PXE Booting

Preboot Execution Environment (PXE, pronounced *pixie*) is designed to allow a computer to boot from an image on its network. While it is generally used to boot diskless or low end workstations, it has several other applications. A nice discusion of it is found on Wikipedia Preboot_Execution_Environment.

As stated above, the “normal” way PXE is used it to boot small workstations through special purpose setups, like the Linux Terminal Services Project (LTSP, ltsp.org) or booting multiple servers from a standardized image (as used in some FreeBSD applications). It also appears to be an option for Microsoft products (see https://technet.microsoft.com/en-us/library/2008.07.desktopfiles.aspx).

Our need at Daily Data was different. One of our tasks is setting up servers and workstations for clients, meaning we have a lot of DVD's and USB Thumbdrives with installation images. The problems with this are media degradation and updating the installers as they are patched by the provider.

The initial solution to this was to set up a PXE server which would contain multiple images. We can then boot a new server via PXE, choose the installation, and do our work.

We chose to use a Debian Wheezy “server” (actually and old DE-2700) with a used SSD, since documentation (albeit older) was reliably available. This article describes the steps taken to build the server.

NOTE: the PXE server is not resource intensive. The machine we used has a dual core Atom processor and 1 Gigabyte of RAM, with a 30 Gigabyte SSD and two Gigabit network ports (we only used one). This is also an excellent candidate for a Virtual server.

While the following describes installation on Debian Wheezy (v 7.11), I'll try to explain the procedures so they can be adapted to any Linux, BSD, or other Unix service available. It is very possible this could be adapted to Microsoft products.

Debian server installation

The requirements for the server are very simple; a DHCP server and an FTP server. An NFS server is recommended (required?) for booting FreeBSD installation images, and having some form of package caching like Debian's apt-cacher (https://packages.debian.org/wheezy/apt-cacher) is also useful to cut down on Internet network usage. However, this document only describes the basics of DHCP and tftp server (tftpd) installation and configuration.

First, do the basic installation of a Debian Wheezy server to your liking. Then, execute the following commands which will install a nice DHCP and tftp server. One good reason for using tftpd-hpa is that it is run as a service, not from inetd, so you do not need the inetd service installed.

```
apt-get install isc-dhcp-server tftpd-hpa
```

DHCP Server

I wondered how I could have more than one DHCP sever on my network. How would they know which one to use. The key is the the configuration 'authoritive;' which is commented out by default in *isc-dhcp-server*. If this is not defined, the DHCP server will let assume someone else is in charge of assiging IP's, routes, and all the rest of the stuff we normally associate with DHCP servers. Thus, our DHCP server configuration is very minimal, but with some important additions.

Edit dhcp.conf (/etc/dhcp/dhcpd.conf on wheezy). Actually, I generally rename the original file to something like *dhcp.conf.original* and create a new one. The following example shows a base configuration which assumes you have another “real” dhcp server on the net (probably your firewall/router). All it contains is the following (change the IP's and domain names to match your “real” dhcp server).

```
ddns-update-style none;

option domain-name "example.com";
option domain-name-servers 192.168.0.1;

default-lease-time 600;
max-lease-time 7200;
log-facility local7;

subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.128 192.168.0.253;
  option broadcast-address 192.168.0.255;
  option routers 192.168.0.1;             # our router
  option domain-name-servers 192.168.0.1; # our router, again
  filename "pxelinux.0";
}

group {
  next-server 192.168.0.5;                # our Server
  host tftpclient {
    filename "pxelinux.0";
  }
}
```

- 192.168.0.5 is the IP of the PXE server you are working on, ie the machine you are currently building.
- IF you are using this as your primary DHCP server, all you need do is add the *filename “pxelinux.0”;* clause in the subnet and the *group* block to your existing configuration.

restart the dhcp server

```
/etc/init.d/isc-dhcp-server restart
/etc/init.d/isc-dhcp-server status
```

tftp server

In the past, tftp was run from inetd, but tftpd-hpa is a stand alone. You can verify it is running with one of the following commands:

```
netstat -uap | grep tftp
```

Or

```
/etc/init.d/tftpd-hpa status
```

IF it is running from inetd, you will need the following line. DO NOT USE THIS ON NEWER SYSTEMS. Instead, try // /etc/init.d/tftpd-hpa start// and fix any errors.

```
tftp           dgram   udp     wait    root  /usr/sbin/in.tftpd
/usr/sbin/in.tftpd -s /srv/tftp
```

