

# PXE: Not Just for Server Networks Anymore

Bill Childers

(Reprinted from the Linux Journal)

In the April 2008 issue of *Linux Journal*, Kyle Rankin's article "PXE Magic" explains how PXE (Preboot eXecution Environment) works and how you can install your own PXE server and integrate rescue tools like Knoppix along with a PXE-capable Kickstart installation. I've used much of Kyle's PXE Magic before (he and I worked together in a previous life), but recently I found myself managing not only a network of Linux servers, but also the entire LAN, encompassing Ubuntu laptops, desktops and servers, along with Windows laptops, desktops and servers. I found myself imagineering a PXE server that would not only allow me to kickstart servers and boot rescue tools off the network, but that also could provide a temporary environment for my users in the event that their computers broke. In my mind, the Holy Grail of this PXE server even would be able to install Windows machines via the network. After a fair amount of trial and error, I finally figured out the recipe, and in a strange twist, I was able to automate a network-based Windows installation...by bootstrapping Linux first.

## Setting Up an Ubuntu Terminal Server

I knew one of my goals for this system would be to give the users of my network an environment they could PXE boot to in a pinch—something that would appear familiar to them, as well as allow them the ability to perform basic tasks like check e-mail, surf the Web, instant message and so on. Luckily, much of our staff here runs Ubuntu on the desktop, so the decision to implement an Ubuntu Terminal Server using the Linux Terminal Server Project (LTSP) was a simple one.

Like any PXE implementation, the LTSP server requires a TFTP server, a properly configured DHCP server and the syslinux software. In a nutshell, the client boots; the PXE code in the network adapter runs; the machine gets a DHCP address and the address of a server to grab the syslinux code via TFTP; and then, it actually runs a TFTP client and downloads that code and executes it, starting the boot process. Thanks to the hard work of the Ubuntu LTSP maintainers, setting up the server was fast and easy.

There are two paths you can take to install an LTSP server: normal or standalone. A normal LTSP installation assumes you have a pre-existing DHCP server on your network, and a standalone LTSP install assumes no DHCP server, and it will install the DHCP infrastructure and integrate it with the LTSP server automatically. There already was a DHCP server on our corporate LAN, so I elected to do the normal LTSP installation and integrate it with our existing Microsoft Windows DHCP server.

I began the installation by installing a standard Ubuntu 8.04 desktop on a Dell 1950 server, as the LTSP server will have to act as a GNOME desktop for anyone who would be logging in to it. After that, I assigned the server a static IP on our LAN (on the same subnet as the desktops and laptops).

Installing the LTSP server was a piece of cake—a simple `sudo apt-get install ltsp-server openssh-server` at the GNOME terminal, and that task was complete. The final step on the LTSP server was to build the thin-client environment. Simply running `sudo ltsp-build-client` at the GNOME terminal fired off the remaining configuration steps and built the LTSP chroot.

Now that the LTSP server itself was ready, I had to enable our network for PXE booting, and this meant messing with the Windows DHCP server. It took a little bit of trial and error, but much like in the DHCP server config that Kyle mentions in his article, there were only two configuration options that needed to be added to the DHCP scope. In Microsoft-ese, these were Option "066 Boot Server Host

# PXE: Not Just for Server Networks Anymore

Bill Childers

(Reprinted from the Linux Journal)

Name”, which I set to the IP address assigned to the LTSP server and Option “067 Bootfile Name”, which I set to “ltsp/i386/pxelinux.0”. The last DHCP option seemed a little obscure, until I realized that the Ubuntu TFTP server's root directory was /var/lib/tftpboot. If you're running some other DHCP server, see the DHCP Notes sidebar, or refer to your DHCP server's documentation on adding options to the DHCP scope.

## DHCP Notes

I mentioned integrating the LTSP server with a Microsoft Windows DHCP server, but it's not difficult to get the server to work with other DHCP servers. If you are running the “standard” ISC dhcpd server, see Kyle's “PXE Magic” article (listed in Resources). He includes example configuration options along with excellent explanations as to how they work. If you are running dnsmasq (popular in OpenWRT and other embedded or lightweight Linux distributions), the dhcp-option=66,<ltsp\_ip\_address>and dhcp-option=pxe,67,pxelinux.0 in the dnsmasq.conf file should be what you need (I run this configuration at my home).

