PXE Booting Utilities With FreeBSD

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
	2016-10-07		WB

Contents

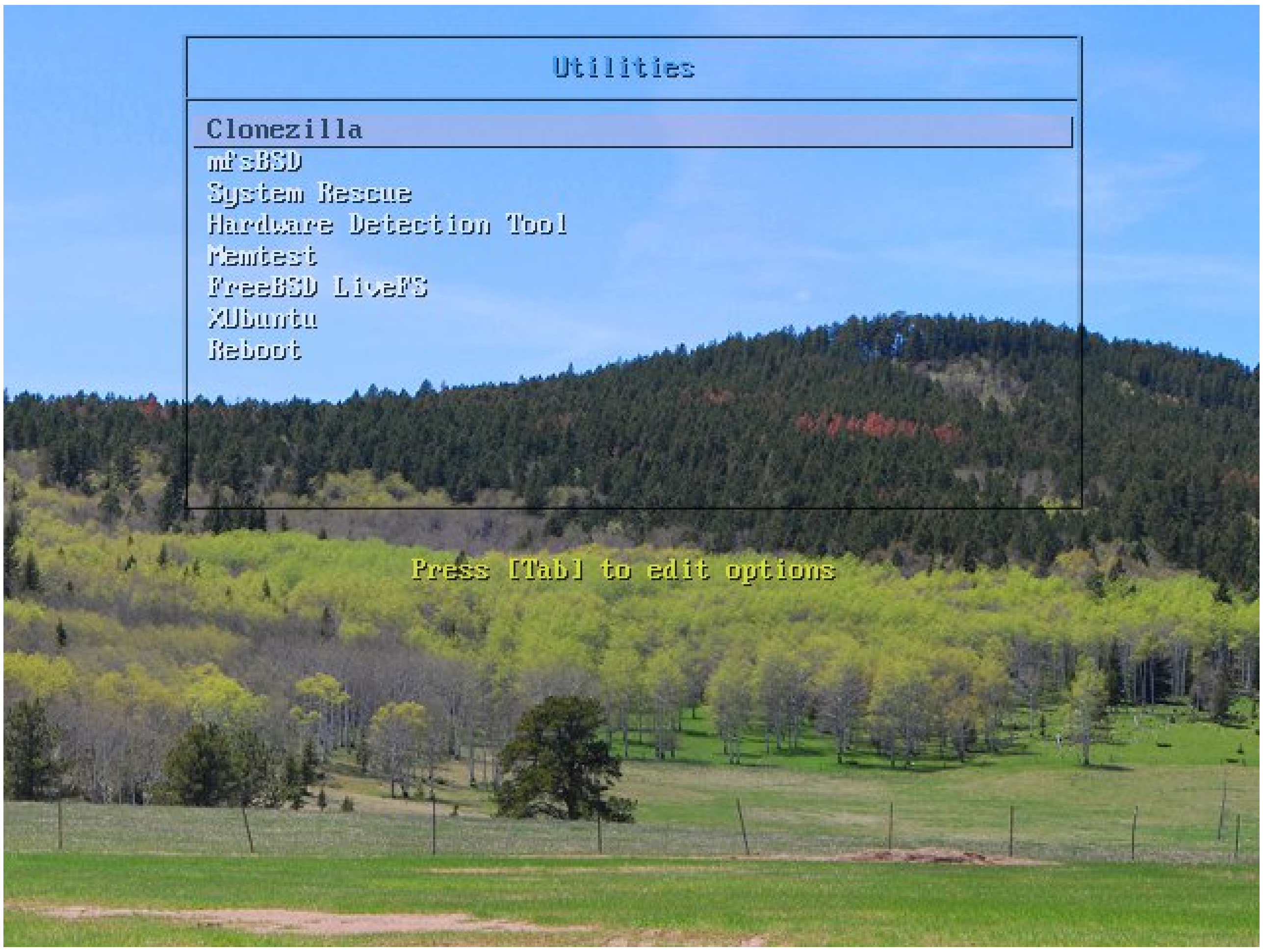
Last updated 2016-10-07

Available in HTML or PDF. Links to all my articles here. Created with AsciiDoc.

1 Introduction

Numerous useful utilities like Clonezilla, mfsBSD, and HDT are available as bootable images. A PXE boot server can make these utilities more convenient and accessible. With PXE, no external media is needed, just boot the target computer from the network.

We will set up all of the components necessary to provide PXE booting to any computer networked with the FreeBSD server. Instead of booting a single install CD image, this setup will boot into a menu with multiple utilities.