- Note the configuration file from tftpd-ha is /etc/default/tftpd-ha

On Debian, the default location for the files is /srv/tftp. Let's put a file in there and see if tfptd is working correctly. Use any file, but the hostname file is pretty non-sensitive.

```
cp -av /etc/hostname /srv/tftp/
```

On any machine (can even be your server), run a tftp client and see if you can get that file. If you don't have a machine with a tftp client,

```
apt-get install tftp-ha
```

Now, run the client with the name/ip address of the server (localhost if you're on the same one)

```
cd /tmp
tftp serverIP
tftp> get hostname
tftp> quit
cat hostname
```

You should see the contents of the file; if it doesn't work, go back and figure out what is wrong.

Modify your primary DHCP Server

Obviously, if this is your primary DHCP server, there is no need. However, if you have a different one, you'll need to modify its configuration so it knows how to find your PXE server. Basically, you set the “next-server” to point to your PXE server and set the name of the image file.

Following is the procedures for opnSense

1. Open administrative interface to opnSense

2. DCHP | Server
3. TFTP Server | Advanced - IP address of PXE Server
4. Enable network booting | Advanced - Enter IP and *pxelinux.0* for default bios filename

- pxelinux.0 is a standard filename used in PXE booting. There is nothing special about the name, from what I can tell, but it is the “normal” way to name things, so we'll use it.

Verifying Installation

Get an install image

Now, let's get one image and see if it all will work. We'll be modifying it later, but initially, we'll just have one Debian Wheezy installer and see if we can boot that. BE CAREFUL; this is an installer, remember? So you can wipe out the test box.

Download one of the Debian PXE Installer images. They are called netboot in Debian. I have ftp.us.debian.org as the mirror I get stuff from; change that to whatever mirror is closer to you. Hint: *cat /etc/apt/sources.list* to find what your server is.

The following example downloads the amd64 version of Debian Wheezy.

NOTE: I am creating a separate directory in /tmp for this download. That way, later, if we want to redo this step, we do not have to get a fresh copy; it will use the one we have already downloaded so we don't use up more of the Debian download server than we need. Believe it or not, the complexity is warranted as you'll see later.

Also, the script will automatically clean up the tftp directory (first line in script). I did **not** comment this as some versions of *sh* use different commenting styles than bash.

getwheezy64

```
#! /usr/bin/env sh

DOWNLOADDIR=/tmp/debian/wheezy/amd64

# remove EVERYTHING in tftp server root
rm -fR /srv/tftp/*

# only execute this if file has not been downloaded
if [ ! -e $DOWNLOADDIR ]
then
   mkdir -p $DOWNLOADDIR
   cd $DOWNLOADDIR
   wget ftp://ftp.us.debian.org/debian/dists/wheezy/main/installer-
amd64/current/images/netboot/netboot.tar.gz
fi
cd /srv/tftp
tar -xzvf $DOWNLOADDIR/netboot.tar.gz
```

Following is a list of some of the netboot images for Debian

- ftp://ftp.us.debian.org/debian/dists/wheezy/main/installer-amd64/current/images/netboot/netboot.tar.gz
- ftp://ftp.us.debian.org/debian/dists/wheezy/main/installer-i386/current/images/netboot/netboot.tar.gz
- ftp://ftp.us.debian.org/debian/dists/jessie/main/installer-amd64/current/images/netboot/netboot.tar.gz
- ftp://ftp.us.debian.org/debian/dists/jessie/main/installer-i386/current/images/netboot/netboot.tar.gz

This will create two directories and two files in your /srv/tftp directory:

- pxelinux.0 - the boot image for pxe booting
- version - A debian file which gives information about the version you have installed
- pexlinux.conf - A directory which contains one file (*default*) which is the menu used for booting
- debian-installer - a directory containing all the files necessary to complete the boot process.

NOTE: Debian, as usual, has a case of the *cutes* in that they use a lot of symbolic links and include files. For example, pxelinux.0 is actually a symbolic link to debian-installer/pxelinux.0. For this test, we can ignore them.

Test the booting

On a separate machine which has PXE boot capabilities in the network card, boot into PXE mode. It will take a little while (maybe a minute?). On the machines I tested, I saw my MAC address, the address of the DHCP server, then the address of the PXE server, then, the installer took over the screen.