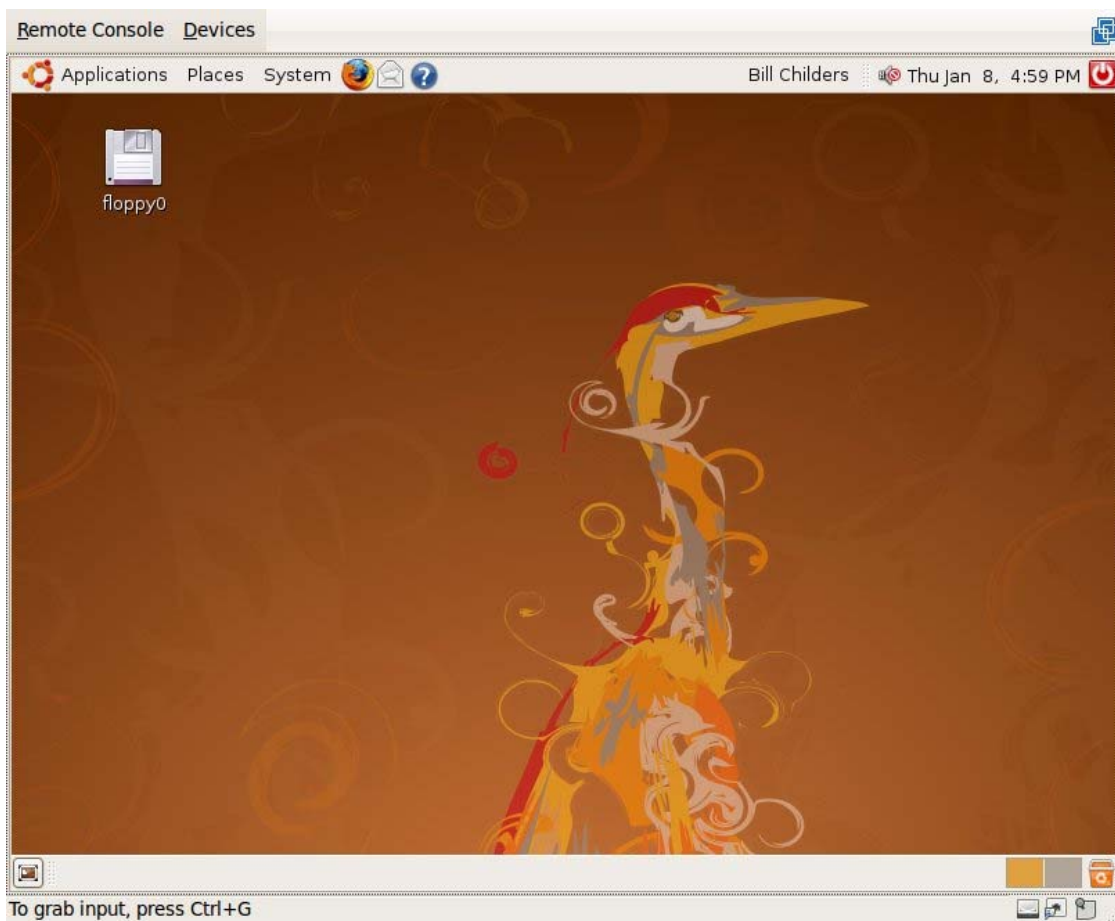


Figure 1. The Ubuntu LTSP GDM Login Screen

## PXE: Not Just for Server Networks Anymore

Bill Childers

(Reprinted from the Linux Journal)

At this point, I could boot a PC on our LAN, press F12, select Onboard NIC as the boot device, and in about 30 seconds, I got a GDM login screen! I could log in to an LTSP session at this point, but I had to do it as one of the users that already was on the Ubuntu server. It was close, but not quite what I wanted, as the ideal setup would allow anyone on our Windows domain to log in to an LTSP session. Fixing this would have meant integrating the server with our corporate Active Directory. That used to be a major chore unto itself, but with Ubuntu 8.04 and higher, it's just an apt-get and a couple commands away.

