Diskless X Server: a how to guide

Jerry Kendall

jerry@kcis.com

With the help of some friends on the FreeBSD-hackers list, I have been able to create a diskless X terminal. The creation of the X terminal required first creating a diskless system with minimal utilities mounted via NFS. These same steps were used to create 2 separate diskless systems. The first is `altair.example.com`. A diskless X terminal that I run on my old 386DX-40. It has a 340Meg hard disk but, I did not want to change it. So, it boots from `antares.example.com` across a Ethernet. The second system is a 486DX2-66. I set up a diskless FreeBSD (complete) that uses no local disk. The server in that case is a Sun 670MP running SunOS™ 4.1.3. The same setup configuration was needed for both.

I am sure that there is stuff that needs to be added to this. Please send me any comments.

1 Creating the boot floppy (On the diskless system)

Since the network boot loaders will not work with some of the TSR's and such that MS-DOS® uses, it is best to create a dedicated boot floppy or, if you can, create an MS-DOS menu that will (via the `config.sys`/`autoexec.bat` files) ask what configuration to load when the system starts. The later is the method that I use and it works great. My MS-DOS (6.x) menu is below.

Example 1. `config.sys`

```
[menu]
menuitem=normal, normal
menuitem=unix, unix
[normal]
....
normal config.sys stuff
...
[unix]
```

Example 2. `autoexec.bat`

```
@ECHO OFF
goto %config%

:normal
...
normal autoexec.bat stuff
...
goto end

:unix
cd \netboot
nb8390.com

:end
```

2 Getting the network boot programs (On the server)

Compile the "net-boot" programs that are located in `/usr/src/sys/i386/boot/netboot`. You should read the comments at the top of the `Makefile`. Adjust as required. Make a backup of the original in case something goes wrong. When the build is done, there should be 2 MS-DOS executables, `nb8390.com` and `nb3c509.com`. One of these two programs will be what you need to run on the diskless server. It will load the kernel from the boot server. At this point, put both programs on the MS-DOS boot floppy created earlier.

3 Determine which program to run (On the diskless system)

If you know the chipset that your Ethernet adapter uses, this is easy. If you have the NS8390 chipset, or a NS8390 based chipset, use `nb8390.com`. If you have a 3Com® 509 based chipset, use the `nb3C509.com` boot program. If you are not sure which you have, try using one, if it says `No adapter found`, try the other. Beyond that, you are pretty much on your own.

4 Booting across the network

Boot the diskless system with out any config.sys/autoexec.bat files. Try running the boot program for your Ethernet adapter.

My Ethernet adapter is running in WD8013 16bit mode so I run `nb8390.com`

```
C:> cd \netboot
C:> nb8390

Boot from Network (Y/N) ?  Y

BOOTP/TFTP/NFS bootstrap loader     ESC for menu

Searching for adapter..
WD8013EBT base 0x0300, memory 0x000D8000, addr 00:40:01:43:26:66

Searching for server...
```

At this point, my diskless system is trying to find a machine to act as a boot server. Make note of the `addr` line above, you will need this number later. Reset the diskless system and modify your `config.sys` and `autoexec.bat` files to do these steps automatically for you. Perhaps in a menu. If you had to run `nb3c509.com` instead of `nb8390.com` the output is the same as above. If you got `No adapter found` at the `Searching for adapter...` message, verify that you did indeed set the compile time defines in the `Makefile` correctly.

5 Allowing systems to boot across the network (On the server)

Make sure the `/etc/inetd.conf` file has entries for tftp and bootps. Mine are listed below:

```
tftp dgram udp wait nobody /usr/libexec/tftpd tftpd /tftpboot
#
# Additions by who ever you are
bootps  dgram  udp  wait  root  /usr/libexec/bootpd bootpd /etc/bootptab
```

If you have to change the `/etc/inetd.conf` file, send a `HUP` signal to inetd(8). To do this, get the process ID of inetd with `ps -ax | grep inetd | grep -v grep`. Once you have it, send it a `HUP` signal. Do this by `kill -HUP <pid>`. This will force `inetd` to re-read its config file.

Did you remember to note the `addr` line from the output of the boot loader on the diskless system? Guess what, here is where you need it.

