Thin Berry Client

Remember thin client computing? Well, it's back again, and it's faster and cheaper than ever

If you don't remember the last time thin client made the news, then here's a brief intro. Back in the days when computers were big and expensive, each one had several terminals attached to it so lots of people could use one computer at the same time. PCs (personal computers) put a stop to that in many situations because they became so cheap you could put a computer on a desk for the price of a terminal. Then we got networked PCs running software held on a server and suddenly the distinction wasn't so clear any more.

'Thin client' became the preferred term for terminals that had very little processing power of their own and no locally installed applications. They did little more than display images created on the server. 'Fat client' referred to PCs that could run their own software, but often ran networked stuff in a business setting. For example, you might run an office suite and a browser on your PC but log into Sage or some other enterprise management package running on the company server.

Thin client computing has never totally gone away, though. They may be pretty rare in offices these days, but thin clients live on as PoS (point of sale) terminals in shops, and as control consoles in industrial processes. And they are sometimes used in education, libraries, computer kiosks and other situations where people want simple, cheap, secure access to basic computing. In my case, I'm experimenting with a system that will eventually go into a social enterprise hub in Llangollen.

LTSP (The Linux Terminal Server Project - **www.ltsp.org**) exists to meet that need. LTSP servers are usually just desktop PCs with a lot of memory and plenty of hard disk space. Until recently, LTSP terminals were usually either old terminals from commercial mainframe computers or just old PCs given a new lease of life as thin clients. But now we have the Raspberry Pi to play with.

BerryTerminal

BerryTerminal (**berryterminal.com**) is a minimal Linux distribution designed to turn a Raspberry Pi into a low-cost thin client, which will connect to any Linux distro running LSTP. Like all Pi projects, it's come a long way in a short time, and there are almost daily updates.

We'll look at the server setup first, though, because then you can use that to create the BerryTerminal SD card.

Server on the right, terminal on the left

LTSP Server Hardware

There's loads of information about LTSP on the LTSP website at **wiki.ltsp.org**. Specifying a server is not straightforward; it very much depends on what your users will be doing, and how many of them you have. The wiki pages suggest that if you have a few people playing simple games, a few browsing and a few using Libre Office, a single 2GHz processor should handle 20 users. I'm pretty sceptical about that, and given that you can get a less than state-of-the-art dual- or even quad-core device pretty cheap these days, I'd aim a bit higher.

The LTSP wiki gives a formula for calculating how much RAM you need and comes up with 4GB for 20 users. Elsewhere, though, I've read users suggesting 1GB for the server itself plus 500MB for each user if you are using a graphical desktop, watching video and visiting Flash encumbered websites.

Each user will get their own 'Home' directory on the server, so plenty of disk space is essential, and if you want to avoid being lynched by disgruntled users, make sure you have a RAID 1 system in case a disk fails!

For 20 or more users, you really need a gigabit network interface from server to network switch, though 100Mb is okay for each terminal. Most of the documentation assumes you have a server with two network interfaces: one to link the server to the internet, the other to handle the clients. In a home or small office you can use a single NIC or on-board Ethernet and a domestic router. As we'll see later, a separate network switch is a big help rather than relying on Ethernet ports in the router itself. For really big networks, multiple server setups can be used to share the load.

My testbed machine currently is a three-year-old, dual-core Dell Inspiron with 2GB of RAM, but I'm only running one client so far. With LTSP used in quite a few schools, a lot of references are to Edubuntu, but I did a fresh install of Lubuntu, which has the LXDE desktop as standard. The Raspberry Pi client won't handle Ubuntu's default Unity desktop, and I'm not keen on the fallback Gnome. LXDE is light and fast.

There are a couple network things to deal with before we get into LTSP itself.

