

PXE Magic: Flexible Network Booting with Menus

By Kyle Rankin

(Reprinted from the Linux Journal)

It's funny how automation evolves as system administrators manage larger numbers of servers. When you manage only a few servers, it's fine to pop in an install CD and set options manually. As the number of servers grows, you might realize it makes sense to set up a kickstart or FAI (Debian's Fully Automated Installer) environment to automate all that manual configuration at install time. Now, you boot the install CD, type in a few boot arguments to point the machine to the kickstart server, and go get a cup of coffee as the machine installs.

When the day comes that you have to install three or four machines at once, you either can burn extra CDs or investigate PXE boot. The Preboot eXecution Environment is an open standard developed by Intel to allow machines to boot over a network instead of from local media, such as a floppy, CD or hard drive. Modern servers and newer laptops and desktops with integrated NICs should support PXE booting in the BIOS—in some cases, it's enabled by default, and in other cases, you need to go into your BIOS settings to enable it.

Because many modern servers these days offer built-in remote power and remote terminals or otherwise are remotely accessible via serial console servers or networked KVM, if you have a PXE boot environment set up, you can power on remotely, then boot and install a machine from miles away.

If you have never set up a PXE boot server before, the first part of this article covers the steps to get your first PXE server up and running. If PXE booting is old hat to you, skip ahead to the section called PXE Menu Magic. There, I cover how to configure boot menus when you PXE boot, so instead of hunting down MAC addresses and doing a lot of setup before an install, you simply can boot, select your OS, and you are off and running. After that, I discuss how to integrate rescue tools, such as Knoppix and memtest86+, into your PXE environment, so they are available to any machine that can boot from the network.

PXE Setup

You need three main pieces of infrastructure for a PXE setup: a DHCP server, a TFTP server and the syslinux software. Both DHCP and TFTP can reside on the same server. When a system attempts to boot from the network, the DHCP server gives it an IP address and then tells it the address for the TFTP server and the name of the bootstrap program to run. The TFTP server then serves that file, which in our case is a PXE-enabled syslinux binary. That program runs on the booted machine and then can load Linux kernels or other OS files that also are shared on the TFTP server over the network. Once the kernel is loaded, the OS starts as normal, and if you have configured a kickstart install correctly, the install begins.

Configure DHCP

Any relatively new DHCP server will support PXE booting, so if you don't already have a DHCP server set up, just use your distribution's DHCP server package (possibly named dhcpd, dhcp3-server or something similar). Configuring DHCP to suit your network is somewhat beyond the scope of this article, but many distributions ship a default configuration file that should provide a good place to start. Once the DHCP server is installed, edit the configuration file (often in /etc/dhcpd.conf), and locate the subnet section (or each host section if you configured static IP assignment via DHCP and want these hosts to PXE boot), and add two lines:

PXE Magic: Flexible Network Booting with Menus

By Kyle Rankin

(Reprinted from the Linux Journal)

```
next-server ip_of_pxe_server;  
filename "pxelinux.0";
```

The next-server directive tells the host the IP address of the TFTP server, and the filename directive tells it which file to download and execute from that server. Change the next-server argument to match the IP address of your TFTP server, and keep filename set to pxelinux.0, as that is the name of the syslinux PXE-enabled executable.

In the subnet section, you also need to add dynamic-bootp to the range directive. Here is an example subnet section after the changes:

```
subnet 10.0.0.0 netmask 255.255.255.0 {  
    range dynamic-bootp 10.0.0.200 10.0.0.220;  
    next-server 10.0.0.1;  
    filename "pxelinux.0";  
}
```

Install TFTP

After the DHCP server is configured and running, you are ready to install TFTP. The pxelinux executable requires a TFTP server that supports the tsize option, and two good choices are either tftpd-hpa or atftp. In many distributions, these options already are packaged under these names, so just install your distribution's package or otherwise follow the installation instructions from the project's official site.

Depending on your TFTP package, you might need to add an entry to /etc/inetd.conf if it wasn't already added for you:

```
tftp dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s var/lib/tftpboot
```

As you can see in this example, the -s option (used for tftpd-hpa) specified /var/lib/tftpboot as the directory to contain my files, but on some systems, these files are commonly stored in /tftpboot, so see your /etc/inetd.conf file and your tftpd man page and check on its conventions if you are unsure. If your distribution uses xinetd and doesn't create a file in /etc/xinetd.d for you, create a file called /etc/xinetd.d/tftp that contains the following:

```
# default: off  
# description: The tftp server serves files using  
# the trivial file transfer protocol.  
# The tftp protocol is often used to boot diskless  
# workstations, download configuration files to network-aware  
# printers, and to start the installation process for  
# some operating systems.  
service tftp  
{  
    Disable          = no  
    socket_type      = dgram
```

PXE Magic: Flexible Network Booting with Menus

By Kyle Rankin

(Reprinted from the Linux Journal)

```
protocol      = udp
wait          = yes
user          = root
server        = /usr/sbin/in.tftpd
server_args   = -s /var/lib/tftpboot
per_source    = 11
cps           = 100 2
flags         = IPv4
}
```

As tftpd is part of inetd or xinetd, you will not need to start any service. At most, you might need to reload inetd or xinetd; however, make sure that any software firewall you have running allows the TFTP port (port 69 udp) as input.