2 A Note About Security

A reasonable effort has been made to present configurations that are secure, but no guarantees are made or implied as to the safety or accuracy of this information. Use this information at your own risk, and test and verify before implementing it on your own network.

3 Baseline Setup

FreeBSD server address of 192.168.1.1.

Tip
A bare IP address is used in the menu files instead of a server hostname because Clonezilla and some other *initramfs*-based utilities have problems with DNS resolution on boot. See Preprocessing Menus With cpp for a way to define hostname constants in the menus.

net/isc-dhcp41-server installed and already working as a DHCP server.

ftp/tftp-hpa installed as a TFTP server. It's faster and more capable than FreeBSD's `tftpd`.

gpxelinux.0 from the SYSLINUX project will be used as a PXE boot loader.

4 DHCP Configuration

A computer booting with PXE from the network looks for a boot file name from the local DHCP server. Ours is the not-yet-installed *gpxelinux.0*.

Some TFTP clients expect the *next-server* value to provide the address of the TFTP server. In this example, the DHCP and TFTP servers are on the same computer, but that's not required.

Add these two settings to the "subnet" block of */usr/local/etc/dhcpd.conf* and restart the DHCP server:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
        range 192.168.1.100 192.168.1.200;
        next-server 192.168.1.1;  # TFTP server address
        filename "gpxelinux.0";   # PXE boot loader filename
}
```

```
# service isc-dhcpd restart
```

5 TFTP Configuration

After the PXE-booted computer receives a filename from DHCP, it has to actually load that file. TFTP is the only file transfer protocol supported by most PXE loaders.

The TFTP directory will hold all of the floppy and CD images or "live" filesystem trees from bootable utility disks. Since that directory can be quite large, this example places it in */usr*. A subdirectory to hold the boot menu files is also needed:

```
# mkdir -p /usr/tftpboot/images
# mkdir /usr/tftpboot/pxelinux.cfg
```

Enable the TFTP server in */etc/rc.conf*:

```
tftpd_enable="YES"
tftpd_flags="-p -s /usr/tftpboot -B 1024 --ipv4"
```

Tip
Testing on my network showed that limiting block size to 1024 bytes provided the best performance, and prevented errors sometimes seen with larger blocks. Forcing IPv4 was needed on a system without IPv6 support.

Tip

`tftp-hpa` can transfer large files. Other TFTP servers—or the TFTP clients used by utility disks—may fail with files larger than 32M. Alternate protocols are discussed in Advanced Options.

Warning

TFTP allows writes to files owned by the TFTP user. `in.tftpd` runs as user *nobody* by default; another user can be specified with *-u*. For read-only usage, make sure that none of the files in */usr/tftpboot* are owned or writable by the TFTP user.

Start the TFTP server:

```
# service tftpd start
```

6 PXE Boot Loader Installation

We'll use *gpxelinux.0*, a PXE boot loader from the SYSLINUX package. Download the archive from http://syslinux.zytor.com/-wiki/index.php/The_Syslinux_Project.

Copy the PXE boot loader and some other needed files to the right locations:

```
# tar tar xjf syslinux-4.04.tar.bz2 -C /tmp
# cp /tmp/syslinux-4.04/gpxe/gpxelinux.0 /usr/tftpboot/
# cp /tmp/syslinux-4.04/com32/menu/menu.c32 /usr/tftpboot/
# cp /tmp/syslinux-4.04/com32/menu/vesamenu.c32 /usr/tftpboot/
# cp /tmp/syslinux-4.04/com32/modules/reboot.c32 /usr/tftpboot/
# cp /tmp/syslinux-4.04/com32/modules/chain.c32 /usr/tftpboot/
# cp /tmp/syslinux-4.04/memdisk/memdisk /usr/tftpboot/
```

Table 1: File Descriptions

gpxelinux.0	PXE boot loader
menu.c32	text menu module
vesamenu.c32	graphic menu module
reboot.c32	reboot module
chain.c32	chain loader module
memdisk	memory disk emulator

7 Clonezilla

Clonezilla is a great bootable backup utility. Download the zip file from http://www.clonezilla.org and extract it into the image directory.