The package that makes all this magic happen is called likewise-open. First, I ran:

```
sudo apt-get install likewise-open
```

to get the likewise package. After that, I had to get the Ubuntu server to "join" the Windows domain. I did this by running:

```
sudo domainjoin-cli join <fqdn.mydomain.com> <DomainAdminUID>
```

I wanted likewise to run when the machine boots, so I issued a:

```
sudo update-rc.d likewise-open defaults
```

I also wanted the logins to be checked against the default domain, so I added the following line to the /etc/samba/lwiauthd.conf file:

```
winbind use default domain = yes
```

Finally, I started the likewise-open daemon using:

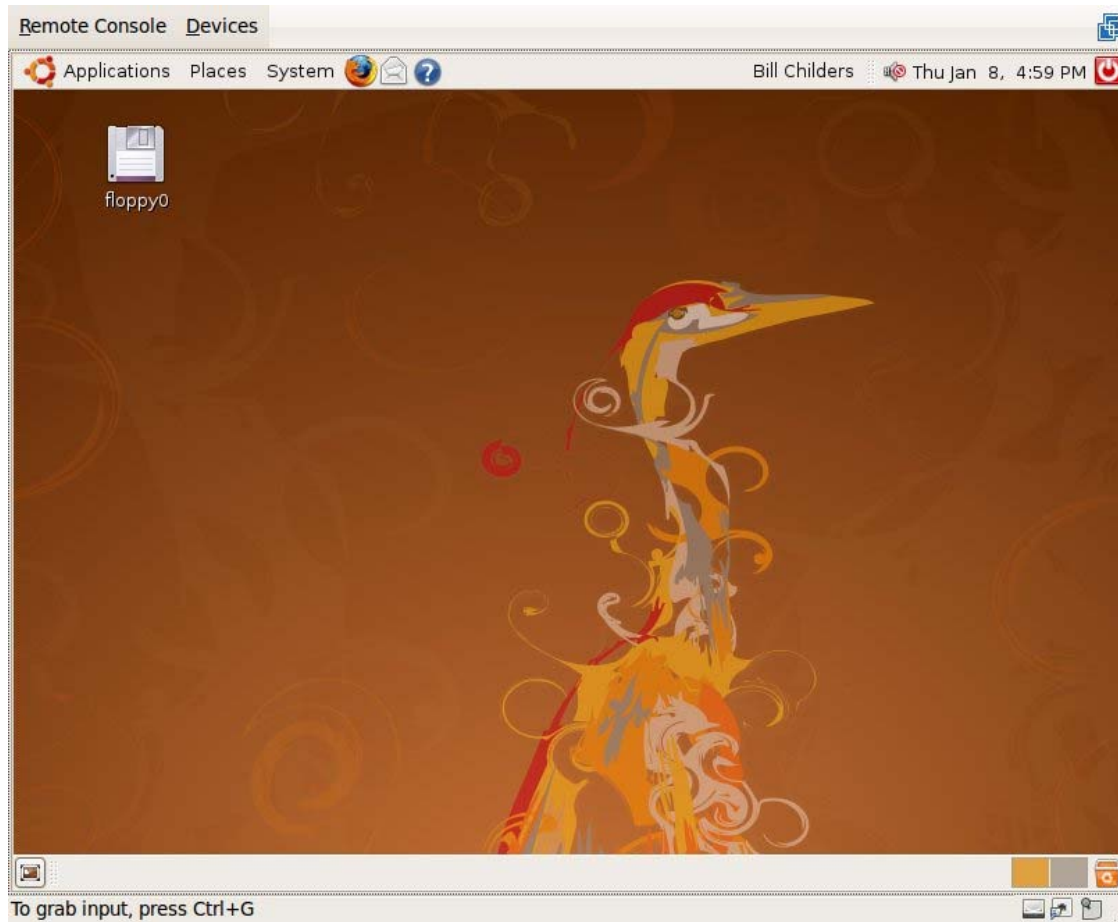
```
sudo /etc/init.d/likewise-open start
```

Now, my PXE LTSP clients could authenticate against the corporate Active Directory. Step one of the mission was complete!

# PXE: Not Just for Server Networks Anymore

Bill Childers

(Reprinted from the Linux Journal)



**Figure 2. The Ubuntu desktop—it's working!**

## Setting Up Unattended Windows Installs via PXE

The next step in my PXE adventures came when I was told I needed to refresh about 30 laptops with fresh builds of Windows. The method the previous Windows staff used to install Windows was through imaging the machine. Unfortunately, I could not locate the image files that had been used previously. Due to the issues and time constraints involved with trying to redevelop valid images for each hardware platform we had, I elected to do unattended installations of Windows.