If you see the installer, you're done. You can go ahead and test the install, or you can just turn off the machine. In most cases, turning off doesn't hurt anything, because all it did was load stuff into memory.

Configure for multiple images

Basics

To boot multiple images, we need to know what is actually happening. Note: this is my understanding, and is limited by the research I've done, so don't take this as gospel. If you have better ideas, go to http://dailydata.net/about-us/contact-us/ and drop me a line.

The PXE loader in the Network Card asks the DHCP server where to go to find a PXE image, which it then downloads via tftp. The image in this case is pxelinux.0. This file is then executed in memory. It downloads pxelinux.cfg/default, and executes the default stanza in it.

NOTE: this is all done in memory on your client (the one you are installing to), so we can't get too carried away with large images or anything.

In the case of our test above, it loaded the linux kernel with some parameters (including an initrd).

From this we can see we need two things to make this work, both stored in the root of the tftp server's directory. pxelinux.0 and pxelinux.cfg/default.

The image we used for our test above is really not that great. Debian assumes this will be the only boot image, so they use a lot of includes and symbolic links.

First, I want a different directory structure so I could tell what was what.

Currently, the directory looks like this:

```
tftp_root
   |
   \ debian-installer
       |
       \- i386
```

But, I would like to see something I can rapidly get to, since I plan to add/remove offerings. Falko at HowtoForge https://www.howtoforge.com/setting-up-a-pxe-install-server-for-multiple-linux-distributions-on-debian-lenny-p2 already laid it out, so I'll just steal what he did, with a few modifications:

```
     tftp_root
             |
             +freebsd
             |     |
             |     +10
             |     |  |
             |     |  +i386
             |     |  |
             |     |  +x86_64
             |     +11
             |        |
             |        +i386
             |        |
             |        +x86_64
             |
             +debian
             |     |
             |     +wheezy
             |     |    |
             |     |    +i386
             |     |    |
             |     |    +x86_64
             |     +jessie
             |          |
             |          +i386
             |          |
             |          +x86_64
             |
             |
             |
```

The procedure is fairly simple.

1. Download the image into a temporary directory and unpack it
2. Determine the subdirectory needed for the image and move it into our tftp tree
3. Create one or more entries in tftp_root/pxelinux.cfg/default
4. Add an entry to boot.txt

First, let's clean up the tftp server directory, then re-add the Debian Wheezy 64 package, but in the directory structure above.

The only things we really need are pxelinux.0 and pxelinux.cfg/default, both in the root of the tftp server. The following script basically grabs the pxelinux.0 file and moves it into the root directory, then deletes everything else. Then, it creates the directory pxelinux.cfg and puts an empty file in there.

cleanup

```
#! /usr/bin/env sh

mv /srv/tftp/debian-installer/amd64/pxelinux.0 /tmp
rm -fR /srv/tftp/*
mv /tmp/pxelinux.0 /srv/tftp/
mkdir /srv/tftp/pxelinux.cfg
```

Now, a modified getwheezy64 shell script. Replace the one you already have with this one and run it.

getwheezy64

```
#! /usr/bin/env sh

DOWNLOADDIR=/tmp/debian/wheezy/amd64
TFTPDIR=/srv/tftp/debian/wheezy
PACKAGEDIR=amd64
TEMPDIR=/tmp/working

if [ -e $TEMPDIR ]
then
   rm -fR $TEMPDIR
fi

if [ -e $TFTPDIR/$PACKAGEDIR ]
then
   rm -fR $TFTPDIR/$PACKAGEDIR
fi

mkdir -p $TFTPDIR

# only execute this if file has not been downloaded
if [ ! -e $DOWNLOADDIR/netboot.tar.gz ]
then
```

```
   mkdir -p $DOWNLOADDIR
   cd $DOWNLOADDIR
   wget ftp://ftp.us.debian.org/debian/dists/wheezy/main/installer-
amd64/current/images/netboot/netboot.tar.gz
fi

mkdir -p $TEMPDIR
cd $TEMPDIR
tar -xzvf $DOWNLOADDIR/netboot.tar.gz
mv debian-installer/$PACKAGEDIR $TFTPDIR
```

Run this script. It will rebuild the directory (using our already downloaded image package)