```
# unzip -j -o clonezilla-live-2.1.2-43-i686-pae.zip -d /usr/tftpboot/images/clonezilla
```

To create a menu that will be displayed after the PXE boot, create and edit the menu file */usr/tftpboot/pxelinux.cfg/default*:

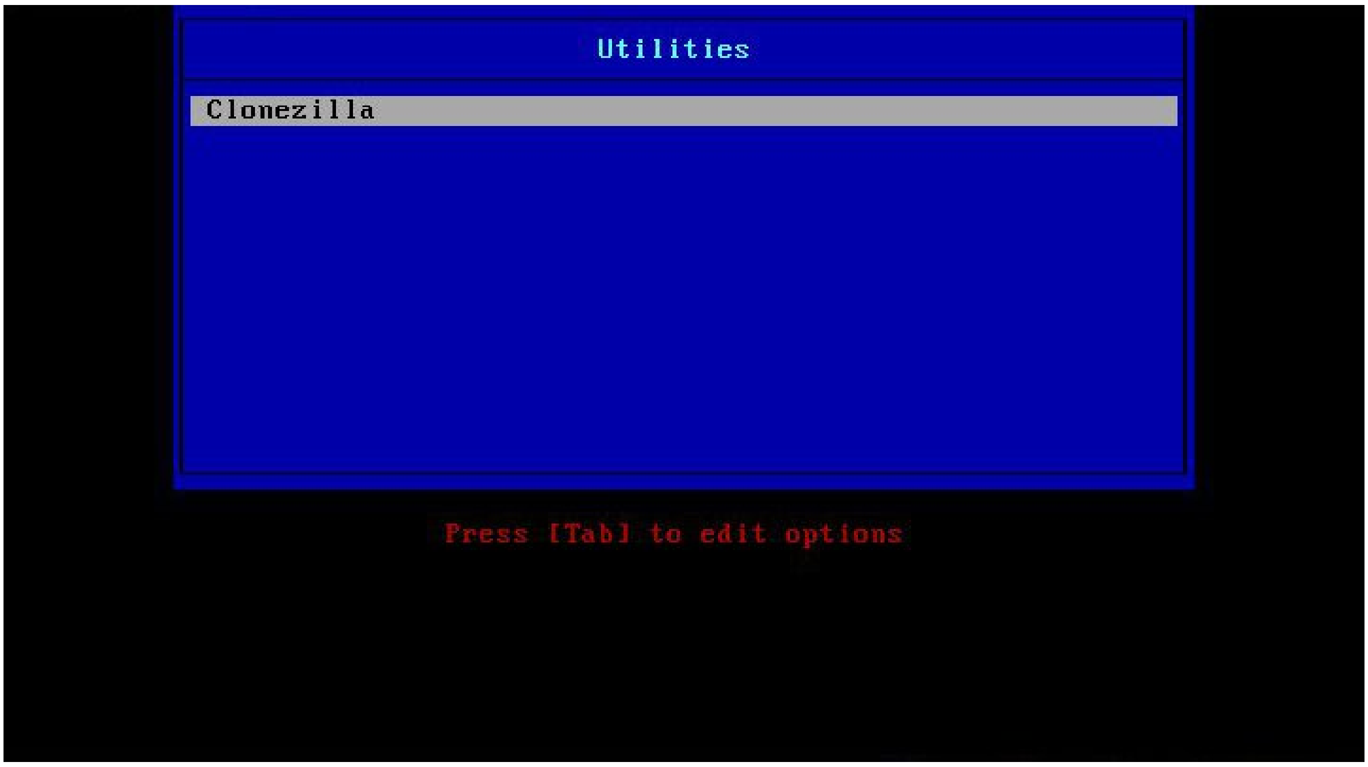
```
ui menu.c32
menu title Utilities

label clonezilla
  menu label Clonezilla
  kernel images/clonezilla/live/vmlinuz
  append boot=live live-config noswap nolocales edd=on nomodeset \
  ocs_live_run="ocs-live-general" ocs_live_extra_param="" \
  keyboard-layouts=us locales=en_US.UTF-8 \
  ocs_live_batch="no" vga=788 nosplash fetch=tftp://192.168.1.1/images/clonezilla/live/ ↩
      filesystem.squashfs
  initrd images/clonezilla/live/initrd.img
```

Table 2: Menu Commands

ui	loads the *menu.c32* text menu module.
menu title	sets the title shown at the top of the menu.
label	creates a menu entry.
kernel	loads the kernel; the path is relative to the TFTP server's root directory, */usr/tftpboot/*.
append	adds options to the kernel command line. Clonezilla uses *initramfs*, and *fetch* is an *initramfs*-specific option that tells the kernel where and how to download the live filesystem image.
initrd	points to the initial ramdisk image.

The live filesystem image will use about 100M of the target computer's memory.

To test the configuration, boot your target computer and choose "Boot from network" or "PXE". After a few seconds, a blue text menu with the single Clonezilla option appears. Press Enter to boot and start Clonezilla.



8 mfsBSD

mfsBSD is a great FreeBSD live image that runs completely from memory. Download the USB memstick image and copy it to a subdirectory in *images*. Making the filename shorter, like sticking with the 8.3 format, can help to avoid problems.