Add Syslinux

Now that TFTP is set up, all that is left to do is to install the syslinux package (available for most distributions, or you can follow the installation instructions from the project's main Web page), copy the supplied pxelinux.0 file to /var/lib/tftpboot (or your TFTP directory), and then create a /var/lib/tftpboot/pxelinux.cfg directory to hold pxelinux configuration files.

PXE Menu Magic

You can configure pxelinux with or without menus, and many administrators use pxelinux without them. There are compelling reasons to use pxelinux menus, which I discuss below, but first, here's how some pxelinux setups are configured.

When many people configure pxelinux, they create configuration files for a machine or class of machines based on the fact that when pxelinux loads it searches the pxelinux.cfg directory on the TFTP server for configuration files in the following order:

Files named 01-MACADDRESS with hyphens in between each hex pair. So, for a server with a MAC address of 88:99:AA:BB:CC:DD, a configuration file that would target only that machine would be named 01-88-99-aa-bb-cc-dd (and I've noticed it does matter that it is lowercase).

Files named after the host's IP address in hex. Here, pxelinux will drop a digit from the end of the hex IP and try again as each file search fails. This is often used when an administrator buys a lot of the same brand of machine, which often will have very similar MAC addresses. The administrator then can configure DHCP to assign a certain IP range to those MAC addresses. Then, a boot option can be applied to all of that group.

Finally, if no specific files can be found, pxelinux will look for a file named default and use it.

One nice feature of pxelinux is that it uses the same syntax as syslinux, so porting over a configuration from a CD, for instance, can start with the syslinux options and follow with your custom network options. Here is an example configuration for an old CentOS 3.6 kickstart:

```
default linux
```

PXE Magic: Flexible Network Booting with Menus

By Kyle Rankin

(Reprinted from the Linux Journal)

```
label linux
kernel vmlinuz-centos-3.6
append text nofb load_ramdisk=1 initrd=initrd-centos-3.6.img
network ks=http://10.0.0.1/kickstart/centos3.cfg
```

Why Use Menus?

The standard sort of pxelinux setup works fine, and many administrators use it, but one of the annoying aspects of it is that even if you know you want to install, say, CentOS 3.6 on a server, you first have to get the MAC address. So, you either go to the machine and find a sticker that lists the MAC address, boot the machine into the BIOS to read the MAC, or let it get a lease on the network. Then, you need to create either a custom configuration file for that host's MAC or make sure its MAC is part of a group you already have configured. Depending on your infrastructure, this step can add substantial time to each server. Even if you buy servers in batches and group in IP ranges, what happens if you want to install a different OS on one of the servers? You then have to go through the additional work of tracking down the MAC to set up an exclusion.

With pxelinux menus, I can preconfigure any of the different network boot scenarios I need and assign a number to them. Then, when a machine boots, I get an ASCII menu I can customize that lists all of these options and their number. Then, I can select the option I want, press Enter, and the install is off and running. Beyond that, now I have the option of adding non-kickstart images and can make them available to all of my servers, not just certain groups. With this feature, you can make rescue tools like Knoppix and memtest86+ available to any machine on the network that can PXE boot. You even can set a timeout, like with boot CDs, that will select a default option. I use this to select my standard Knoppix rescue mode after 30 seconds.

Configure PXE Menus

Because pxelinux shares the syntax of syslinux, if you have any CDs that have fancy syslinux menus, you can refer to them for examples. Because you want to make this available to all hosts, move any more specific configuration files out of pxelinux.cfg, and create a file named default. When the pxelinux program fails to find any more specific files, it then will load this configuration. Here is a sample menu configuration with two options: the first boots Knoppix over the network, and the second boots a CentOS 4.5 kickstart:

```
default 1
timeout 300
prompt 1
display f1.msg
F1 f1.msg
F2 f2.msg

label 1
kernel vmlinuz-knx5.1.1
append secure nfsdir=10.0.0.1:/mnt/knoppix/5.1.1
nodhcp lang=us ramdisk_size=100000 init=/etc/init
2 apm=power-off nomce vga=normal
initrd=miniroot-knx5.1.1.gz quiet BOOT_IMAGE=knoppix

label 2
```

PXE Magic: Flexible Network Booting with Menus

By Kyle Rankin

(Reprinted from the Linux Journal)

```
kernel vmlinuz-centos-4.5-64
append text nofb ksdevice=eth0 load_ramdisk=1
initrd=initrd-centos-4.5-64.img network
ks=http://10.0.0.1/kickstart/centos4-64.cfg
```

