussion of Quality Assurance, usually pending a release

List	Purpose
freebsd-rc (http://lists.FreeBSD.org/mailman/listinfo/freebsd-rc)	Discussion related to the `rc.d` system and its development
freebsd-realtime (http://lists.FreeBSD.org/mailman/listinfo/freebsd-realtime)	Development of realtime extensions to FreeBSD
freebsd-ruby (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ruby)	FreeBSD-specific Ruby discussions
freebsd-scsi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-scsi)	The SCSI subsystem
freebsd-security (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security)	Security issues affecting FreeBSD
freebsd-small (http://lists.FreeBSD.org/mailman/listinfo/freebsd-small)	Using FreeBSD in embedded applications (obsolete; use freebsd-embedded (http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded) instead)
freebsd-smp (http://lists.FreeBSD.org/mailman/listinfo/freebsd-smp)	Design discussions for [A]Symmetric MultiProcessing
freebsd-sparc64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-sparc64)	Porting FreeBSD to SPARC® based systems
freebsd-standards (http://lists.FreeBSD.org/mailman/listinfo/freebsd-standards)	FreeBSD's conformance to the C99 and the POSIX standards
freebsd-sun4v (http://lists.FreeBSD.org/mailman/listinfo/freebsd-sun4v)	Porting FreeBSD to UltraSPARC T1 based systems
freebsd-sysinstall (http://lists.FreeBSD.org/mailman/listinfo/freebsd-sysinstall)	sysinstall(8) development
freebsd-threads (http://lists.FreeBSD.org/mailman/listinfo/freebsd-threads)	Threading in FreeBSD
freebsd-testing (http://lists.FreeBSD.org/mailman/listinfo/freebsd-testing)	FreeBSD Performance and Stability Tests

List	Purpose
freebsd-tilera (http://lists.FreeBSD.org/mailman/listinfo/freebsd-tilera)	Porting FreeBSD to the Tilera family of CPUs
freebsd-tokenring (http://lists.FreeBSD.org/mailman/listinfo/freebsd-tokenring)	Support Token Ring in FreeBSD
freebsd-toolchain (http://lists.FreeBSD.org/mailman/listinfo/freebsd-toolchain)	Maintenance of FreeBSD's integrated toolchain
freebsd-usb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-usb)	Discussing FreeBSD support for USB
freebsd-virtualization (http://lists.FreeBSD.org/mailman/listinfo/freebsd-virtualization)	Discussion of various virtualization techniques supported by FreeBSD
freebsd-vuxml (http://lists.FreeBSD.org/mailman/listinfo/freebsd-vuxml)	Discussion on VuXML infrastructure
freebsd-x11 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-x11)	Maintenance and support of X11 on FreeBSD
freebsd-xen (http://lists.FreeBSD.org/mailman/listinfo/freebsd-xen)	Discussion of the FreeBSD port to Xen™ — implementation and usage

Limited lists: The following lists are for more specialized (and demanding) audiences and are probably not of interest to the general public. It is also a good idea to establish a presence in the technical lists before joining one of these limited lists so that you will understand the communications etiquette involved.

List	Purpose
freebsd-hubs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hubs)	People running mirror sites (infrastructural support)
freebsd-user-groups (http://lists.FreeBSD.org/mailman/listinfo/freebsd-user-groups)	User group coordination
freebsd-vendors (http://lists.FreeBSD.org/mailman/listinfo/freebsd-vendors)	Vendors pre-release coordination
freebsd-wip-status (http://lists.FreeBSD.org/mailman/listinfo/freebsd-wip-status)	FreeBSD Work-In-Progress Status

List	Purpose
freebsd-www (http://lists.FreeBSD.org/mailman/listinfo/freebsd-www)	Maintainers of www.FreeBSD.org (http://www.FreeBSD.org/index.html)

Digest lists: All of the above lists are available in a digest format. Once subscribed to a list, you can change your digest options in your account options section.

CVS & SVN lists: The following lists are for people interested in seeing the log messages for changes to various areas of the source tree. They are *Read-Only* lists and should not have mail sent to them.

List	Source area	Area Description (source for)
cvs-all (http://lists.FreeBSD.org/mailman/listinfo/cvs-all)	`/usr/(CVSROOT|doc|ports)`	All changes to any place in the tree (superset of other CVS commit lists)
cvs-doc (http://lists.FreeBSD.org/mailman/listinfo/cvs-doc)	`/usr/(doc|www)`	All changes to the doc and www trees
cvs-ports (http://lists.FreeBSD.org/mailman/listinfo/cvs-ports)	`/usr/ports`	All changes to the ports tree
cvs-projects (http://lists.FreeBSD.org/mailman/listinfo/cvs-projects)	`/usr/projects`	All changes to the projects tree
cvs-src (http://lists.FreeBSD.org/mailman/listinfo/cvs-src)	`/usr/src`	All changes to the src tree (generated by the svn-to-cvs importer commits)
svn-src-all (http://lists.FreeBSD.org/mailman/listinfo/svn-src-all)	`/usr/src`	All changes to the Subversion repository (except for `user` and `projects`)
svn-src-head (http://lists.FreeBSD.org/mailman/listinfo/svn-src-head)	`/usr/src`	All changes to the “head” branch of the Subversion repository (the FreeBSD-CURRENT branch)
svn-src-projects (http://lists.FreeBSD.org/mailman/listinfo/svn-src-projects)	`/usr/projects`	All changes to the `projects` area of the src Subversion repository

List	Source area	Area Description (source for)
svn-src-release (http://lists.FreeBSD.org/mailman/listinfo/svn-src-release)	`/usr/src`	All changes to the `releases` area of the src Subversion repository
svn-src-releng (http://lists.FreeBSD.org/mailman/listinfo/svn-src-releng)	`/usr/src`	All changes to the `releng` branches of the src Subversion repository (the security / release engineering branches)
svn-src-stable (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable)	`/usr/src`	All changes to the all stable branches of the src Subversion repository
svn-src-stable-6 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-6)	`/usr/src`	All changes to the `stable/6` branch of the src Subversion repository
svn-src-stable-7 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-7)	`/usr/src`	All changes to the `stable/7` branch of the src Subversion repository
svn-src-stable-8 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-8)	`/usr/src`	All changes to the `stable/8` branch of the src Subversion repository
svn-src-stable-other (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-other)	`/usr/src`	All changes to the older `stable` branches of the src Subversion repository
svn-src-svnadmin (http://lists.FreeBSD.org/mailman/listinfo/svn-src-svnadmin)	`/usr/src`	All changes to the administrative scripts, hooks, and other configuration data of the src Subversion repository
svn-src-user (http://lists.FreeBSD.org/mailman/listinfo/svn-src-user)	`/usr/src`	All changes to the experimental `user` area of the src Subversion repository
svn-src-vendor (http://lists.FreeBSD.org/mailman/listinfo/svn-src-vendor)	`/usr/src`	All changes to the vendor work area of the src Subversion repository

C.1.2 How to Subscribe

To subscribe to a list, click on the list name above or go to http://lists.FreeBSD.org/mailman/listinfo and click on the

list that you are interested in. The list page should contain all of the necessary subscription instructions.

To actually post to a given list you simply send mail to <*listname*@FreeBSD.org>. It will then be redistributed to mailing list members world-wide.

To unsubscribe yourself from a list, click on the URL found at the bottom of every email received from the list. It is also possible to send an email to <*listname*-unsubscribe@FreeBSD.org> to unsubscribe yourself.

Again, we would like to request that you keep discussion in the technical mailing lists on a technical track. If you are only interested in important announcements then it is suggested that you join the FreeBSD announcements mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce), which is intended only for infrequent traffic.

C.1.3 List Charters

All FreeBSD mailing lists have certain basic rules which must be adhered to by anyone using them. Failure to comply with these guidelines will result in two (2) written warnings from the FreeBSD Postmaster <postmaster@FreeBSD.org>, after which, on a third offense, the poster will removed from all FreeBSD mailing lists and filtered from further posting to them. We regret that such rules and measures are necessary at all, but today's Internet is a pretty harsh environment, it would seem, and many fail to appreciate just how fragile some of its mechanisms are.

Rules of the road:

- The topic of any posting should adhere to the basic charter of the list it is posted to, e.g. if the list is about technical issues then your posting should contain technical discussion. Ongoing irrelevant chatter or flaming only detracts from the value of the mailing list for everyone on it and will not be tolerated. For free-form discussion on no particular topic, the FreeBSD chat mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat) is freely available and should be used instead.
- No posting should be made to more than 2 mailing lists, and only to 2 when a clear and obvious need to post to both lists exists. For most lists, there is already a great deal of subscriber overlap and except for the most esoteric mixes (say "-stable & -scsi"), there really is no reason to post to more than one list at a time. If a message is sent to you in such a way that multiple mailing lists appear on the `Cc` line then the `Cc` line should also be trimmed before sending it out again. *You are still responsible for your own cross-postings, no matter who the originator might have been.*
- Personal attacks and profanity (in the context of an argument) are not allowed, and that includes users and developers alike. Gross breaches of netiquette, like excerpting or reposting private mail when permission to do so was not and would not be forthcoming, are frowned upon but not specifically enforced. *However*, there are also very few cases where such content would fit within the charter of a list and it would therefore probably rate a warning (or ban) on that basis alone.
- Advertising of non-FreeBSD related products or services is strictly prohibited and will result in an immediate ban if it is clear that the offender is advertising by spam.

Individual list charters:

freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi)

ACPI and power management development

freebsd-afs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-afs)

Andrew File System

This list is for discussion on porting and using AFS from CMU/Transarc

freebsd-announce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce)

Important events / milestones

This is the mailing list for people interested only in occasional announcements of significant FreeBSD events. This includes announcements about snapshots and other releases. It contains announcements of new FreeBSD capabilities. It may contain calls for volunteers etc. This is a low volume, strictly moderated mailing list.

freebsd-arch (http://lists.FreeBSD.org/mailman/listinfo/freebsd-arch)

Architecture and design discussions

This list is for discussion of the FreeBSD architecture. Messages will mostly be kept strictly technical in nature. Examples of suitable topics are:

- How to re-vamp the build system to have several customized builds running at the same time.
- What needs to be fixed with VFS to make Heidemann layers work.
- How do we change the device driver interface to be able to use the same drivers cleanly on many buses and architectures.
- How to write a network driver.

freebsd-audit (http://lists.FreeBSD.org/mailman/listinfo/freebsd-audit)

Source code audit project

This is the mailing list for the FreeBSD source code audit project. Although this was originally intended for security-related changes, its charter has been expanded to review any code changes.