```
# mkdir /usr/tftpboot/images/mfsbsd
# cp /tmp/mfsbsd-9.1-RELEASE-i386.img /usr/tftpboot/images/mfsbsd/mfsbsd
```

Tip
Syslinux can handle disk images that have been compressed with zip or gzip. The smaller files load faster and CRC checks provide data integrity. Just compress the image and use the compressed image filename in the menu entry.

```
label mfsbsd
  menu label mfsBSD
  kernel memdisk
  initrd images/mfsbsd/mfsbsd.gz raw
```

9 System Rescue CD

Another very useful utility is from http://sysresccd.org/. As before, only a few files are needed from the ISO image:

```
# mkdir /usr/tftpboot/images/systemrescue/
# mount -t cd9660 /dev/`mdconfig -f systemrescuecd-x86-2.3.1.iso` /mnt
# cp /mnt/isolinux/rescuecd /usr/tftpboot/images/systemrescue/
# cp /mnt/isolinux/initram.igz /usr/tftpboot/images/systemrescue/
# cp /mnt/sysrcd.* /usr/tftpboot/images/systemrescue/
# umount /mnt
# mdconfig -d -u 0
```

The menu entry is also similar:

```
label systemrescue
  menu label System Rescue
  kernel images/systemrescue/rescuecd
  append setkmap=us netboot=tftp://192.168.1.1/images/systemrescue/sysrcd.dat
  initrd images/systemrescue/initram.igz
```

Tip
The System Rescue CD includes additional 64-bit and alternate kernels: *rescue64*, *altker32*, *altker64*. One of these other kernels may work on a target system that has problems booting or running the the *rescuecd* kernel. To add these kernels to the PXE menu, copy them from the *isolinux* directory of the ISO to the *images/rescuecd* directory and create additional menu entries in the *default* menu file. They all share the same *initram.igz* and *sysrcd.dat* files already present for the *rescuecd* kernel.

Note
sysrcd.dat is about 200M, and can take a long time to load via TFTP. Faster protocols are discussed in Advanced Options.

10 Hardware Detection Tool

Hardware Detection Tool is a useful utility from http://www.hdt-project.org. A *com32* version of HDT is included with the Syslinux archive already downloaded above.

Copy *hdt.c32* to the TFTP directory:

```
# cp /tmp/syslinux-4.04/com32/hdt/hdt.c32 /usr/tftpboot/
```

The menu entry is uncomplicated:

```
label hdt
  menu label Hardware Detection Tool
  kernel hdt.c32
```

`kernel` is used to load and start the *hdt.c32* module.

Tip
HDT auto-detects the presence of Memtest (installed in the next step) and shows it as an option on the HDT menus under Memory/Run Test.

Tip

Installing *pci.ids* and *modules.pcimap* files from a Linux CD allows HDT to display PCI information. After installing an XUbuntu image as shown in the XUbuntu With NFS section below, install the *sysutils/squashfs-tools* port. Then use it to extract those files from the XUbuntu squashfs filesystem:

```
# mkdir /usr/tftpboot/images/hdt
# cd /tmp
# unsquashfs /usr/tftpboot/images/xubuntu/casper/filesystem.squashfs /usr/share/misc/pci. ↩
    ids
# mv /tmp/squashfs-root/usr/share/misc/pci.ids /usr/tftpboot/images/hdt/
# rm -rf /tmp/squashfs-root
# unsquashfs /usr/tftpboot/images/xubuntu/casper/filesystem.squashfs /lib/modules ↩
    /2.6.38-8-generic/modules.pcimap
# mv /tmp/squashfs-root/lib/modules/2.6.38-8-generic/modules.pcimap /usr/tftpboot/images/ ↩
    hdt/
# rm -rf /tmp/squashfs-root
```

Compress the data files to speed loading:

```
# gzip /usr/tftpboot/images/hdt/*
```

Finally, modify the menu entry to point to these files.

```
label hdt
  menu label Hardware Detection Tool
  kernel hdt.c32
  append pciids=images/hdt/pci.ids.gz modules_pcimap=images/hdt/modules.pcimap.gz
```

11 Memtest

The classic Memtest86 is the next image to be installed. The System Rescue CD has a copy that's more convenient than downloading a separate image.