Each of these options is documented in the syslinux man page, but I highlight a few here. The default option sets which label to boot when the timeout expires. The timeout is in tenths of a second, so in this example, the timeout is 30 seconds, after which it will boot using the options set under label 1. The display option lists a message if there are any to display by default, so if you want to display a fancy menu for these two options, you could create a file called f1.msg in /var/lib/tftpboot/ that contains something like:

```
----| Boot Options |-----
|                                     |
| 1. Knoppix 5.1.1                   |
| 2. CentOS 4.5 64 bit               |
|                                     |
-----
```

```
<F1> Main | <F2> Help
Default image will boot in 30 seconds...
```

Notice that I listed F1 and F2 in the menu. You can create multiple files that will be output to the screen when the user presses the function keys. This can be useful if you have more menu options than can fit on a single screen, or if you want to provide extra documentation at boot time (this is handy if you are like me and create custom boot arguments for your kickstart servers). In this example, I could create a /var/lib/tftpboot/f2.msg file and add a short help file.

Although this menu is rather basic, check out the syslinux configuration file and project page for examples of how to jazz it up with color and even custom graphics.

Extra Features: PXE Rescue Disk

One of my favorite features of a PXE server is the addition of a Knoppix rescue disk. Now, whenever I need to recover a machine, I don't need to hunt around for a disk, I can just boot the server off the network.

First, get a Knoppix disk. I use a Knoppix 5.1.1 CD for this example, but I've been successful with much older Knoppix CDs. Mount the CD-ROM, and then go to the boot/isolinux directory on the CD. Copy the miniroot.gz and vmlinuz files to your /var/lib/tftpboot directory, except rename them something distinct, such as miniroot-knx5.1.1.gz and vmlinuz-knx5.1.1, respectively. Now, edit your pxelinux.cfg/default file, and add lines like the one I used above in my example:

```
label 1
kernel vmlinuz-knx5.1.1
append secure nfsdir=10.0.0.1:/mnt/knoppix/5.1.1 nodhcp
lang=us ramdisk_size=100000 init=/etc/init 2
apm=power-off nomce vga=normal
```

PXE Magic: Flexible Network Booting with Menus

By Kyle Rankin

(Reprinted from the Linux Journal)

```
initrd=miniroot-knx5.1.1.gz quiet BOOT_IMAGE=knoppix
```

Notice here that I labeled it 1, so if you already have a label with that name, you need to decide which of the two to rename. Also notice that this example references the renamed vmlinuz-knx5.1.1 and miniroot-knx5.1.1.gz files. If you named your files something else, be sure to change the names here as well. Because I am mostly dealing with servers, I added 2 after init=/etc/init on the append line, so it would boot into runlevel 2 (console-only mode). If you want to boot to a full graphical environment, remove 2 from the append line.

The final step might be the largest for you if you don't have an NFS server set up. For Knoppix to boot over the network, you have to have its CD contents shared on an NFS server. NFS server configuration is beyond the scope of this article, but in my example, I set up an NFS share on 10.0.0.1 at /mnt/knoppix/5.1.1. I then mounted my Knoppix CD and copied the full contents to that directory. Alternatively, you could mount a Knoppix CD or ISO directly to that directory. When the Knoppix kernel boots, it will then mount that NFS share and access the rest of the files it needs directly over the network.

Extra Features: Memtest86+

Another nice addition to a PXE environment is the memtest86+ program. This program does a thorough scan of a system's RAM and reports any errors. These days, some distributions even install it by default and make it available during the boot process because it is so useful. Compared to Knoppix, it is very simple to add memtest86+ to your PXE server, because it runs from a single bootable file. First, install your distribution's memtest86+ package (most make it available), or otherwise download it from the memtest86+ site. Then, copy the program binary to /var/lib/tftpboot/memtest. Finally, add a new label to your pxelinux.cfg/default file:

```
label 3
    kernel memtest
```

That's it. When you type 3 at the boot prompt, the memtest86+ program loads over the network and starts the scan.

Conclusion

There are a number of extra features beyond the ones I give here. For instance, a number of DOS boot floppy images, such as Peter Nordahl's NT Password and Registry Editor Boot Disk, can be added to a PXE environment. My own use of the pxelinux menu helps me streamline server kickstarts and makes it simple to kickstart many servers all at the same time. At boot time, I can not only indicate which OS to load, but also more specific options, such as the type of server (Web, database and so forth) to install, what hostname to use, and other very specific tweaks. Besides the benefit of no longer tracking down MAC addresses, you also can create a nice colorful user-friendly boot menu that can be documented, so it's simpler for new administrators to pick up. Finally, I've been able to customize Knoppix disks so that they do very specific things at boot, such as perform load tests or even set up a Webcam server—all from the network.