This list is very heavy on patches, and is probably of no interest to the average FreeBSD user. Security discussions not related to a particular code change are held on freebsd-security. Conversely, all developers are encouraged to send their patches here for review, especially if they touch a part of the system where a bug may adversely affect the integrity of the system.

freebsd-binup (http://lists.FreeBSD.org/mailman/listinfo/freebsd-binup)

FreeBSD Binary Update Project

This list exists to provide discussion for the binary update system, or **binup**. Design issues, implementation details, patches, bug reports, status reports, feature requests, commit logs, and all other things related to **binup** are fair game.

freebsd-bluetooth (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bluetooth)

Bluetooth in FreeBSD

This is the forum where FreeBSD's Bluetooth users congregate. Design issues, implementation details, patches, bug reports, status reports, feature requests, and all matters related to Bluetooth are fair game.

freebsd-bugbusters (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugbusters)

Coordination of the Problem Report handling effort

The purpose of this list is to serve as a coordination and discussion forum for the Bugmeister, his Bugbusters, and any other parties who have a genuine interest in the PR database. This list is not for discussions about specific bugs, patches or PRs.

freebsd-bugs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugs)

Bug reports

This is the mailing list for reporting bugs in FreeBSD. Whenever possible, bugs should be submitted using the send-pr(1) command or the WEB interface (http://www.FreeBSD.org/send-pr.html) to it.

freebsd-chat (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat)

Non technical items related to the FreeBSD community

This list contains the overflow from the other lists about non-technical, social information. It includes discussion about whether Jordan looks like a toon ferret or not, whether or not to type in capitals, who is drinking too much coffee, where the best beer is brewed, who is brewing beer in their basement, and so on. Occasional announcements of important events (such as upcoming parties, weddings, births, new jobs, etc) can be made to the technical lists, but the follow ups should be directed to this -chat list.

freebsd-core

FreeBSD core team

This is an internal mailing list for use by the core members. Messages can be sent to it when a serious FreeBSD-related matter requires arbitration or high-level scrutiny.

freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current)

Discussions about the use of FreeBSD-CURRENT

This is the mailing list for users of FreeBSD-CURRENT. It includes warnings about new features coming out in -CURRENT that will affect the users, and instructions on steps that must be taken to remain -CURRENT. Anyone running “CURRENT” must subscribe to this list. This is a technical mailing list for which strictly technical content is expected.

freebsd-cvsweb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cvsweb)

FreeBSD CVSweb Project

Technical discussions about use, development and maintenance of FreeBSD-CVSweb.

freebsd-doc (http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc)

Documentation project

This mailing list is for the discussion of issues and projects related to the creation of documentation for FreeBSD. The members of this mailing list are collectively referred to as “The FreeBSD Documentation Project”. It is an open list; feel free to join and contribute!

freebsd-drivers (http://lists.FreeBSD.org/mailman/listinfo/freebsd-drivers)

Writing device drivers for FreeBSD

This is a forum for technical discussions related to device drivers on FreeBSD. It is primarily a place for device driver writers to ask questions about how to write device drivers using the APIs in the FreeBSD kernel.

freebsd-eclipse (http://lists.FreeBSD.org/mailman/listinfo/freebsd-eclipse)

FreeBSD users of Eclipse IDE, tools, rich client applications and ports.

The intention of this list is to provide mutual support for everything to do with choosing, installing, using, developing and maintaining the Eclipse IDE, tools, rich client applications on the FreeBSD platform and assisting with the porting of Eclipse IDE and plugins to the FreeBSD environment.

The intention is also to facilitate exchange of information between the Eclipse community and the FreeBSD community to the mutual benefit of both.

Although this list is focused primarily on the needs of Eclipse users it will also provide a forum for those who would like to develop FreeBSD specific applications using the Eclipse framework.

freebsd-embedded (http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded)

Using FreeBSD in embedded applications

This list discusses topics related to using FreeBSD in embedded systems. This is a technical mailing list for which strictly technical content is expected. For the purpose of this list we define embedded systems as those computing devices which are not desktops and which usually serve a single purpose as opposed to being general computing environments. Examples include, but are not limited to, all kinds of phone handsets, network equipment such as routers, switches and PBXs, remote measuring equipment, PDAs, Point Of Sale systems, and so on.

freebsd-emulation (http://lists.FreeBSD.org/mailman/listinfo/freebsd-emulation)

Emulation of other systems such as Linux/MS-DOS/Windows

This is a forum for technical discussions related to running programs written for other operating systems on FreeBSD.

freebsd-eol (http://lists.FreeBSD.org/mailman/listinfo/freebsd-eol)

Peer support of FreeBSD-related software that is no longer supported by the FreeBSD project.

This list is for those interested in providing or making use of peer support of FreeBSD-related software for which the FreeBSD project no longer provides official support (e.g., in the form of security advisories and patches).

freebsd-firewire (http://lists.FreeBSD.org/mailman/listinfo/freebsd-firewire)

FireWire (iLink, IEEE 1394)

This is a mailing list for discussion of the design and implementation of a FireWire (aka IEEE 1394 aka iLink) subsystem for FreeBSD. Relevant topics specifically include the standards, bus devices and their protocols, adapter boards/cards/chips sets, and the architecture and implementation of code for their proper support.

freebsd-fs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-fs)

File systems

Discussions concerning FreeBSD file systems. This is a technical mailing list for which strictly technical content is expected.

freebsd-gecko (http://lists.FreeBSD.org/mailman/listinfo/freebsd-gecko)

Gecko Rendering Engine

This is a forum about **Gecko** applications using FreeBSD.

Discussion centers around Gecko Ports applications, their installation, their development and their support within FreeBSD.

freebsd-geom (http://lists.FreeBSD.org/mailman/listinfo/freebsd-geom)

GEOM

Discussions specific to GEOM and related implementations. This is a technical mailing list for which strictly technical content is expected.

freebsd-gnome (http://lists.FreeBSD.org/mailman/listinfo/freebsd-gnome)

GNOME

Discussions concerning The **GNOME** Desktop Environment for FreeBSD systems. This is a technical mailing list for which strictly technical content is expected.

freebsd-ipfw (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ipfw)

IP Firewall

This is the forum for technical discussions concerning the redesign of the IP firewall code in FreeBSD. This is a technical mailing list for which strictly technical content is expected.

freebsd-ia64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia64)

Porting FreeBSD to IA64

This is a technical mailing list for individuals actively working on porting FreeBSD to the IA-64 platform from Intel, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome.

freebsd-isdn (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn)

ISDN Communications

This is the mailing list for people discussing the development of ISDN support for FreeBSD.

freebsd-java (http://lists.FreeBSD.org/mailman/listinfo/freebsd-java)

Java Development

This is the mailing list for people discussing the development of significant Java applications for FreeBSD and the porting and maintenance of JDKs.

freebsd-jobs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-jobs)

Jobs offered and sought

This is a forum for posting employment notices and resumes specifically related to FreeBSD, e.g. if you are seeking FreeBSD-related employment or have a job involving FreeBSD to advertise then this is the right place. This is *not* a mailing list for general employment issues since adequate forums for that already exist elsewhere.

Note that this list, like other `FreeBSD.org` mailing lists, is distributed worldwide. Thus, you need to be clear about location and the extent to which telecommuting or assistance with relocation is available.

Email should use open formats only — preferably plain text, but basic Portable Document Format (PDF), HTML, and a few others are acceptable to many readers. Closed formats such as Microsoft Word (`.doc`) will be rejected by the mailing list server.

freebsd-kde (https://mail.kde.org/mailman/listinfo/kde-freebsd)

KDE

Discussions concerning **KDE** on FreeBSD systems. This is a technical mailing list for which strictly technical content is expected.

freebsd-hackers (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers)

Technical discussions

This is a forum for technical discussions related to FreeBSD. This is the primary technical mailing list. It is for individuals actively working on FreeBSD, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome. This is a technical mailing list for which strictly technical content is expected.

freebsd-hardware (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hardware)

General discussion of FreeBSD hardware

General discussion about the types of hardware that FreeBSD runs on, various problems and suggestions concerning what to buy or avoid.

freebsd-hubs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hubs)

Mirror sites

Announcements and discussion for people who run FreeBSD mirror sites.

freebsd-isp (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp)

Issues for Internet Service Providers

This mailing list is for discussing topics relevant to Internet Service Providers (ISPs) using FreeBSD. This is a technical mailing list for which strictly technical content is expected.

freebsd-mono (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mono)

Mono and C# applications on FreeBSD

This is a list for discussions related to the Mono development framework on FreeBSD. This is a technical mailing list. It is for individuals actively working on porting Mono or C# applications to FreeBSD, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome.

freebsd-openoffice (http://lists.FreeBSD.org/mailman/listinfo/freebsd-openoffice)

OpenOffice.org

Discussions concerning the porting and maintenance of **OpenOffice.org** and **StarOffice**.

freebsd-performance (http://lists.FreeBSD.org/mailman/listinfo/freebsd-performance)

Discussions about tuning or speeding up FreeBSD

This mailing list exists to provide a place for hackers, administrators, and/or concerned parties to discuss performance related topics pertaining to FreeBSD. Acceptable topics includes talking about FreeBSD installations that are either under high load, are experiencing performance problems, or are pushing the limits of FreeBSD. Concerned parties that are willing to work toward improving the performance of FreeBSD are highly encouraged to subscribe to this list. This is a highly technical list ideally suited for experienced FreeBSD users, hackers, or administrators interested in keeping FreeBSD fast, robust, and scalable. This list is not a question-and-answer list that replaces reading through documentation, but it is a place to make contributions or inquire about unanswered performance related topics.

freebsd-pf (http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf)

Discussion and questions about the packet filter firewall system

Discussion concerning the packet filter (pf) firewall system in terms of FreeBSD. Technical discussion and user questions are both welcome. This list is also a place to discuss the ALTQ QoS framework.

freebsd-platforms (http://lists.FreeBSD.org/mailman/listinfo/freebsd-platforms)

Porting to Non Intel platforms

Cross-platform FreeBSD issues, general discussion and proposals for non Intel FreeBSD ports. This is a technical mailing list for which strictly technical content is expected.

freebsd-policy (http://lists.FreeBSD.org/mailman/listinfo/freebsd-policy)