```
# mkdir /usr/tftpboot/images/memtest/
# mount -t cd9660 /dev/`mdconfig -f systemrescuecd-x86-2.3.1.iso` /mnt
# cp /mnt/bootdisk/memtestp /usr/tftpboot/images/memtest
# umount /mnt
```

The menu entry is brief:

```
label memtest
  menu label Memtest
  kernel images/memtest/memtestp
```

12 Reboot

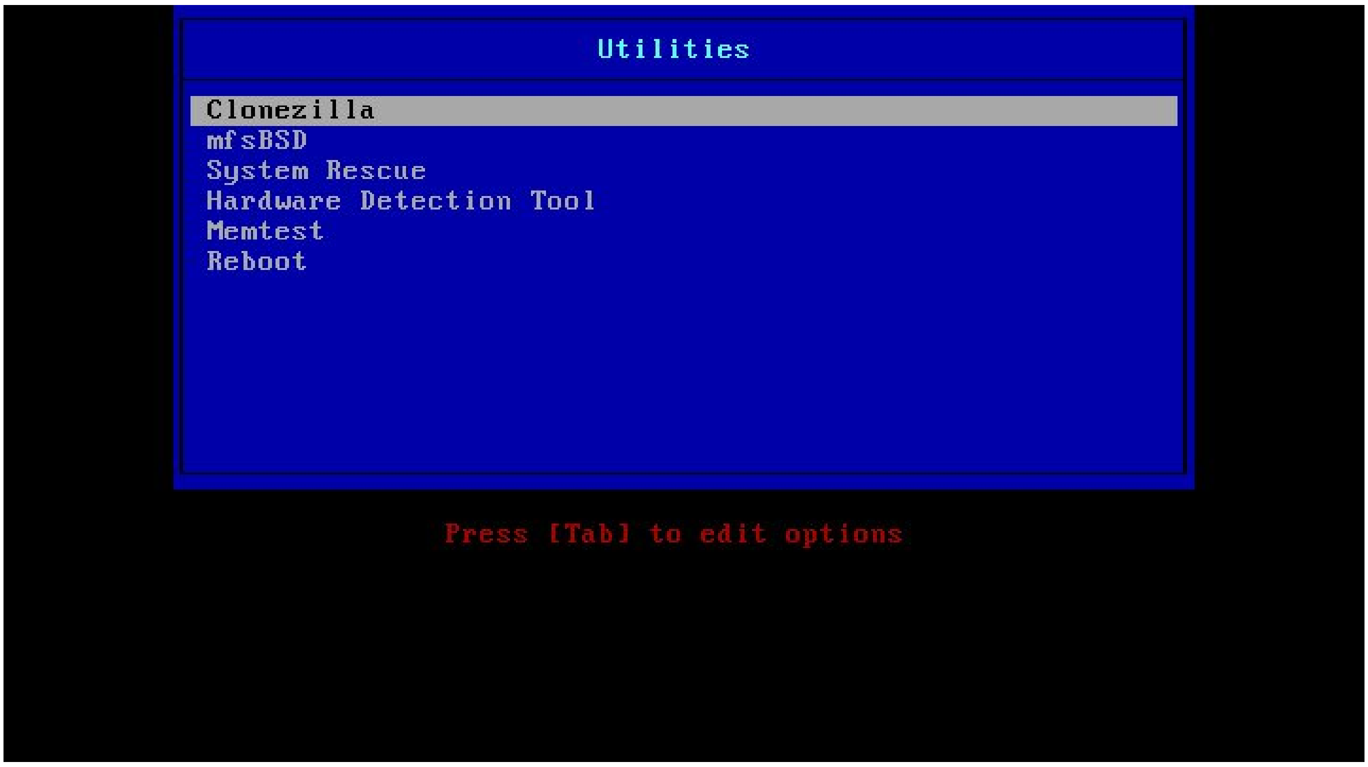
A menu option to reboot the target computer uses the *reboot.c32* module:

```
label reboot
  menu label Reboot
  kernel reboot.c32
```

13 The Complete Text Menu

/usr/tftpboot/pxelinux.cfg/default