You have the files installed at this point, but you need the configuration file created in /srv/tftp/pxelinux.cfg/default

default

```
DISPLAY boot.txt

default install

label install
        menu label ^Install
        menu default
        kernel debian/wheezy/amd64/linux
        append vga=788 initrd=debian/wheezy/amd64/initrd.gz -- quiet


prompt 1
timeout 0
```

- The first line says “display the contents of the file boot.txt in the tftp root directory”. If we do not do that, the machine will hang, waiting for you to choose a configuration to execute.
- The line *default install* says “if I press the enter key, choose this configuration”
- The block of code beginning with *label install* determines what will be done when *install* is entered at the command line (or, the enter key is pressed on a blank line)
- *prompt 1* say “wait for the user to make a selection”. Without this, the default images is automatically selected.
- *timeout 0* says “wait forever for the user to enter a selection”

We only have one entry, so we'll create boot.txt with only one entry. I chose to put in the keyword (*install*) at the beginning, then a description of what this does. But, like I said, it is just a text file which is displayed on the screen.

boot.txt

```
PXE Boot Options. Choose One below
==================================
```

```
install -- Run the Debian Wheezy AMD64 Installer
```

Now, test by booting your test machine again using PXE. It should show the contents of boot.txt, then wait for you to either type 'install' or just press the Enter key (since it is the default).

Adding images

You can add other images. One of the simplest would be to use another script like getwheezy64, and modify it for installing the i386 package. All you would need to do is edit the file, changing amd64 for i386 (simple search and replace). After running it, you would need to create an additional entry in pxelinux.cfg/default and boot.txt.

The following code creates the getwheezyi386 script by passing getwheezy64 through sed, changing every instance of amd64 to i386. It assumes you are in the directory with getwheezy64. Run getwheezyi386 and you'll have a second install image.

<code bash> cat getwheezy64 | sed 's/amd64/i386/' > getwheezyi386

Now, change pxelinux.cfg/default to the following.

default

```
DISPLAY boot.txt

default install_i386

label install_amd64
        menu label ^Install Debian Wheezy AMD64
        kernel debian/wheezy/amd64/linux
        append vga=788 initrd=debian/wheezy/amd64/initrd.gz -- quiet

label install_i386
        menu label ^Install Debian Wheezy i386
        menu default
        kernel debian/wheezy/i386/linux
        append vga=788 initrd=debian/wheezy/i386/initrd.gz -- quiet


prompt 1
timeout 0
```

and add a line to boot.txt so the user knows what to do:

boot.txt

```
PXE Boot Options. Choose One below
==================================
```

```
install_amd64 -- Run the Debian Wheezy AMD64 Installer
install_i386  -- Run the Debian Wheezy i386 Installer
```

You'll note I took the simple 'install' label and modified it to show which installer we were going to do. Since the i386 will work on the amd64 chipset, but not visa versa, I chose that to be the default.

However, there is an easier way, if all you want to do is Debian. Read Debian PXE Installer Images for a faster, easier way of doing things, and also a better explanation of how Debian actually sets things up.

Questions

I still don't know exactly what is going on, so there are some questions I have. I'm putting them here so I can remember to research some more.

- why is the boot file called *pxelinux.0*? Does that imply a pxebsd.0, a pxewin32.0? If so, what are the differences.
- Debian does some really cool things with their menu. When you go into their base configuration, it is a full screen, almost curses screen with submenus. Is this something special for them, or do the developers just know things I haven't learned yet. Menus and submenus would be way cooler than what we have here.
- Where is the damned documentation on the whole tftpd pxe stuff?
- Do you really need a dhcp server on the tftp server, or can you just point the dhcp server to the other machine (ie, forget about next-server and just point tftp to the one we built?)
- Why does BSD use the NFS. I know it is because they want to mount an NFS share (for all the ports, etc...), but is that also the way “real” stuff should work?
- Many references to Windows doing this. How does it work?

Bibliography

- https://wiki.debian.org/PXEBootInstall
- https://www.howtoforge.com/setting-up-a-pxe-install-server-for-multiple-linux-distributions-on-debian-lenny
- https://forum.pfsense.org/index.php?topic=77516.0
- https://technet.microsoft.com/en-us/library/2008.07.desktopfiles.aspx

From:
https://kb.unixservertech.com/ - **Unix Server Tech Knowledge Base**

Permanent link:
https://kb.unixservertech.com/other/networking/pxebooting/start

Last update: **2017/05/02 01:17**