I knew Windows included Remote Installation Service (RIS), but because I was in a time crunch, I was reluctant to learn a completely new technology. However, there was another option: Unattended, an open-source project. I'd found the Unattended project about a year earlier, and although I'd dabbled with it in my home lab, I'd never tried it in a corporate environment. Like many Linux administrators, I hear "Windows" and I cringe, but because I was tasked with this, I figured I'd do my best to make sense of the Windows install process, as well as get some repeatability and understanding out of it.

# PXE: Not Just for Server Networks Anymore

Bill Childers

(Reprinted from the Linux Journal)

Unattended relies on the fact that the first step of the Windows 2000/XP installer is essentially a DOS program. What happens when a machine is PXE booted to an Unattended install is a little convoluted, but it allows for great flexibility. Basically, the machine boots to a Linux kernel and shell, where some scripts provided by Unattended step in. The script partitions the system's disk and creates a basic FAT filesystem, and then it walks you through some menus where you can make choices as to the OS type (if you've set up Unattended with different Microsoft OS flavors), installation options and optional software you may have packaged. You're asked all the questions for a particular installation up front, including the CD Key, User Name, workgroup or domain to join, and administrative users. The Unattended scripts automatically digest all of this and create an unattend.txt file, which is dropped on the newly created FAT filesystem. Then, a FreeDOS session is started, and the Windows installer and OS bits are copied from a Samba share on the Unattended server, and then the installer is launched using the unattend.txt file. At this point, the installation is hands-off and proceeds without administrator intervention. The Unattended team has even gone so far as to create custom scripts that can install other pieces of software you may want to add to your configuration (automated VPN or Microsoft Office, for example).

Because there is no packaged install for Unattended, and the install process is quite different from the standard `./configure && make install`, I sat down for a bit and read the documentation on the site. Basically, the Unattended system leverages PXE and Linux as stated above, plus Samba for the distribution of the installation bits, and a bunch of Perl, shell and batch scripts to do a lot of the installation "magic".

The documentation asks that you have a working DHCP and DNS server, as well as a Samba server. I had the DHCP and DNS figured out for the LTSP server, so as per the step-by-step documentation, a `sudo apt-get install samba` got the Samba server installed. Next, I downloaded the Unattended distribution from the Web site and unpacked it in `/opt/unattended`. Then, I created a CNAME record on our DNS server that pointed `ntinstall` to the real hostname of the installation server. I then configured the Samba server with the following share information in `/etc/samba/smb.conf`:

```
[global]
...
guest account = guest
unix extensions = off
...
[install]
  comment = Unattended
  writable = no
  locking = no
  path = /path/to/unattended/install
  guest ok = yes
```

Finally, it was time to populate the OS distribution point with the Windows bits. This is done by creating a directory under the `<unattended root>/install/os` directory for whatever flavor of Windows you choose to install. In my case, I created an `/opt/unattended/install/os/winxp` directory and mounted that directory via Samba on my desktop. Then, I dropped the Windows installation media into the CD drive on my desktop and copied the `/i386` directory from the CD to the `/install/os/winxp` share on the

# PXE: Not Just for Server Networks Anymore

Bill Childers

(Reprinted from the Linux Journal)

server. Now my Unattended server was, in theory, ready to install a system...except there was no way to select the Unattended install from a boot menu.

Here's where Kyle's article helped out again. What I needed was a PXE boot menu, and thanks to his article, I was able to whip up one in fairly short order. I had to take the bzImage and initrd files out of the tftpboot directory in the linuxboot zip file on the Unattended site and place them in the /var/lib/tftpboot/ltsp/i386/ directory (I renamed the bzImage to unat and the intird to unatin.img to help distinguish them better).

Then, I created a /var/lib/tftpboot/ltsp/i386/pxelinux.cfg/default file (Listing 1) on the server, containing a combination of the syslinux boot arguments from the LTSP server and the Unattended server's configurations. Note the DISPLAY and LABEL directives. The DISPLAY directive states that when the machine boots you see the file pxemenu.msg displayed on the screen. This contains the text of the menu. The LABEL directive is what you type to boot a particular menu option. In this case, if I type "1", I get the Ubuntu LTSP session (this is also the default), and if I type "2", I get the Unattended Windows install.