```
ui menu.c32
menu title Utilities

label clonezilla
  menu label Clonezilla
  kernel images/clonezilla/live/vmlinuz
  append boot=live live-config noswap nolocales edd=on nomodeset \
  ocs_live_run="ocs-live-general" ocs_live_extra_param="" \
  keyboard-layouts=us locales=en_US.UTF-8 \
  ocs_live_batch="no" vga=788 nosplash fetch=tftp://192.168.1.1/images/clonezilla/live/ ↩
      filesystem.squashfs
  initrd http://192.168.1.1/images/clonezilla/live/initrd.img

label mfsbsd
  menu label mfsBSD
  kernel memdisk
  initrd images/mfsbsd/mfsbsd.gz raw

label systemrescue
  menu label System Rescue
  kernel images/systemrescue/rescuecd
  append setkmap=us netboot=tftp://192.168.1.1:/images/systemrescue/sysrcd.dat
  initrd images/systemrescue/initram.igz

label hdt
  menu label Hardware Detection Tool
  kernel hdt.c32
  append pciids=images/hdt/pci.ids.gz modules_pcimap=images/hdt/modules.pcimap.gz

label memtest
  menu label Memtest
  kernel images/memtest/memtestp

label reboot
  menu label Reboot
  kernel reboot.c32
```



14 Virtual Machines

VirtualBox (*/usr/ports/emulators/virtualbox-ose*) is the best-supported virtual machine host for FreeBSD.

To PXE-boot virtual machines, enable the host's VirtualBox network module in */etc/rc.conf* and start the script:

```
vboxnet_enable="YES"
```

```
# service vboxnet start
```

Set the guest's network interface to *Bridged Adapter* and select the *PCnet-PCI II* emulated adapter; as of VirtualBox 4.1.8, this is the only emulated adapter that supports PXE booting. Start the VM, press **F12** for the boot menu, then press **L** to boot from the LAN.

15 Advanced Options

15.1 Faster Transfers With HTTP

TFTP is not particularly fast when transferring the 200M or larger compressed filesystems used by some bootable utilities. Newer utilities can often retrieve their filesystem images by HTTP, which is much faster. One target system took three minutes to load the *sysrcd.dat* file by TFTP. With HTTP, it took only 18 seconds.

Installation and basic configuration of Apache 22 is, as they say, "left as an exercise for the reader".

Add a link in your *DocumentRoot* directory to the TFTP root directory:

```
# ln -s /usr/tftpboot/images/ /usr/local/www/apache22/data/
```

/usr/local/etc/apache22/httpd.conf is configured so image directory access is only allowed to specific portions of the network:

```
<Directory "/images">
    Order Deny,Allow
    Deny from all
    Allow from 192.168.1.0/24
    Options Indexes
</Directory>
```

The menu entries in */usr/tftpboot/pxelinux.cfg/default* are then changed to use HTTP. With the *images* directory linked, paths will be the same as for TFTP. *gpxelinux.0* can even load the kernel and initrd files with HTTP.

```
label clonezilla
  menu label Clonezilla
  kernel images/clonezilla/live/vmlinuz
  append boot=live live-config noswap nolocales edd=on nomodeset \
  ocs_live_run="ocs-live-general" ocs_live_extra_param="" \
  keyboard-layouts=us locales=en_US.UTF-8 \
  ocs_live_batch="no" vga=788 nosplash fetch=tftp://192.168.1.1images/clonezilla/live/ ↩
      filesyste.squashfs
  initrd http://192.168.1.1/images/clonezilla/live/initrd.img

label systemrescue
  menu label System Rescue
  kernel http://192.168.1.1/images/systemrescue/rescuecd
  append setkmap=us netboot=http://192.168.1.1/images/systemrescue/sysrcd.dat
  initrd http://192.168.1.1/images/systemrescue/initram.igz
```

See gPXE for more information on gPXE's capabilities.

15.2 NFS: Network File System

Memory-resident file systems can be limiting. At best, they steal memory from the operating system. At worst, a RAM filesystem may be difficult to create or not available at all.

NFS allows remote filesystems to be mounted over the network, without having to transfer a large data file or lose a large chunk of RAM on the target computer.

/etc/exports lists the directories to be network-mounted along with access limits and permissions. Export the */usr/tftpboot* directory:

```
/usr/tftpboot -alldirs,ro -mapall=nobody:nobody -network 192.168.1.0/24
```

Modify */etc/hosts.allow* to allow NFS access to the internal network by adding a line just before the existing *rpcbind : ALL : deny* line:

```
rpcbind : 192.168.1.0/24 : allow
```

NFS server programs are enabled in */etc/rc.conf*:

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

The NFS server will start on reboot. For now, start it manually:

```
# /etc/rc.d/rpcbind start
# /etc/rc.d/nfsd start
```

15.2.1 FreeBSD With NFS

The FreeBSD "livefs" image is a working FreeBSD system that can be used without installation on the target machine. An NFS-mounted filesystem can replace the CD or memory stick media for faster operation.

Download the livefs memory stick image from ftp://ftp.freebsd.org/pub/FreeBSD/ISO-IMAGES-i386/8.2/ and copy the entire directory structure to a subdirectory in the TFTP *images* directory:

```
# mount /dev/`mdconfig -f FreeBSD-8.2-RELEASE-i386-memstick.img`a /mnt   ❶
# rsync -aH /mnt/ /usr/tftpboot/images/freebsd/
# umount /mnt
# mdconfig -d -u 0
```

❶ Note the "a" after that second backtick. `mdconfig` will return *md0* when it creates the device. The UFS file system on that device is in the *md0a* partition.

Modify */etc/gettytab* to autologin as root, and set the local DNS server:

```
# sed -i -e 's/:ht:np:/:al=root:ht:np:/' /usr/tftpboot/images/freebsd/etc/gettytab
# echo 'nameserver 192.168.1.1' > /usr/tftpboot/images/freebsd/etc/resolv.conf
```

This leaves a fairly large FreeBSD directory, about 849M. If the installation or package directories won't be used, 600M of space can be reclaimed by deleting them:

```
# rm -rf /usr/tftpboot/images/freebsd/8.2-RELEASE
# rm -rf /usr/tftpboot/images/freebsd/packages
```

FreeBSD's *pxeboot(8)* loader looks to the DHCP server for a *root-path* that identifies the NFS file system root, so modify *dhcpd.conf* again:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
        range 192.168.1.100 192.168.1.200;
        next-server 192.168.1.1; # TFTP server address
        filename "gpxelinux.0";  # PXE boot loader filename
        option root-path "/usr/tftpboot/images/freebsd";  # NFS root
}
```

Warning
Anything not running FreeBSD may be surprised, or more likely appalled, to find a FreeBSD file tree NFS-mounted where it was least expected. Fortunately many of the Linux systems allow a manual NFS root to be passed to the kernel.

Restart the DHCP server after changing settings:

```
# service isc-dhcpd restart
```

The menu entry just PXE-boots the FreeBSD *pxeboot* loader:

```
label freebsd
  menu label FreeBSD LiveFS
  pxe images/freebsd/boot/pxeboot
```

15.2.2 XUbuntu With NFS

Setting up XUbuntu to use an NFS filesystem is similar to FreeBSD. NFS mounting the 643M *filesystem.squashfs* leaves it on the NFS server instead of copying the whole thing into memory on the target computer.

Note
Why XUbuntu? *Because **shut up**, that's why*!

Download the ISO image from http://xubuntu.org/ and copy the directory tree to a subdir in *images*:

```
# mkdir /usr/tftpboot/images/xubuntu
# mount -t cd9660 /dev/`mdconfig -f xubuntu-11.04-desktop-i386.iso` /mnt
# rsync -a /mnt/ /usr/tftpboot/images/xubuntu/
# umount /mnt
# mdconfig -d -u 0
```

The menu entry sets the *netboot* and *nfsroot* options:

```
label xubuntu
  menu label XUbuntu
  kernel images/xubuntu/casper/vmlinuz
  append boot=casper netboot=nfs nfsroot=192.168.1.1:/usr/tftpboot/images/xubuntu
  initrd images/xubuntu/casper/initrd.lz
```

Tip
Usefulness of NFS varies depending on a particular target computer's Ethernet interface. For example, NFS-booting FreeBSD or XUbuntu fails on an Acer Aspire One D250, apparently because of driver problems with the oddball Atheros wired Ethernet controller on that system. Other computers with better-supported Ethernet boot from the same NFS server without complaint.

16 Graphic Menus

The VESA menu module displays menus in graphics mode, and allows background graphics. Modify the *ui* command at the start of the menu to load *vesamenu.c32* and add a *menu background* command to load a background picture. The picture is PNG, 640x480, 8-bit color. Samples of the various menus can be seen at http://syslinux.zytor.com/wiki/index.php/Screenshots.

```
ui vesamenu.c32
menu title Utilities
menu background backgnd.png
...
```

17 Conclusion

Trying to find specific examples while setting up PXE booting was frustrating for me. The ones that were out there generally only covered the simplest situations. Hopefully the examples and specifics here will reduce some of that frustration for others.