Core team policy decisions

This is a low volume, read-only mailing list for FreeBSD Core Team Policy decisions.

freebsd-ports (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports)

Discussion of "ports"

Discussions concerning FreeBSD's "ports collection" (`/usr/ports`), ports infrastructure, and general ports coordination efforts. This is a technical mailing list for which strictly technical content is expected.

freebsd-ports-bugs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs)

Discussion of "ports" bugs

Discussions concerning problem reports for FreeBSD's "ports collection" (`/usr/ports`), proposed ports, or modifications to ports. This is a technical mailing list for which strictly technical content is expected.

freebsd-proliant (http://lists.FreeBSD.org/mailman/listinfo/freebsd-proliant)

Technical discussion of FreeBSD on HP ProLiant server platforms

This mailing list is to be used for the technical discussion of the usage of FreeBSD on HP ProLiant servers, including the discussion of ProLiant-specific drivers, management software, configuration tools, and BIOS updates. As such, this is the primary place to discuss the hpasmd, hpasmcli, and hpacucli modules.

freebsd-python (http://lists.FreeBSD.org/mailman/listinfo/freebsd-python)

Python on FreeBSD

This is a list for discussions related to improving Python-support on FreeBSD. This is a technical mailing list. It is for individuals working on porting Python, its 3rd party modules and **Zope** stuff to FreeBSD. Individuals interested in following the technical discussion are also welcome.

freebsd-questions (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions)

User questions

This is the mailing list for questions about FreeBSD. You should not send “how to” questions to the technical lists unless you consider the question to be pretty technical.

freebsd-ruby (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ruby)

FreeBSD-specific Ruby discussions

This is a list for discussions related to the Ruby support on FreeBSD. This is a technical mailing list. It is for individuals working on Ruby ports, 3rd party libraries and frameworks.

Individuals interested in the technical discussion are also welcome.

freebsd-scsi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-scsi)

SCSI subsystem

This is the mailing list for people working on the SCSI subsystem for FreeBSD. This is a technical mailing list for which strictly technical content is expected.

freebsd-security (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security)

Security issues

FreeBSD computer security issues (DES, Kerberos, known security holes and fixes, etc). This is a technical mailing list for which strictly technical discussion is expected. Note that this is not a question-and-answer list, but that contributions (BOTH question AND answer) to the FAQ are welcome.

freebsd-security-notifications (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications)

Security Notifications

Notifications of FreeBSD security problems and fixes. This is not a discussion list. The discussion list is FreeBSD-security.

freebsd-small (http://lists.FreeBSD.org/mailman/listinfo/freebsd-small)

Using FreeBSD in embedded applications

This list discusses topics related to unusually small and embedded FreeBSD installations. This is a technical mailing list for which strictly technical content is expected.

> **Note:** This list has been obsoleted by freebsd-embedded (http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded).

freebsd-stable (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable)

Discussions about the use of FreeBSD-STABLE

This is the mailing list for users of FreeBSD-STABLE. It includes warnings about new features coming out in -STABLE that will affect the users, and instructions on steps that must be taken to remain -STABLE. Anyone running “STABLE” should subscribe to this list. This is a technical mailing list for which strictly technical content is expected.

freebsd-standards (http://lists.FreeBSD.org/mailman/listinfo/freebsd-standards)

C99 & POSIX Conformance

This is a forum for technical discussions related to FreeBSD Conformance to the C99 and the POSIX standards.

freebsd-toolchain (http://lists.FreeBSD.org/mailman/listinfo/freebsd-toolchain)

Maintenance of FreeBSD’s integrated toolchain

This is the mailing list for discussions related to the maintenance of the toolchain shipped with FreeBSD. This could include the state of Clang and GCC, but also pieces of software such as assemblers, linkers and debuggers.

freebsd-usb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-usb)

Discussing FreeBSD support for USB

This is a mailing list for technical discussions related to FreeBSD support for USB.

freebsd-user-groups (http://lists.FreeBSD.org/mailman/listinfo/freebsd-user-groups)

User Group Coordination List

This is the mailing list for the coordinators from each of the local area Users Groups to discuss matters with each other and a designated individual from the Core Team. This mail list should be limited to meeting synopsis and coordination of projects that span User Groups.

freebsd-vendors (http://lists.FreeBSD.org/mailman/listinfo/freebsd-vendors)

Vendors

Coordination discussions between The FreeBSD Project and Vendors of software and hardware for FreeBSD.

freebsd-virtualization (http://lists.FreeBSD.org/mailman/listinfo/freebsd-virtualization)

Discussion of various virtualization techniques supported by FreeBSD

A list to discuss the various virtualization techniques supported by FreeBSD. On one hand the focus will be on the implementation of the basic functionality as well as adding new features. On the other hand users will have a forum to ask for help in case of problems or to discuss their use cases.

freebsd-wip-status (http://lists.FreeBSD.org/mailman/listinfo/freebsd-wip-status)

FreeBSD Work-In-Progress Status

This mailing list can be used to announce creation and progress of your FreeBSD related work. Messages will be moderated. It is suggested to send the message "To:" a more topical FreeBSD list and only "BCC:" this list. This way your WIP can also be discussed on the topical list, as no discussion is allowed on this list.

Look inside the archives for examples of suitable messages.

An editorial digest of the messages to this list might be posted to the FreeBSD website every few month as part of the Status Reports [1]. You can find more examples and past reports there, too.

freebsd-xen (http://lists.FreeBSD.org/mailman/listinfo/freebsd-xen)

Discussion of the FreeBSD port to Xen — implementation and usage

A list that focuses on the FreeBSD Xen port. The anticipated traffic level is small enough that it is intended as a forum for both technical discussions of the implementation and design details as well as administrative deployment issues.

C.1.4 Filtering on the Mailing Lists

The FreeBSD mailing lists are filtered in multiple ways to avoid the distribution of spam, viruses, and other unwanted emails. The filtering actions described in this section do not include all those used to protect the mailing lists.

Only certain types of attachments are allowed on the mailing lists. All attachments with a MIME content type not found in the list below will be stripped before an email is distributed on the mailing lists.

- 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

Note: Some of the mailing lists might allow attachments of other MIME content types, but the above list should be applicable for most of the mailing lists.

If an email contains both an HTML and a plain text version, the HTML version will be removed. If an email contains only an HTML version, it will be converted to plain text.

C.2 Usenet Newsgroups

In addition to two FreeBSD specific newsgroups, there are many others in which FreeBSD is discussed or are otherwise relevant to FreeBSD users. Keyword searchable archives

(http://minnie.tuhs.org/BSD-info/bsdnews_search.html) are available for some of these newsgroups from courtesy of Warren Toomey <wkt@cs.adfa.edu.au>.

C.2.1 BSD Specific Newsgroups

- 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) (German)
- fr.comp.os.bsd (news:fr.comp.os.bsd) (French)
- it.comp.os.freebsd (news:it.comp.os.freebsd) (Italian)
- tw.bbs.comp.386bsd (news:tw.bbs.comp.386bsd) (Traditional Chinese)

C.2.2 Other UNIX Newsgroups of Interest

- 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 World Wide Web Servers

C.3.1 Forums, Blogs, and Social Networks

- The FreeBSD Forums (http://forums.freebsd.org/) provide a web based discussion forum for FreeBSD questions and technical discussion.
- Planet FreeBSD (http://planet.freebsdish.org/) offers an aggregation feed of dozens of blogs written by FreeBSD developers. Many developers use this to post quick notes about what they are working on, new patches, and other works in progress.
- The BSDConferences YouTube Channel (http://www.youtube.com/bsdconferences) provides a collection of high quality videos from BSD Conferences around the world. This is a great way to watch key developers give presentations about new work in FreeBSD.

C.3.2 Official Mirrors

Central Servers, Argentina, Armenia, Australia, Austria, Belgium, Brazil, Bulgaria, Canada, China, Costa Rica, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hong Kong, Hungary, Iceland, Indonesia, Italy, Japan, Korea, Kuwait, Kyrgyzstan, Latvia, Lithuania, Netherlands, Norway, Philippines, Portugal, Romania, Russia, San Marino, Singapore, Slovak Republic, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Thailand, Turkey, Ukraine, United Kingdom, USA.

(as of 2010/11/13 13:50:55 UTC)

- Central Servers
  - http://www.FreeBSD.org/
- Argentina
  - http://www.ar.FreeBSD.org/
- Armenia
  - http://www1.am.FreeBSD.org/ (IPv6)

- Australia
  - http://www.au.FreeBSD.org/
  - http://www2.au.FreeBSD.org/
- Austria
  - http://www.at.FreeBSD.org/ (IPv6)
  - http://www2.at.FreeBSD.org/ (IPv6)
- Belgium
  - http://freebsd.unixtech.be/
- Brazil
  - http://www.br.FreeBSD.org/ (IPv6)
  - http://www2.br.FreeBSD.org/www.freebsd.org/
  - http://www3.br.FreeBSD.org/
- Bulgaria
  - http://www.bg.FreeBSD.org/
  - http://www2.bg.FreeBSD.org/
- Canada
  - http://www.ca.FreeBSD.org/
  - http://www2.ca.FreeBSD.org/
- China
  - http://www.cn.FreeBSD.org/
-

Costa Rica

- http://www1.cr.FreeBSD.org/

- Czech Republic
  - http://www.cz.FreeBSD.org/ (IPv6)

- Denmark
  - http://www.dk.FreeBSD.org/ (IPv6)
  - http://www3.dk.FreeBSD.org/

- Estonia
  - http://www.ee.FreeBSD.org/

- Finland
  - http://www.fi.FreeBSD.org/
  - http://www2.fi.FreeBSD.org/

- France
  - http://www.fr.FreeBSD.org/
  - http://www1.fr.FreeBSD.org/

- Germany
  - http://www.de.FreeBSD.org/

- Greece
  - http://www.gr.FreeBSD.org/

-

Hong Kong

- http://www.hk.FreeBSD.org/

- Hungary
  - http://www.hu.FreeBSD.org/
  - http://www2.hu.FreeBSD.org/

- Iceland
  - http://www.is.FreeBSD.org/

- Indonesia
  - http://www.id.FreeBSD.org/

- Italy
  - http://www.it.FreeBSD.org/
  - http://www.gufi.org/mirrors/www.freebsd.org/data/

- Japan
  - http://www.jp.FreeBSD.org/www.FreeBSD.org/ (IPv6)