## Listing 1. Example pxelinux.cfg/default file

```
default 1
serial 0,9600n8
timeout 300
prompt 1
DISPLAY pxemenu.msg
F1 pxemenu.msg

LABEL 1
KERNEL vmlinuz
APPEND ro initrd=initrd.img quiet splash

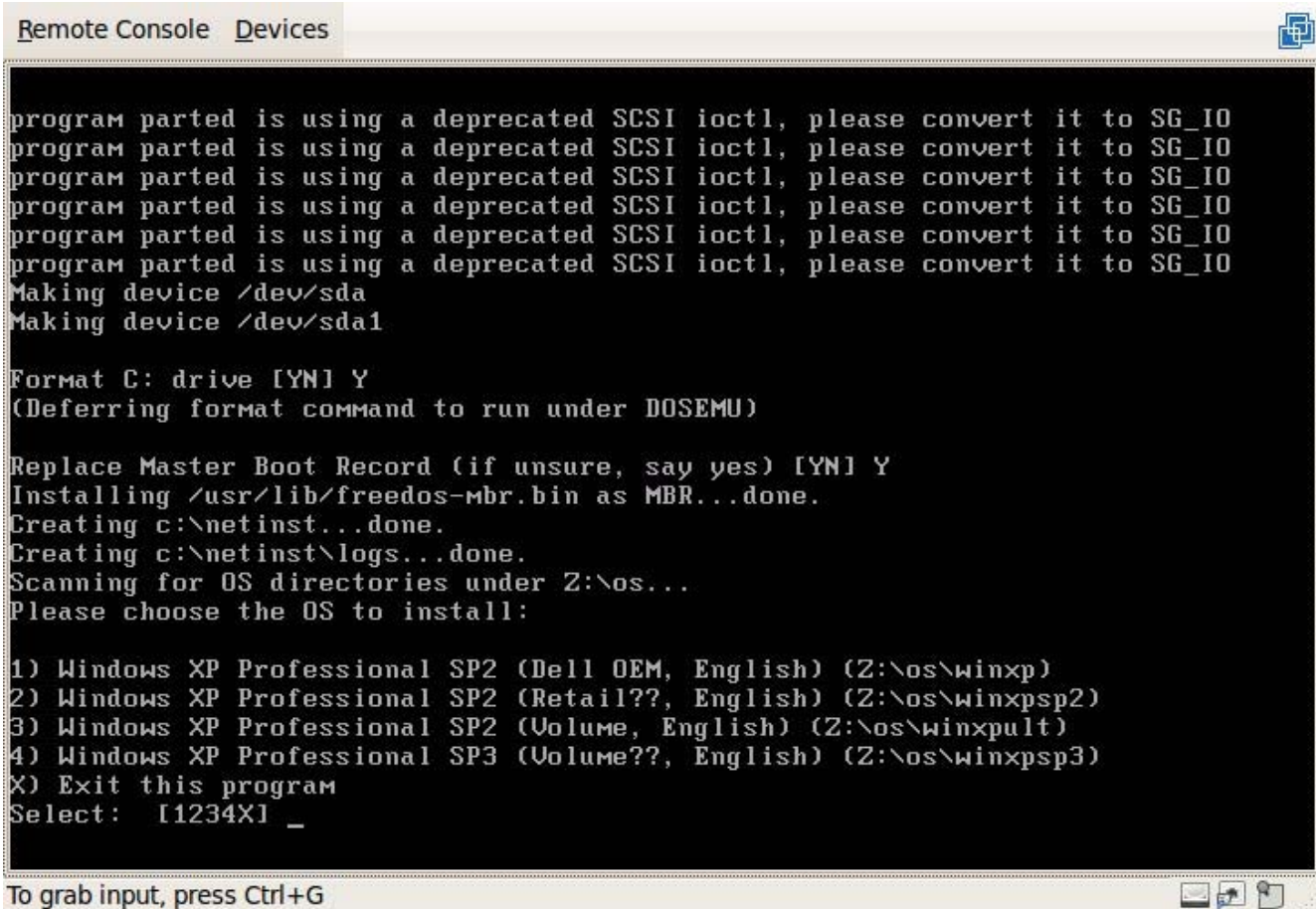
LABEL 2
KERNEL unat
APPEND initrd=unatin.img z_user=guest z_password=guest
z_path=//192.168.1.20/install
```