A Complete VESA Graphic Menu

/usr/tftpboot/pxelinux.cfg/default

```
ui vesamenu.c32
menu title Utilities
menu background backgnd.png

label clonezilla
  menu label Clonezilla
  kernel images/clonezilla/vmlinuz
  append boot=live live-config union=aufs noswap vga=788 fetch=tftp://192.168.1$
  initrd images/clonezilla/initrd.img

label mfsbsd
  menu label mfsBSD
  kernel memdisk
  initrd images/mfsbsd/mfsbsd.gz raw

label systemrescue
  menu label System Rescue
  kernel images/systemrescue/rescuecd
  append setkmap=us netboot=tftp://192.168.1.1:/images/systemrescue/sysrcd.dat
  initrd images/systemrescue/initram.igz

label hdt
  menu label Hardware Detection Tool
  kernel hdt.c32
  append pciids=images/hdt/pci.ids.gz modules_pcimap=images/hdt/modules.pcimap.gz

label memtest
  menu label Memtest
  kernel images/memtest/memtestp

label freebsd
  menu label FreeBSD LiveFS
  pxe images/freebsd/boot/pxeboot

label xubuntu
  menu label XUbuntu
  kernel images/xubuntu/casper/vmlinuz
  append boot=casper netboot=nfs nfsroot=192.168.1.1:/usr/tftpboot/images/xubuntu
  initrd images/xubuntu/casper/initrd.lz

label reboot
  menu label Reboot
  kernel reboot.c32
```

B Preprocessing Menus With `cpp`

Some convenience features would make the menu file easier to maintain. Being able to define a constant for the HTTP server, for instance, or breaking up `append` lines so they are easier to edit.

Using `cpp` and `make`, it's not too hard to add these features. The *Makefile*:

/usr/tftpboot/pxelinux.cfg/Makefile

```
# Makefile for preprocessing gpxelinux.0 menu config files
# WB 2010-07-14

CPP=            /usr/bin/cpp
CFGNAME=        default
PRINTF=         /usr/bin/printf
```

```
RM=             /bin/rm
SED=            /usr/bin/sed
SRCNAME=        menu

TAB!=           ${PRINTF} "\t"

${CFGNAME}:     ${SRCNAME}.src
        ${CPP} -traditional-cpp ${SRCNAME}.src \                ❶
        | ${SED} -e '/^[ ${TAB}]*#/d' -e '/^$$/d' > ${CFGNAME}
```

❶ *-traditional-cpp* is to keep `cpp` from stripping out everything following "//" as a comment (and thus breaking URLs).

The menu source file:

/usr/tftpboot/pxelinux.cfg/menu.src

```
#define SERVER  192.168.1.1             ❶
#define HTTPURL http://SERVER
#define NFSROOT SERVER:/usr/tftpboot
#define NFSURL  nfs://NFSROOT

ui vesamenu.c32
menu title Utilities
menu background backgnd.png

 # clonezilla                           ❷
label clonezilla
#define CLZDIR HTTPURL/images/clonezilla  ❸
  menu label Clonezilla
  kernel CLZDIR/vmlinuz
  append boot=live live-config union=aufs noswap vga=788 \
  ocs_lang="en_US.UTF-8" \
  ocs_live_keymap="/usr/share/keymaps/i386/qwerty/us.kmap.gz" \
  fetch=CLZDIR/filesystem.squashfs
  initrd CLZDIR/initrd.img

 # mfsBSD
label mfsbsd
  menu label mfsBSD
  kernel memdisk
  initrd HTTPURL/images/mfsbsd/mfsbsd.gz raw

 # systemrescue
label systemrescue
#define SYSRSDIR HTTPURL/images/systemrescue
  menu label System Rescue
  kernel SYRSDIR/rescuecd
  append setkmap=us SYSRSDIR/sysrcd.dat
  initrd SYSRSDIR/initram.igz

label hdt
  menu label Hardware Detection Tool
  kernel hdt.c32
  append pciids=images/hdt/pci.ids.gz \
  modules_pcimap=images/hdt/modules.pcimap.gz

label memtest
  menu label Memtest
  kernel images/memtest/memtestp

 # xubuntu
```

```
label xubuntu
#define XUDIR HTTPURL/images/xubuntu/casper
  menu label XUbuntu
  kernel XUDIR/vmlinuz
  append boot=casper netboot=nfs \
  nfsroot=NFSROOT/images/xubuntu
  initrd XUDIR/initrd.lz

label reboot
  menu label Reboot
  kernel reboot.c32
```

1. The server address is defined, then the HTTP URL is defined based on the server address. NFS constants are also defined, and include */usr/tftpboot* so all of the paths in the menu entries are consistent.
2. The "#" comments are indented with spaces or tabs so they aren't interpreted as `cpp` directives.
3. Define the path to an image with *#define* to avoid needless repetition.

Building *default* from *menu.src* is just

```
# make
```