- Korea
  - http://www.kr.FreeBSD.org/
  - http://www2.kr.FreeBSD.org/

- Kuwait
  - http://www.kw.FreeBSD.org/

-

Kyrgyzstan

- http://www.kg.FreeBSD.org/

- Latvia
  - http://www.lv.FreeBSD.org/
  - http://www2.lv.FreeBSD.org/

- Lithuania
  - http://www.lt.FreeBSD.org/

- Netherlands
  - http://www.nl.FreeBSD.org/
  - http://www2.nl.FreeBSD.org/

- Norway
  - http://www.no.FreeBSD.org/

- Philippines
  - http://www.FreeBSD.org.ph/

- Portugal
  - http://www.pt.FreeBSD.org/
  - http://www1.pt.FreeBSD.org/
  - http://www4.pt.FreeBSD.org/
  - http://www5.pt.FreeBSD.org/

- Romania
  - http://www.ro.FreeBSD.org/

- http://www1.ro.FreeBSD.org/
- http://www2.ro.FreeBSD.org/
- http://www3.ro.FreeBSD.org/

- Russia
  - http://www.ru.FreeBSD.org/
  - http://www2.ru.FreeBSD.org/
  - http://www3.ru.FreeBSD.org/
  - http://www4.ru.FreeBSD.org/
  - http://www5.ru.FreeBSD.org/

- San Marino
  - http://www.sm.FreeBSD.org/

- Singapore
  - http://www2.sg.FreeBSD.org/

- Slovak Republic
  - http://www.sk.FreeBSD.org/

- Slovenia
  - http://www.si.FreeBSD.org/
  - http://www2.si.FreeBSD.org/

- South Africa
  - http://www.za.FreeBSD.org/
  - http://www2.za.FreeBSD.org/

-

Spain

- http://www.es.FreeBSD.org/
- http://www2.es.FreeBSD.org/
- http://www3.es.FreeBSD.org/

- Sweden
  - http://www.se.FreeBSD.org/
  - http://www2.se.FreeBSD.org/

- Switzerland
  - http://www.ch.FreeBSD.org/
  - http://www2.ch.FreeBSD.org/

- Taiwan
  - http://www.tw.FreeBSD.org/ (IPv6)
  - http://www2.tw.FreeBSD.org/
  - http://www3.tw.FreeBSD.org/
  - http://www4.tw.FreeBSD.org/
  - http://www5.tw.FreeBSD.org/ (IPv6)
  - http://www6.tw.FreeBSD.org/
  - http://www7.tw.FreeBSD.org/

- Thailand
  - http://www.th.FreeBSD.org/

- Turkey
  - http://www.tr.FreeBSD.org/
  - http://www2.tr.FreeBSD.org/
  - http://www3.tr.FreeBSD.org/ (IPv6)

- Ukraine
  - http://www.ua.FreeBSD.org/
  - http://www2.ua.FreeBSD.org/
  - http://www5.ua.FreeBSD.org/
  - http://www4.ua.FreeBSD.org/

- United Kingdom
  - http://www1.uk.FreeBSD.org/
  - http://www3.uk.FreeBSD.org/

- USA
  - http://www2.us.FreeBSD.org/
  - http://www4.us.FreeBSD.org/ (IPv6)
  - http://www5.us.FreeBSD.org/ (IPv6)

C.4 Email Addresses

The following user groups provide FreeBSD related email addresses for their members. The listed administrator reserves the right to revoke the address if it is abused in any way.

Domain	Facilities	User Group	Administrator
ukug.uk.FreeBSD.org	Forwarding only	<ukfreebsd@uk.FreeBSD.org>	Lee Johnston <lee@uk.FreeBSD.org>

Notes

1. http://www.freebsd.org/news/status/

Appendix D.
PGP Keys

In case you need to verify a signature or send encrypted email to one of the officers or developers a number of keys are provided here for your convenience. A complete keyring of `FreeBSD.org` users is available for download from http://www.FreeBSD.org/doc/pgpkeyring.txt.

D.1 Officers

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 Secretary <core-secretary@FreeBSD.org>

```
pub  1024R/FF8AE305 2002-01-08 core-secretary@FreeBSD.org
     Key fingerprint = CE EF 8A 48 70 00 B5 A9  55 69 DE 87 E3 9A E1 CD
```

D.1.3 Ports Management Team Secretary <portmgr-secretary@FreeBSD.org>

```
pub   1024D/7414629C 2005-11-30
      Key fingerprint = D50C BA61 8DC6 C42E 4C05  BF9A 79F6 E071 7414 629C
uid                  FreeBSD portmgr secretary <portmgr-secretary@FreeBSD.org>
sub   2048g/80B696E6 2005-11-30
```

D.2 Core Team Members

D.2.1 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.2.2 Konstantin Belousov <kib@FreeBSD.org>

```
pub   1024D/DD4C6F88 2004-07-29
      Key fingerprint = 39DA E615 A45C 111D 777B  3AD0 0B7F 8C04 DD4C 6F88
uid                  Konstantin Belousov <kib@freebsd.org>
uid                  Konstantin Belousov <konstantin.belousov@zoral.com.ua>
uid                  Kostik Belousov <kostikbel@ukr.net>
uid                  Kostik Belousov <kostikbel@gmail.com>
sub   2048g/18488597 2004-07-29
```

D.2.3 Wilko Bulte <wilko@FreeBSD.org>

```
pub   1024D/186B8DBD 2006-07-29
      Key fingerprint = 07C2 6CB3 9C18 D290 6C5F  8879 CF83 EC86 186B 8DBD
uid                  Wilko Bulte (wilko@FreeBSD.org) <wilko@FreeBSD.org>
sub   2048g/1C4683F1 2006-07-29
```

D.2.4 Brooks Davis <brooks@FreeBSD.org>

```
pub  1024D/F2381AD4 2001-02-10 Brooks Davis (The Aerospace Corporation) <brooks@aero.org>
     Key fingerprint = 655D 519C 26A7 82E7 2529  9BF0 5D8E 8BE9 F238 1AD4
uid                            Brooks Davis <brooks@one-eyed-alien.net>
uid                            Brooks Davis <brooks@FreeBSD.org>
uid                            Brooks Davis <brooks@aero.org>
sub  2048g/CFDACA7A 2003-01-25 [expires: 2008-01-24]
sub  1024g/42921194 2001-02-10 [expires: 2009-02-08]
```

D.2.5 Warner Losh <imp@FreeBSD.org>

```
pub   1024D/1EF6D8A7 2006-08-15
      Key fingerprint = AEC9 99C1 3212 1A86 93A6  A96B DB9F 6F12 1EF6 D8A7
uid                  M. Warner Losh <imp@bsdimp.com>
sub   4096g/34FC5B17 2006-08-15
```

D.2.6 Pav Lucistnik <pav@FreeBSD.org>

```
pub  1024D/C14EB282 2003-08-25 Pav Lucistnik <pav@FreeBSD.org>
     Key fingerprint = 2622 B7E3 7DA5 5C53 2079  855B 9ED7 583F C14E B282
uid                            Pav Lucistnik <pav@oook.cz>
sub  1024g/7287A947 2003-08-25
```

D.2.7 Colin Percival <cperciva@FreeBSD.org>

```
pub   1024D/0C6A6A6E 2009-01-12
      Key fingerprint = EAF4 8BBA 7CC7 7A30 FEFC  0DA9 38CE CA69 0C6A 6A6E
uid                  Colin Percival <cperciva@tarsnap.com>
uid                  Colin Percival <cperciva@FreeBSD.org>
uid                  Colin Percival <cperciva@alumni.sfu.ca>
sub   2048g/DC606691 2009-01-12
```

D.2.8 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>
sub   1024g/8CD251FF 2001-06-12
```

D.3 Developers

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 <thomas@goodking.ca>
uid                  Thomas Abthorpe <tabthorpe@goodking.org>
sub   2048R/8CA60EE0 2010-05-28
```

D.3.3 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.3.4 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>
sub   1024g/3B13C209 2003-04-09
```

D.3.5 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.3.6 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.3.7 Dimitry Andric <dim@FreeBSD.org>

```
pub   1024D/2E2096A3 1997-11-17
      Key fingerprint = 7AB4 62D2 CE35 FC6D 4239  4FCD B05E A30A 2E20 96A3
uid                  Dimitry Andric <dimitry@andric.com>
uid                  Dimitry Andric <dim@xs4all.nl>
uid                  Dimitry Andric <dimitry.andric@tomtom.com>
uid                  [jpeg image of size 5132]
```

```
uid                  Dimitry Andric <dim@nah6.com>
uid                  Dimitry Andric <dim@FreeBSD.org>
sub   4096g/6852A5C5 1997-11-17
```

D.3.8 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.3.9 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.3.10 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.3.11 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.3.12 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.3.13 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 <gavin@16squared.co.uk>
uid                  Gavin Atkinson (FreeBSD key) <gavin@FreeBSD.org>
uid                  Gavin Atkinson (Work e-mail) <ga9@york.ac.uk>
uid                  Gavin Atkinson <gavin.atkinson@ury.york.ac.uk>
sub   2048g/58F40B3D 2005-02-18
```

D.3.14 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.3.15 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.3.16 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.3.17 Glen Barber <gjb@FreeBSD.org>

```
pub   2048R/A0B946A3 2010-08-03
      Key fingerprint = 78B3 42BA 26C7 B2AC 681E  A7BE 524F 0C37 A0B9 46A3
uid                  Glen Barber <glen.j.barber@gmail.com>
uid                  Glen Barber <gjb35@drexel.edu>
uid                  Glen Barber <gjb@glenbarber.us>
uid                  Glen Barber <gjb@FreeBSD.org>
sub   2048R/6C0527E5 2010-08-03
```

D.3.18 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.3.19 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.3.20 Doug Barton <dougb@FreeBSD.org>

```
pub   2048R/1A1ABC84 2010-03-23
      Key fingerprint = E352 0E14 9D05 3533 C33A  67DB 5CC6 86F1 1A1A BC84
uid                  Douglas Barton <dougb@dougbarton.us>
uid                  Douglas Barton <dougb@FreeBSD.org>
uid                  [jpeg image of size 6140]
sub   3072R/498795B4 2010-03-23
      Key fingerprint = C0BE C1E3 8DC8 D7F4 8E6C  732B 0C14 D9CF 4987 95B4
```