Now, when I booted a PXE client, I got a choice as to whether to go to the Ubuntu LTSP session or the Unattended install. At this point, I tested the Unattended installation, and it sort of worked—it installed a base Windows system just fine, but it didn't install any of the drivers, nor any of the patches to the operating system. I realized just how spoiled I am by Ubuntu's driver coverage and update manager, but I slogged ahead and continued to work to refine the system so that the driver and update installation happened without my intervention.

# PXE: Not Just for Server Networks Anymore

Bill Childers

(Reprinted from the Linux Journal)



```
Remote Console Devices
program parted is using a deprecated SCSI ioctl, please convert it to SG_IO
program parted is using a deprecated SCSI ioctl, please convert it to SG_IO
program parted is using a deprecated SCSI ioctl, please convert it to SG_IO
program parted is using a deprecated SCSI ioctl, please convert it to SG_IO
program parted is using a deprecated SCSI ioctl, please convert it to SG_IO
program parted is using a deprecated SCSI ioctl, please convert it to SG_IO
Making device /dev/sda
Making device /dev/sda1

Format C: drive [YN] Y
(Deferring format command to run under DOSEMU)

Replace Master Boot Record (if unsure, say yes) [YN] Y
Installing /usr/lib/freedos-mbr.bin as MBR...done.
Creating c:\netinst...done.
Creating c:\netinst\logs...done.
Scanning for OS directories under Z:\os...
Please choose the OS to install:

1) Windows XP Professional SP2 (Dell OEM, English) (Z:\os\winxp)
2) Windows XP Professional SP2 (Retail??, English) (Z:\os\winxpsp2)
3) Windows XP Professional SP2 (Volume, English) (Z:\os\winxpult)
4) Windows XP Professional SP3 (Volume??, English) (Z:\os\winxpsp3)
X) Exit this program
Select: [1234X] _
```

Figure 3. Choosing What Software Goes on the Machine

It turns out I didn't have to re-invent the wheel, as the driver issue and the update issue both have been addressed by the Unattended team. As far as the driver stuff goes, there is a method to integrate DriverPacks (which are large compressed archives of drivers) into the Unattended system. It's a little bit too involved for the scope of this article, but see the DriverPack link in the Resources section for more information.

With respect to automatic update installation, the method the Unattended folks use is very Linux-like in its resourcefulness. Under the Unattended root path, there are two directories: the /install/scripts and /install/tools directories. The scripts directory contains Windows batch files (.bat) that are used to do automated installation of various software packages, as well as some basic updates. The tools directory contains a set of scripts that will look at your Unattended server's current configuration and scripts directory, and then compare it to the CVS tree maintained by the Unattended team. The scripts then will grab the latest .bat files and drop them in the correct place in the scripts directory. At that point, the next Windows install that's done with the Unattended system will get all the patches and install them automatically. The system even will reboot at the appropriate times, then pick up the next patch in the series and install it. To update the Unattended system's patch repository, it's as simple as running a ./script-update; ./check; ./prepare from the /install/tools directory under the Unattended root.

# PXE: Not Just for Server Networks Anymore

Bill Childers

(Reprinted from the Linux Journal)



**Figure 4. Windows installing! Faster and more reliable, that's debatable.**

The CVS archive of scripts, as well as the script archive on the wiki, proved to be invaluable. Those resources allowed me to finish the complete automation of my install, and now, I have a configuration that meets my company's needs for Windows. After about 30 seconds of typing the machine-specific information at the beginning of the installation, I now can walk away and know that Windows, Office, the Cisco VPN client, Symantec Anti-Virus and many other things my Windows users need will be done my way, automagically, without requiring myself or another staff member to babysit it.

In closing, thanks to the efforts of the Ubuntu and LTSP teams, I now have an environment that lets my users do some kind of work, even when their systems may have some kind of issue. And, thanks to the Unattended team, I don't have to sit at a Windows machine physically to install it, nor do I have to mess with half-baked images or other strange packaging solutions. I'm already getting other ideas on how to extend this system even further.