Add an entry to `/etc/bootptab` (maybe creating the file). It should be laid out identical to this:

```
altair:\
        :ht=ether:\
        :ha=004001432666:\
        :sm=255.255.255.0:\
        :hn:\
        :ds=199.246.76.1:\
        :ip=199.246.76.2:\
        :gw=199.246.76.1:\
        :vm=rfc1048:
```

The lines are as follows:

`altair`	the diskless systems name without the domain name.
`ht=ether`	the hardware type of "ethernet".
`ha=004001432666`	the hardware address (the number noted above).
`sm=255.255.255.0`	the subnet mask.
`hn`	tells server to send client's hostname to the client.
`ds=199.246.76.1`	tells the client who the domain server is.
`ip=199.246.76.2`	tells the client what its IP address is.
`gw=199.246.76.1`	tells the client what the default gateway is.
`vm=...`	just leave it there.

> **Note:** Be sure to set up the IP addresses correctly, the addresses above are my own.

Create the directory `/tftpboot` on the server it will contain the configuration files for the diskless systems that the server will serve. These files will be named `cfg.`*`ip`* where *`ip`* is the IP address of the diskless system. The config file for `altair` is `/tftpboot/cfg.199.246.76.2`. The contents is:

```
rootfs 199.246.76.1:/DiskLess/rootfs/altair
hostname altair.example.com
```

The line `hostname altair.example.com` simply tells the diskless system what its fully qualified domain name is.

The line `rootfs 199.246.76.1:/DiskLess/rootfs/altair` tells the diskless system where its NFS mountable root filesystem is located.

> **Note:** The NFS mounted root filesystem will be mounted *read only*.

The hierarchy for the diskless system can be re-mounted allowing read-write operations if required.

I use my spare 386DX-40 as a dedicated X terminal.

The hierarchy for `altair` is:

```
/
/bin
/etc
/tmp
/sbin
/dev
/dev/fd
/usr
/var
/var/run
```

The actual list of files is:

```
-r-xr-xr-x  1 root   wheel  779984 Dec 11 23:44 ./kernel
-r-xr-xr-x  1 root    bin   299008 Dec 12 00:22 ./bin/sh
-rw-r--r--  1 root   wheel     499 Dec 15 15:54 ./etc/rc
-rw-r--r--  1 root   wheel    1411 Dec 11 23:19 ./etc/ttys
-rw-r--r--  1 root   wheel     157 Dec 15 15:42 ./etc/hosts
-rw-r--r--  1 root    bin    1569 Dec 15 15:26 ./etc/XF86Config.altair
-r-x------  1 bin     bin   151552 Jun 10  1995 ./sbin/init
-r-xr-xr-x  1 bin     bin   176128 Jun 10  1995 ./sbin/ifconfig
-r-xr-xr-x  1 bin     bin   110592 Jun 10  1995 ./sbin/mount_nfs
-r-xr-xr-x  1 bin     bin   135168 Jun 10  1995 ./sbin/reboot
-r-xr-xr-x  1 root    bin    73728 Dec 13 22:38 ./sbin/mount
-r-xr-xr-x  1 root   wheel    1992 Jun 10  1995 ./dev/MAKEDEV.local
-r-xr-xr-x  1 root   wheel   24419 Jun 10  1995 ./dev/MAKEDEV
```

If you are not using devfs(5) (which is the default in FreeBSD 5.X), you should make sure that you do not forget to run `MAKEDEV all` in the `dev` directory.

My `/etc/rc` for `altair` is:

```
#!/bin/sh
#
PATH=/bin:/
export PATH
#
# configure the localhost
/sbin/ifconfig lo0 127.0.0.1
#
# configure the ethernet card
/sbin/ifconfig ed0 199.246.76.2 netmask 0xffffff00
#
# mount the root filesystem via NFS
/sbin/mount antares:/DiskLess/rootfs/altair /
#
# mount the /usr filesystem via NFS
/sbin/mount antares:/DiskLess/usr /usr
#
/usr/X11R6/bin/XF86_SVGA -query antares -xf86config /etc/XF86Config.altair > /dev/null 2>&1
#
# Reboot after X exits
/sbin/reboot
#
# We blew up....
exit 1
```

Any comments and all questions welcome.