D.3.21 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.3.22 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.3.23 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.3.24 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.3.25 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.3.26 Roman Bogorodskiy <novel@FreeBSD.org>

```
pub   1024R/1DAACA46 2004-05-25 [expires: 2009-04-26]
      Key fingerprint = AC27 CF29 5E51 E53F 8C8D  DB90 8074 5B38 1DAA CA46
uid                  Roman Bogorodskiy <novel@FreeBSD.org>
uid                  Roman Bogorodskiy <bogorodskiy@gmail.com>
uid                  Roman Bogorodskiy <bogorodskiy@inbox.ru>
uid                  Roman Bogorodskiy <novel@clublife.ru>
```

D.3.27 Renato Botelho <garga@FreeBSD.org>

```
pub   1024D/2244EDA9 2003-12-16 [expires: 2015-10-18]
      Key fingerprint = 4006 C844 BC51 AD75 CE60  6E24 E824 5B89 2244 EDA9
uid                  Renato Botelho <garga@FreeBSD.org>
uid                  Renato Botelho <rbgarga@gmail.com>
uid                  Renato Botelho <garga@freebsdbrasil.com.br>
uid                  Renato Botelho <renato@galle.com.br>
uid                  Renato Botelho <freebsd@galle.com.br>
uid                  Renato Botelho <garga@brainsoft.com.br>
uid                  Renato Botelho <garga.bsd@gmail.com>
sub   1024g/7B295760 2003-12-16
```

D.3.28 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.3.29 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.3.30 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.3.31 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.3.32 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.3.33 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.3.34 Diane Bruce <db@FreeBSD.org>

```
pub   1024D/E08F5B15 2007-01-18
      Key fingerprint = A5FB 296B 5771 C1CD 6183  0FAB 77FF DCBE E08F 5B15
uid                  Diane Bruce <db@db.net>
uid                  Diane Bruce <db@FreeBSD.org>
sub   2048g/73281702 2007-01-18
```

D.3.35 Christian Brüffer <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.3.36 Markus Brüffer <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.3.37 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.3.38 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.3.39 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.3.40 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.3.41 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.3.42 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.3.43 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.3.44 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.3.45 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.3.46 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.3.47 Fukang Chen <loader@FreeBSD.org>

```
pub   1024D/40AB1752 2007-08-01 [expires: 2010-07-31]
      Key fingerprint = 98C4 6E6B 1C21 15E4 5042  01FC C7B7 E152 40AB 1752
uid                  loader <loader@FreeBSD.org>
sub   4096g/9E53A5C7 2007-08-01 [expires: 2010-07-31]
```

D.3.48 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.3.49 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.3.50 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.3.51 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.3.52 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.3.53 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.3.54 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.3.55 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.3.56 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.3.57 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.3.58 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.3.59 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.3.60 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.3.61 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.3.62 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 <so14k@so14k.com>
uid                  Brad Davis <brd@FreeBSD.org>
sub   2048g/1F29D404 2005-05-14 [expires: 2014-02-21]
```

D.3.63 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.3.64 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.3.65 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.3.66 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.3.67 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.3.68 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.3.69 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.3.70 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.3.71 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.3.72 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.3.73 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.3.74 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.3.75 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.3.76 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.3.77 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.3.78 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.3.79 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.3.80 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.3.81 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.3.82 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.3.83 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.3.84 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.3.85 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.3.86 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.3.87 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.3.88 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.3.89 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@icyb.net.ua>
sub   4096R/F9A4D312 2009-02-16
```

D.3.90 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.3.91 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.3.92 Sebastien Gioria <gioria@FreeBSD.org>

```
pub  1024D/7C8DA4F4 2002-02-09 Sebastien Gioria <eagle@freebsd-fr.org>
     Key fingerprint = 41F4 4885 7C23 6ED3 CC24  97AA 6DDD B426 7C8D A4F4
uid                            Sebastien Gioria <gioria@FreeBSD.ORG>
uid                            Sebastien Gioria <gioria@Francenet.fr>
uid                            Sebastien Gioria <gioria@fluxus.net>
sub  4096g/F147E4D3 2002-02-09
```

D.3.93 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.3.94 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.3.95 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.3.96 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.3.97 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.3.98 John-Mark Gurney <jmg@FreeBSD.org>

```
pub  1024R/3F9951F5 1997-02-11 John-Mark Gurney <johnmark@gladstone.uoregon.edu>
     Key fingerprint = B7 EC EF F8 AE ED A7 31  96 7A 22 B3 D8 56 36 F4
uid                            John-Mark Gurney <gurney_j@efn.org>
uid                            John-Mark Gurney <jmg@cs.uoregon.edu>
uid                            John-Mark Gurney <gurney_j@resnet.uoregon.edu>
```

D.3.99 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.3.100 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.3.101 Olli 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.3.102 Emanuel Haupt <ehaupt@FreeBSD.org>

```
pub   2048R/C06D09BE 2010-09-24 [expires: 2011-09-24]
      Key fingerprint = CC88 5081 78D1 39C3 B467  865A 348E F6CC C06D 09BE
uid                   Emanuel Haupt <ehaupt@FreeBSD.org>
sub   2048R/F658659F 2010-09-24 [expires: 2011-09-24]
```

D.3.103 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.3.104 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.3.105 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.3.106 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.3.107 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.3.108 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.3.109 Guy Helmer <ghelmer@FreeBSD.org>

```
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.3.110 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.3.111 Dennis Herrmann <dhn@FreeBSD.org>

```
pub   1024D/65181EA0 2008-09-07 [expires: 2009-03-06]
      Key fingerprint = D4DB A438 EB5E 1B26 C782  F969 820B 66B3 6518 1EA0
uid                  Dennis Herrmann (Vi veri universum vivus vici) <adox@mcx2.org>
sub   4096g/C003C5DD 2008-09-07 [expires: 2009-03-06]
```

D.3.112 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.3.113 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.3.114 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.3.115 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.3.116 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.3.117 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.3.118 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.3.119 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.3.120 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.3.121 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.3.122 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.3.123 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.3.124 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.3.125 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.3.126 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.3.127 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@bsdi.com>
uid                            [revoked] Josef Karthauser <joe@pavilion.net>
sub  2048g/1178B692 2000-10-19
```

D.3.128 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.3.129 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.3.130 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.3.131 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.3.132 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.3.133 Jung-uk Kim <jkim@FreeBSD.org>

```
pub   1024D/BF6A9D53 2004-04-07
      Key fingerprint = F841 0339 93EF D27D 32AD  3261 9A56 B2D5 BF6A 9D53
uid                  Jung-uk Kim <jkim@FreeBSD.org>
uid                  Jung-uk Kim <jkim@niksun.com>
sub   4096g/B01CA5A0 2004-04-07
```

D.3.134 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.3.135 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.3.136 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.3.137 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.3.138 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.3.139 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.3.140 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.3.141 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.czest.pl>
uid                  Wojciech A. Koszek <wkoszek@FreeBSD.org>
sub   4096g/3BBD20A5 2006-02-15
```

D.3.142 Steven Kreuzer <skreuzer@FreeBSD.org>

```
pub   1024D/E0D6F907 2009-03-16 [expires: 2011-03-16]
      Key fingerprint = 8D8F 14D6 ED9F 6BD0 7756  7A46 66BA B4B6 E0D6 F907
uid                  Steven Kreuzer <skreuzer@freebsd.org>
uid                  Steven Kreuzer <skreuzer@exit2shell.com>
sub   4096g/76940A06 2009-03-16 [expires: 2011-03-16]
```

D.3.143 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.3.144 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.3.145 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.3.146 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.3.147 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.3.148 René Ladan <rene@FreeBSD.org>

```
pub   1024D/E5642BFC 2008-11-03
      Key fingerprint = ADBC ECCD EB5F A6B4 549F  600D 8C9E 647A E564 2BFC
uid                  Rene Ladan <rene@freebsd.org>
sub   2048g/C54EA560 2008-11-03
```

D.3.149 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.3.150 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.3.151 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.3.152 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.3.153 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.3.154 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.3.155 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.3.156 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.3.157 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.3.158 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.3.159 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.3.160 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.3.161 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.3.162 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.3.163 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.3.164 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.3.165 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 [expires: 2012-01-10]
      Key fingerprint = 27EA 5D6C 9398 BA7F B205  8F70 04CE F812 3FCA 37C1
uid                  Xin LI <delphij@geekcn.org>
uid                  Xin LI <delphij@delphij.net>
uid                  Xin LI <delphij@FreeBSD.org>
sub   2048R/F956339F 2010-01-10 [expires: 2012-01-10]
```

D.3.166 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.3.167 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.3.168 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 <ulf.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.3.169 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.3.170 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.3.171 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.3.172 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 KeskinÃ¶z <arved@arved.at>
uid                  Tilman KeskinÃ¶z <arved@FreeBSD.org>
sub   1024g/FA351986 2002-06-03 [expires: 2013-09-07]
```

D.3.173 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.3.174 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.3.175 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.3.176 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.3.177 Remko Lodder <remko@FreeBSD.org>

```
pub   2048R/6EB8C8C8 2010-05-28 [expires: 2012-05-27]
      Key fingerprint = D692 91F9 F4EF D363 7F3F  4D17 9C75 DF7B 6EB8 C8C8
uid                  Remko Lodder (Remko Lodder's Key) <remko@FreeBSD.org>
sub   2048R/011C6AA0 2010-05-28 [expires: 2012-05-27]
```

D.3.178 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.3.179 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.3.180 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.3.181 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.3.182 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.3.183 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.3.184 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.3.185 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.3.186 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.3.187 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.3.188 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.3.189 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.3.190 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.3.191 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.3.192 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.3.193 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.3.194 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.3.195 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.3.196 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.3.197 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.3.198 Felippe 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                  Felippe de Meirelles Motta (FreeBSD Ports Committer) <lippe@FreeBSD.org>
sub   2048g/38E8EEF3 2008-09-02 [expires: 2010-09-02]
```

D.3.199 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.3.200 Akinori MUSHA <knu@FreeBSD.org>

```
pub  1024D/9FD9E1EE 2000-03-21 Akinori MUSHA <knu@and.or.jp>
     Key fingerprint = 081D 099C 1705 861D 4B70  B04A 920B EFC7 9FD9 E1EE
uid                            Akinori MUSHA <knu@FreeBSD.org>
uid                            Akinori MUSHA <knu@idaemons.org>
uid                            Akinori MUSHA <knu@ruby-lang.org>
sub  1024g/71BA9D45 2000-03-21
```

D.3.201 Thomas Möstl <tmm@FreeBSD.org>

```
pub  1024D/419C776C 2000-11-28 Thomas Moestl <tmm@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.3.202 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.3.203 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.3.204 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.3.205 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.3.206 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.3.207 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.3.208 Simon L. 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.3.209 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.3.210 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.3.211 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.3.212 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.3.213 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 [expires: 2010-10-13]
sub   2048g/6E5FD7D6 2006-10-22 [expires: 2010-10-14]
```

D.3.214 Josh Paetzel <jpaetzel@FreeBSD.org>

```
pub   1024D/27AFAECB 2007-05-11
      Key fingerprint = 8A48 EF36 5E9F 4EDA 5A8C  11B4 26F9 01F1 27AF AECB
uid                  Josh Paetzel (BSD UNIX) <josh@tcbug.org>
uid                  Josh Paetzel <josh@rephunter.net>
uid                  Josh Paetzel <josh@pcbsd.org>
uid                  Josh Paetzel <jpaetzel@FreeBSD.org>
sub   2048g/E0F5996B 2007-05-11
```

D.3.215 Gábor Páli <pgj@FreeBSD.org>

```
pub   1024D/9E3F9BE6 2008-04-17 [expires: 2013-04-16]
      Key fingerprint = DA0B 2143 0FC8 EE5F E211  D329 7D4B 6E18 9E3F 9BE6
uid                  Gabor PALI <pgj@FreeBSD.org>
uid                  PÁLI Gábor János <pali.gabor@gmail.com>
sub   2048g/A780C60B 2008-04-17 [expires: 2013-04-16]
```

D.3.216 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.3.217 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  BAFC 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.3.218 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.3.219 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.3.220 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.3.221 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.3.222 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.office1.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.office1.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.3.223 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.3.224 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.3.225 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.3.226 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.3.227 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.3.228 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.3.229 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.3.230 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.3.231 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.3.232 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.3.233 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.3.234 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.3.235 Lars Balker Rasmussen <lbr@FreeBSD.org>

```
pub   1024D/9EF6F27F 2006-04-30
      Key fingerprint = F251 28B7 897C 293E 04F8  71EE 4697 F477 9EF6 F27F
uid                  Lars Balker Rasmussen <lbr@FreeBSD.org>
sub   2048g/A8C1CFD4 2006-04-30
```

D.3.236 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.3.237 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.3.238 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.3.239 Benno Rice <benno@FreeBSD.org>

```
pub  1024D/87C59909 2002-01-16 Benno Rice <benno@FreeBSD.org>
     Key fingerprint = CE27 DADA 08E3 FAA3 88F1  5B31 5E34 705A 87C5 9909
uid                           Benno Rice <benno@jeamland.net>
sub  1024g/4F7C2BAD 2002-01-16 [expires: 2007-01-15]
```

D.3.240 Beech Rintoul <beech@FreeBSD.org>

```
pub   2048D/11753A7B 2010-11-15
      Key fingerprint = 4DEC C668 9EF9 2AC3 FBE6  99E3 40B3 595D 1175 3A7B
uid                  Beech Rintoul <beech@FreeBSD.org>
sub   2048g/A9AA3FE9 2010-11-15
```

D.3.241 Matteo Riondato <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 Riondato (Rionda) <matteo@FreeBSD.ORG>
uid                  Matteo Riondato (Rionda) <rionda@riondabsd.net>
uid                  Matteo Riondato (Rionda) <rionda@gufi.org>
uid                  Matteo Riondato (Rionda) <matteo@riondato.com>
uid                  Matteo Riondato (Rionda) <rionda@riondato.com>
uid                  Matteo Riondato (Rionda) <rionda@FreeSBIE.ORG>
uid                  Matteo Riondato (Rionda) <rionda@autistici.org>
sub   2048g/87C44A55 2008-09-23 [expires: 2009-09-23]
```

D.3.242 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.3.243 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.3.244 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.3.245 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.3.246 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.3.247 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.3.248 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.3.249 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.3.250 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.3.251 Ed Schouten <ed@FreeBSD.org>

```
pub   1024D/0D9E0B05 2006-03-21 [expires: 2011-03-20]
      Key fingerprint = 9476 D3D6 52BD F249 08A0  ACD5 E764 8318 0D9E 0B05
uid                  Ed Schouten (FreeBSD) <ed@FreeBSD.org>
uid                  Ed Schouten <ed@fxq.nl>
uid                  Ed Schouten (Fontys Hogescholen Eindhoven) <e.schouten@student.fontys.nl>
uid                  Ed Schouten (Dispuut Interlink) <ed@il.fontys.nl>
uid                  Ed Schouten <ed@80386.nl>
sub   4096g/80043EEA 2006-03-21 [expires: 2011-03-20]
```

D.3.252 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.3.253 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.3.254 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.3.255 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.3.256 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.3.257 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.3.258 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.3.259 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.3.260 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.3.261 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.3.262 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.3.263 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.3.264 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.3.265 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.3.266 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.3.267 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.3.268 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.3.269 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.3.270 Gleb Smirnoff <glebius@FreeBSD.org>

```
pub   1024D/1949DC80 2003-08-25
      Key fingerprint = 872C E14A 2F03 A3E8 D882  026E 5DE4 D7FE 1949 DC80
uid                  Gleb Smirnoff <glebius@FreeBSD.org>
uid                  Gleb Smirnoff <glebius@cell.sick.ru>
uid                  Gleb Smirnoff <glebius@bestcom.ru>
uid                  Gleb Smirnoff <glebius@rambler-co.ru>
```

```
uid                  Gleb Smirnoff <glebius@freebsd.org>
uid                  Gleb Smirnoff <glebius@freebsd.int.ru>
sub   1024g/A05118BD 2003-08-25
```

D.3.271 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.3.272 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 <csxbcs@comp.leeds.ac.uk>
uid                            Ben Smithurst <ben@scientia.demon.co.uk>
sub  1024g/347071FF 2001-07-11
```

D.3.273 Dag-Erling C. Smørgrav <des@FreeBSD.org>

```
pub   1024D/64EBE220 2006-11-11 [expires: 2011-05-31]
      Key fingerprint = 3A1C 8E68 952C 3305 6984  6486 30D4 3A6E 64EB E220
uid                  Dag-Erling SmÃ¸rgrav <des@des.no>
uid                  Dag-Erling SmÃ¸rgrav <des@freebsd.org>
uid                  [jpeg image of size 3315]
sub   2048g/920C3313 2006-11-11 [expires: 2011-05-31]
```

D.3.274 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.3.275 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.3.276 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.3.277 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.3.278 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.3.279 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 SpÃ¶rlein <uqs@spoerlein.net>
```

```
uid                  Ulrich Spoerlein <uspoerlein@gmail.com>
uid                  Ulrich SpÃ¶rlein (The FreeBSD Project) <uqs@FreeBSD.org>
uid                  Ulrich SpÃ¶rlein <ulrich.spoerlein@web.de>
sub   2048R/162E8BD2 2010-01-27 [expires: 2015-01-26]
```

D.3.280 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.3.281 Vsevolod Stakhov <vsevolod@FreeBSD.org>

```
pub   1024D/213D0033 2005-03-14 [expires: 2008-03-13]
      Key fingerprint = B852 0010 761E 944A C76D  D447 A25D C12C 213D 0033
uid                  Vsevolod Stakhov <vsevolod@FreeBSD.org>
uid                  Vsevolod Stakhov <cebka@jet.msk.su>
uid                  Vsevolod Stakhov <vsevolod@highsecure.ru>
sub   2048g/786F2187 2005-03-14 [expires: 2008-03-13]
```

D.3.282 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.3.283 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.3.284 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.3.285 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.3.286 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.3.287 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.3.288 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.cs
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.3.289 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.3.290 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.3.291 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.3.292 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.3.293 Yoshihiro Takahashi <nyan@FreeBSD.org>

```
pub  1024D/8394B81F 2001-10-15 Yoshihiro TAKAHASHI <nyan@jp.FreeBSD.org>
     Key fingerprint = D4FA D8CA 2AED FCF4 90A3  3569 8666 0500 8394 B81F
uid                           Yoshihiro TAKAHASHI <nyan@furiru.org>
uid                           Yoshihiro TAKAHASHI <nyan@FreeBSD.org>
sub  1024g/B796F020 2001-10-15
```

D.3.294 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.3.295 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.3.296 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 TartiÃ¨re <romain@blogreen.org>
uid                  Romain TartiÃ¨re (FreeBSD) <romain@FreeBSD.org>
sub   3072R/C1B2B656 2010-04-09
sub   3072R/8F8125F4 2010-04-09
```

D.3.297 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.3.298 Ion-Mihai Tetcu <itetcu@FreeBSD.org>

```
pub   1024D/21FFA1E5 2008-05-08 [expires: 2010-05-08]
      Key fingerprint = A880 42DD BD71 BAA5 AED7  AEA2 27B1 88BA 21FF A1E5
uid                  Ion-Mihai "IOnut" Tetcu <itetcu@FreeBSD.org>
sub   2048g/0B30E680 2008-05-08 [expires: 2010-05-08]
```

D.3.299 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.3.300 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.3.301 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.3.302 David Thiel <lx@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 <lx@FreeBSD.org>
sub   2048g/B9BD92C5 2006-11-30 [expires: 2011-11-29]
```

D.3.303 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.3.304 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.3.305 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.3.306 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.3.307 Yar Tikhiy <yar@FreeBSD.org>

```
pub   1024D/EA04CF5A 2008-08-31
      Key fingerprint = C063 6788 AFF2 A62F 06B7  516D 200F 06AF EA04 CF5A
uid                  Yar Tikhiy <yar@freebsd.org>
sub   2048g/20443F06 2008-08-31
```

D.3.308 Jilles Tjoelker <jilles@FreeBSD.org>

```
pub   1024D/A813D5EE 2001-02-18
      Key fingerprint = 0C82 44F5 0A1B 84E4 A9DD  7032 5102 275F A813 D5EE
uid                  Jilles Tjoelker <jilles@stack.nl>
uid                  Jilles Tjoelker <tjoelker@zonnet.nl>
uid                  Jilles Tjoelker (FreeBSD) <jilles@FreeBSD.org>
sub   2048g/B94834AC 2001-02-18
```

D.3.309 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.3.310 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.3.311 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.3.312 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.3.313 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.3.314 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.3.315 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.3.316 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.3.317 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.3.318 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.3.319 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.3.320 Adam Weinberger <adamw@FreeBSD.org>

```
pub  1024D/42C743FD 2002-10-12 Adam Weinberger <adam@vectors.cx>
     Key fingerprint = A980 3F2E 80A8 9619 9D1C  82E8 A3C2 8CD9 42C7 43FD
sub  1024g/15D67628 2002-10-12
```

D.3.321 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.3.322 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.3.323 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.3.324 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.3.325 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.3.326 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.3.327 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.3.328 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.3.329 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.3.330 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.3.331 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.3.332 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.3.333 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
```

FreeBSD Glossary

This glossary contains terms and acronyms used within the FreeBSD community and documentation.

A

ACL

See: Access Control List

ACPI

See: Advanced Configuration and Power Interface

AMD

See: Automatic Mount Daemon

AML

See: ACPI Machine Language

API

See: Application Programming Interface

APIC

See: Advanced Programmable Interrupt Controller

APM

See: Advanced Power Management

APOP

See: Authenticated Post Office Protocol

ASL

See: ACPI Source Language

ATA

See: Advanced Technology Attachment

ATM

See: Asynchronous Transfer Mode

ACPI Machine Language

Pseudocode, interpreted by a virtual machine within an ACPI-compliant operating system, providing a layer between the underlying hardware and the documented interface presented to the OS.

ACPI Source Language

The programming language AML is written in.

Access Control List

A list of permissions attached to an object, usually either a file or a network device.

Advanced Configuration and Power Interface

A specification which provides an abstraction of the interface the hardware presents to the operating system, so that the operating system should need to know nothing about the underlying hardware to make the most of it. ACPI evolves and supercedes the functionality provided previously by APM, PNPBIOS and other technologies, and provides facilities for controlling power consumption, machine suspension, device enabling and disabling, etc.

Application Programming Interface

A set of procedures, protocols and tools that specify the canonical interaction of one or more program parts; how, when and why they do work together, and what data they share or operate on.

Advanced Power Management

An API enabling the operating system to work in conjunction with the BIOS in order to achieve power management. APM has been superseded by the much more generic and powerful ACPI specification for most applications.

Advanced Programmable Interrupt Controller

Advanced Technology Attachment

Asynchronous Transfer Mode

Authenticated Post Office Protocol

Automatic Mount Daemon

A daemon that automatically mounts a filesystem when a file or directory within that filesystem is accessed.

B

BAR

See: Base Address Register

BIND

See: Berkeley Internet Name Domain

BIOS

See: Basic Input/Output System

BSD

See: Berkeley Software Distribution

Base Address Register

The registers that determine which address range a PCI device will respond to.

Basic Input/Output System

The definition of BIOS depends a bit on the context. Some people refer to it as the ROM chip with a basic set of routines to provide an interface between software and hardware. Others refer to it as the set of routines contained in the chip that help in bootstrapping the system. Some might also refer to it as the screen used to configure the boostrapping process. The BIOS is PC-specific but other systems have something similar.

Berkeley Internet Name Domain

An implementation of the DNS protocols.

Berkeley Software Distribution

This is the name that the Computer Systems Research Group (CSRG) at The University of California at Berkeley (http://www.berkeley.edu) gave to their improvements and modifications to AT&T's 32V UNIX. FreeBSD is a descendant of the CSRG work.

Bikeshed Building

A phenomenon whereby many people will give an opinion on an uncomplicated topic, whilst a complex topic receives little or no discussion. See the FAQ (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/misc.html#BIKESHED-PAINTING) for the origin of the term.

C

CD

See: Carrier Detect

CHAP

See: Challenge Handshake Authentication Protocol

CLIP

See: Classical IP over ATM

COFF

See: Common Object File Format

CPU

See: Central Processing Unit

CTS

See: Clear To Send

CVS

See: Concurrent Versions System

Carrier Detect

An RS232C signal indicating that a carrier has been detected.

Central Processing Unit

Also known as the processor. This is the brain of the computer where all calculations take place. There are a number of different architectures with different instruction sets. Among the more well-known are the Intel-x86 and derivatives, Sun SPARC, PowerPC, and Alpha.

Challenge Handshake Authentication Protocol

A method of authenticating a user, based on a secret shared between client and server.

Classical IP over ATM

Clear To Send

An RS232C signal giving the remote system permission to send data.

See Also: Request To Send.

Common Object File Format

Concurrent Versions System

A version control system, providing a method of working with and keeping track of many different revisions of files. CVS provides the ability to extract, merge and revert individual changes or sets of changes, and offers the ability to keep track of which changes were made, by who and for what reason.

D

DAC

See: Discretionary Access Control

DDB

See: Debugger

DES

See: Data Encryption Standard

DHCP

See: Dynamic Host Configuration Protocol

DNS

See: Domain Name System

DSDT

See: Differentiated System Description Table

DSR

See: Data Set Ready

DTR

See: Data Terminal Ready

DVMRP

See: Distance-Vector Multicast Routing Protocol

Discretionary Access Control

Data Encryption Standard

A method of encrypting information, traditionally used as the method of encryption for UNIX passwords and the crypt(3) function.

Data Set Ready

An RS232C signal sent from the modem to the computer or terminal indicating a readiness to send and receive data.

See Also: Data Terminal Ready.

Data Terminal Ready

An RS232C signal sent from the computer or terminal to the modem indicating a readiness to send and receive data.

Debugger

An interactive in-kernel facility for examining the status of a system, often used after a system has crashed to establish the events surrounding the failure.

Differentiated System Description Table

An ACPI table, supplying basic configuration information about the base system.

Distance-Vector Multicast Routing Protocol

Domain Name System

The system that converts humanly readable hostnames (i.e., mail.example.net) to Internet addresses and vice versa.

Dynamic Host Configuration Protocol

A protocol that dynamically assigns IP addresses to a computer (host) when it requests one from the server. The address assignment is called a “lease”.

E

ECOFF

See: Extended COFF

ELF

See: Executable and Linking Format

ESP

See: Encapsulated Security Payload

Encapsulated Security Payload

Executable and Linking Format

Extended COFF

F

FADT

See: Fixed ACPI Description Table

FAT

See: File Allocation Table

FAT16

See: File Allocation Table (16-bit)

FTP

See: File Transfer Protocol

File Allocation Table

File Allocation Table (16-bit)

File Transfer Protocol

A member of the family of high-level protocols implemented on top of TCP which can be used to transfer files over a TCP/IP network.

Fixed ACPI Description Table

G

GUI

See: Graphical User Interface

Giant

The name of a mutual exclusion mechanism (a `sleep mutex`) that protects a large set of kernel resources. Although a simple locking mechanism was adequate in the days where a machine might have only a few dozen processes, one networking card, and certainly only one processor, in current times it is an unacceptable performance bottleneck. FreeBSD developers are actively working to replace it with locks that protect individual resources, which will allow a much greater degree of parallelism for both single-processor and multi-processor machines.

Graphical User Interface

A system where the user and computer interact with graphics.

H

HTML

See: HyperText Markup Language

HUP

See: HangUp

HangUp

HyperText Markup Language

The markup language used to create web pages.

I

I/O

See: Input/Output

IASL

See: Intel's ASL compiler

IMAP

See: Internet Message Access Protocol

IP

See: Internet Protocol

IPFW

See: IP Firewall

IPP

See: Internet Printing Protocol

IPv4

See: IP Version 4

IPv6

See: IP Version 6

ISP

See: Internet Service Provider

IP Firewall

IP Version 4

The IP protocol version 4, which uses 32 bits for addressing. This version is still the most widely used, but it is slowly being replaced with IPv6.

See Also: IP Version 6.

IP Version 6

The new IP protocol. Invented because the address space in IPv4 is running out. Uses 128 bits for addressing.

Input/Output

Intel's ASL compiler

Intel's compiler for converting ASL into AML.

Internet Message Access Protocol

A protocol for accessing email messages on a mail server, characterised by the messages usually being kept on the server as opposed to being downloaded to the mail reader client.

See Also: Post Office Protocol Version 3.

Internet Printing Protocol

Internet Protocol

The packet transmitting protocol that is the basic protocol on the Internet. Originally developed at the U.S. Department of Defense and an extremly important part of the TCP/IP stack. Without the Internet Protocol, the Internet would not have become what it is today. For more information, see RFC 791 (ftp://ftp.rfc-editor.org/in-notes/rfc791.txt).

Internet Service Provider

A company that provides access to the Internet.

K

KAME

Japanese for "turtle", the term KAME is used in computing circles to refer to the KAME Project (http://www.kame.net/), who work on an implementation of IPv6.

KDC

See: Key Distribution Center

KLD

See: Kernel ld(1)

KSE

See: Kernel Scheduler Entities

KVA

See: Kernel Virtual Address

Kbps

See: Kilo Bits Per Second

Kernel ld(1)

A method of dynamically loading functionality into a FreeBSD kernel without rebooting the system.

Kernel Scheduler Entities

A kernel-supported threading system. See the project home page (http://www.FreeBSD.org/kse) for further details.

Kernel Virtual Address

Key Distribution Center

Kilo Bits Per Second

Used to measure bandwith (how much data can pass a given point at a specified amount of time). Alternates to the Kilo prefix include Mega, Giga, Tera, and so forth.

L

LAN

See: Local Area Network

LOR

See: Lock Order Reversal

LPD

See: Line Printer Daemon

Line Printer Daemon

Local Area Network

A network used on a local area, e.g. office, home, or so forth.

Lock Order Reversal

The FreeBSD kernel uses a number of resource locks to arbitrate contention for those resources. A run-time lock diagnostic system found in FreeBSD-CURRENT kernels (but removed for releases), called witness(4), detects the potential for deadlocks due to locking errors. (witness(4) is actually slightly conservative, so it is possible to get false positives.) A true positive report indicates that “if you were unlucky, a deadlock would have happened here”.

True positive LORs tend to get fixed quickly, so check http://lists.FreeBSD.org/mailman/listinfo/freebsd-current and the LORs Seen (http://sources.zabbadoz.net/freebsd/lor.html) page before posting to the mailing lists.

M

MAC

See: Mandatory Access Control

MADT

See: Multiple APIC Description Table

MFC

See: Merge From Current

MFP4

See: Merge From Perforce

MFS

See: Merge From Stable

MIT

See: Massachusetts Institute of Technology

MLS

See: Multi-Level Security

MOTD

See: Message Of The Day

MTA

See: Mail Transfer Agent

MUA

See: Mail User Agent

Mail Transfer Agent

An application used to transfer email. An MTA has traditionally been part of the BSD base system. Today Sendmail is included in the base system, but there are many other MTAs, such as postfix, qmail and Exim.

Mail User Agent

An application used by users to display and write email.

Mandatory Access Control

Massachusetts Institute of Technology

Merge From Current

To merge functionality or a patch from the -CURRENT branch to another, most often -STABLE.

Merge From Perforce

To merge functionality or a patch from the Perforce repository to the -CURRENT branch.

See Also: Perforce.

Merge From Stable

In the normal course of FreeBSD development, a change will be committed to the -CURRENT branch for testing before being merged to -STABLE. On rare occasions, a change will go into -STABLE first and then be merged to -CURRENT.

This term is also used when a patch is merged from -STABLE to a security branch.

See Also: Merge From Current.

Message Of The Day

A message, usually shown on login, often used to distribute information to users of the system.

Multi-Level Security

Multiple APIC Description Table

N

NAT

See: Network Address Translation

NDISulator

See: Project Evil

NFS

See: Network File System

NTFS

See: New Technology File System

NTP

See: Network Time Protocol

Network Address Translation

A technique where IP packets are rewritten on the way through a gateway, enabling many machines behind the gateway to effectively share a single IP address.

Network File System

New Technology File System

A filesystem developed by Microsoft and available in its “New Technology” operating systems, such as Windows 2000, Windows NT and Windows XP.

Network Time Protocol

A means of synchronizing clocks over a network.

O

OBE

See: Overtaken By Events

ODMR

See: On-Demand Mail Relay

OS

See: Operating System

On-Demand Mail Relay

Operating System

A set of programs, libraries and tools that provide access to the hardware resources of a computer. Operating systems range today from simplistic designs that support only one program running at a time, accessing only one device to fully multi-user, multi-tasking and multi-process systems that can serve thousands of users simultaneously, each of them running dozens of different applications.

Overtaken By Events

Indicates a suggested change (such as a Problem Report or a feature request) which is no longer relevant or applicable due to such things as later changes to FreeBSD, changes in networking standards, the affected hardware having since become obsolete, and so forth.

P

p4

See: Perforce

PAE

See: Physical Address Extensions

PAM

See: Pluggable Authentication Modules

PAP

See: Password Authentication Protocol

PC

See: Personal Computer

PCNSFD

See: Personal Computer Network File System Daemon

PDF

See: Portable Document Format

PID

See: Process ID

POLA

See: Principle Of Least Astonishment

POP

See: Post Office Protocol

POP3

See: Post Office Protocol Version 3

PPD

See: PostScript Printer Description

PPP

See: Point-to-Point Protocol

PPPoA

See: PPP over ATM

PPPoE

See: PPP over Ethernet

PPP over ATM

PPP over Ethernet

PR

See: Problem Report

PXE

See: Preboot eXecution Environment

Password Authentication Protocol

Perforce

A source code control product made by Perforce Software (http://www.perforce.com/) which is more advanced than CVS. Although not open source, its use is free of charge to open-source projects such as FreeBSD.

Some FreeBSD developers use a Perforce repository as a staging area for code that is considered too experimental for the -CURRENT branch.

Personal Computer

Personal Computer Network File System Daemon

Physical Address Extensions

A method of enabling access to up to 64 GB of RAM on systems which only physically have a 32-bit wide address space (and would therefore be limited to 4 GB without PAE).

Pluggable Authentication Modules

Point-to-Point Protocol

Pointy Hat

A mythical piece of headgear, much like a `dunce cap`, awarded to any FreeBSD committer who breaks the build, makes revision numbers go backwards, or creates any other kind of havoc in the source base. Any committer worth his or her salt will soon accumulate a large collection. The usage is (almost always?) humorous.

Portable Document Format

Post Office Protocol

See Also: Post Office Protocol Version 3.

Post Office Protocol Version 3

A protocol for accessing email messages on a mail server, characterised by the messages usually being downloaded from the server to the client, as opposed to remaining on the server.

See Also: Internet Message Access Protocol.

PostScript Printer Description

Preboot eXecution Environment

Principle Of Least Astonishment

As FreeBSD evolves, changes visible to the user should be kept as unsurprising as possible. For example, arbitrarily rearranging system startup variables in `/etc/defaults/rc.conf` violates POLA. Developers consider POLA when contemplating user-visible system changes.

Problem Report

A description of some kind of problem that has been found in either the FreeBSD source or documentation. See Writing FreeBSD Problem Reports (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/problem-reports/index.html).

Process ID

A number, unique to a particular process on a system, which identifies it and allows actions to be taken against it.

Project Evil

The working title for the NDISulator, written by Bill Paul, who named it referring to how awful it is (from a philosophical standpoint) to need to have something like this in the first place. The NDISulator is a special compatibility module to allow Microsoft Windows™ NDIS miniport network drivers to be used with FreeBSD/i386. This is usually the only way to use cards where the driver is closed-source. See `src/sys/compat/ndis/subr_ndis.c`.

R

RA

See: Router Advertisement

RAID

See: Redundant Array of Inexpensive Disks

RAM

See: Random Access Memory

RD

See: Received Data

RFC

See: Request For Comments

RISC

See: Reduced Instruction Set Computer

RPC

See: Remote Procedure Call

RS232C

See: Recommended Standard 232C

RTS

See: Request To Send

Random Access Memory

Revision Control System

The *Revision Control System* (RCS) is one of the oldest software suites that implement "revision control" for plain files. It allows the storage, retrieval, archival, logging, identification and merging of multiple revisions for each file. RCS consists of many small tools that work together. It lacks some of the features found in more modern revision control systems, like CVS or Subversion, but it is very simple to install, configure, and start using for a small set of files. Implementations of RCS can be found on every major UNIX-like OS.

See Also: Concurrent Versions System, Subversion.

Received Data

An RS232C pin or wire that data is recieved on.

See Also: Transmitted Data.

Recommended Standard 232C

A standard for communications between serial devices.

Reduced Instruction Set Computer

An approach to processor design where the operations the hardware can perform are simplified but made as general purpose as possible. This can lead to lower power consumption, fewer transistors and in some cases,

better performance and increased code density. Examples of RISC processors include the Alpha, SPARC, ARM and PowerPC.

Redundant Array of Inexpensive Disks

Remote Procedure Call

repocopy

See: Repository Copy

Repository Copy

A direct copying of files within the CVS repository.

Without a repocopy, if a file needed to be copied or moved to another place in the repository, the committer would run `cvs add` to put the file in its new location, and then `cvs rm` on the old file if the old copy was being removed.

The disadvantage of this method is that the history (i.e. the entries in the CVS logs) of the file would not be copied to the new location. As the FreeBSD Project considers this history very useful, a repository copy is often used instead. This is a process where one of the repository meisters will copy the files directly within the repository, rather than using the cvs(1) program.

Request For Comments

A set of documents defining Internet standards, protocols, and so forth. See www.rfc-editor.org (http://www.rfc-editor.org/).

Also used as a general term when someone has a suggested change and wants feedback.

Request To Send

An RS232C signal requesting that the remote system commences transmission of data.

See Also: Clear To Send.

Router Advertisement

S

SCI

See: System Control Interrupt

SCSI

See: Small Computer System Interface

SG

See: Signal Ground

SMB

See: Server Message Block

SMP

See: Symmetric MultiProcessor

SMTP

See: Simple Mail Transfer Protocol

SMTP AUTH

See: SMTP Authentication

SSH

See: Secure Shell

STR

See: Suspend To RAM

SVN

See: Subversion

SMTP Authentication

Server Message Block

Signal Ground

An RS232 pin or wire that is the ground reference for the signal.

Simple Mail Transfer Protocol

Secure Shell

Small Computer System Interface

Subversion

Subversion is a version control system, similar to CVS, but with an expanded feature list.

See Also: Concurrent Versions System.

Suspend To RAM

Symmetric MultiProcessor

System Control Interrupt

T

TCP

See: Transmission Control Protocol

TCP/IP

See: Transmission Control Protocol/Internet Protocol

TD

See: Transmitted Data

TFTP

See: Trivial FTP

TGT

See: Ticket-Granting Ticket

TSC

See: Time Stamp Counter

Ticket-Granting Ticket

Time Stamp Counter

A profiling counter internal to modern Pentium processors that counts core frequency clock ticks.

Transmission Control Protocol

A protocol that sits on top of (e.g.) the IP protocol and guarantees that packets are delivered in a reliable, ordered, fashion.

Transmission Control Protocol/Internet Protocol

The term for the combination of the TCP protocol running over the IP protocol. Much of the Internet runs over TCP/IP.

Transmitted Data

An RS232C pin or wire that data is transmitted on.

See Also: Received Data.

Trivial FTP

U

UDP

See: User Datagram Protocol

UFS1

See: Unix File System Version 1

UFS2

See: Unix File System Version 2

UID

See: User ID

URL

See: Uniform Resource Locator

USB

See: Universal Serial Bus

Uniform Resource Locator

A method of locating a resource, such as a document on the Internet and a means to identify that resource.

Unix File System Version 1

The original UNIX file system, sometimes called the Berkeley Fast File System.

Unix File System Version 2

An extension to UFS1, introduced in FreeBSD 5-CURRENT. UFS2 adds 64 bit block pointers (breaking the 1T barrier), support for extended file storage and other features.

Universal Serial Bus

A hardware standard used to connect a wide variety of computer peripherals to a universal interface.

User ID

A unique number assigned to each user of a computer, by which the resources and permissions assigned to that user can be identified.

User Datagram Protocol

A simple, unreliable datagram protocol which is used for exchanging data on a TCP/IP network. UDP does not provide error checking and correction like TCP.

V

VPN

See: Virtual Private Network

Virtual Private Network

A method of using a public telecommunication such as the Internet, to provide remote access to a localized network, such as a corporate LAN.

Colophon

This book is the combined work of hundreds of contributors to “The FreeBSD Documentation Project”. The text is authored in SGML according to the DocBook DTD and is formatted from SGML into many different presentation formats using **Jade**, an open source DSSSL engine. Norm Walsh’s DSSSL stylesheets were used with an additional customization layer to provide the presentation instructions for **Jade**. The printed version of this document would not be possible without Donald Knuth’s TEX typesetting language, Leslie Lamport’s LATEX, or Sebastian Rahtz’s **JadeTeX** macro package.