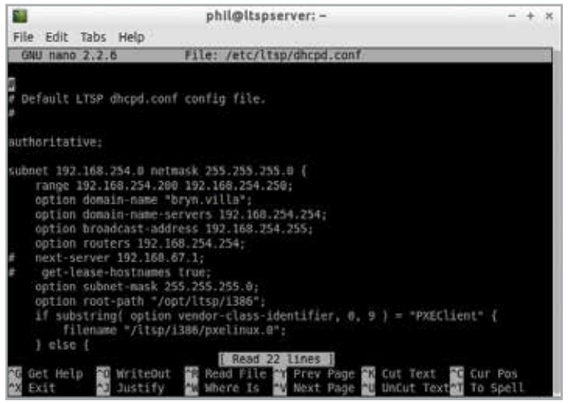
DHCP

Dynamic Host Control Protocol (DHCP) is the system used to hand out IP addresses on a network automatically. In a simple

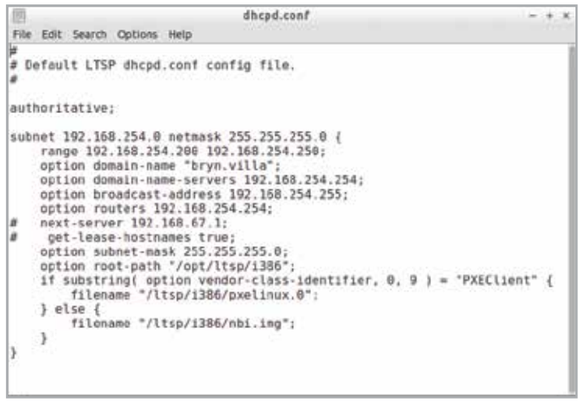
The extended case keeps the SD card secure

Raspberry Pi safely boxed

Raspberry Pi mounted on the rear of the monitor and cables coiled up neatly



Configuring DHCP in Nano



Configuring DHCP in Leafpad

home system, the router functions as a DHCP server and you don't have to think about it, but it's an issue with LTSP. The LTSP server gives IP addresses to each of the clients so they can communicate, but if it tries to give out an address already used by your router you have a problem.

The first step is to find those old login details you got when the router was new and log into it. Somewhere in the menu system you'll find details of its DHCP server settings. IP addresses on home networks are generally 192.168.xxx.yyy, where xxx and yyy can be any number from 0 to 254. You will probably find that the router uses a limited range of numbers in the last group for DHCP. On mine it's 0-199. 254 is the router's own address and three others are reserved. 200-250 are available for LTSP to use. Make a note of your router's settings. You may well find the range is adjustable, in which case adjust it as you wish.

Next go to your server's network settings and set a fixed IP address. This usually means unticking DHCP. Chose a number outside the range used by DHCP on the router (in my case, I use 200).

LTSP Server Setup

You can add LTSP to any distro using the normal package managers or one of the newer application installers. The snag is if you enter 'LTSP' in the search you'll get a huge list of applications and libraries and not much indication of what's really needed to get started. There is an easier way, but it involves a bit of typing. Open a terminal and enter:

```
sudo apt-get install ltsp-server-standalone
```

(Or if your distro allows it, login as root and dispense with 'sudo'.)

This will drag in a whole load of related applications and libraries, saving you the bother. Now we have to configure LTSP server, and that means editing a config file by hand. You need to run any text editor with root privileges, my preferred method is to open a terminal and run 'sudo nano' followed by path and file name.

Nano is a basic text editor that runs in a terminal session. If you're using LXDE, then 'sudo leafpad' opens a more modern editor. In this case the file that needs attention is dhcpd.conf so the command is:

```
sudo nano /etc/ltsp/dhcpd.conf
```

Before you start messing, back up this file. I usually just add .old to the file name so if I screw up I can find it, delete the .old and have the default file back.

As you might guess from the name and path, this file controls how LTSP uses DHCP. Referring to the sample file here, you need to change the first three sets of numbers of the subnet to match your IP address and set the last to 0.

Set range outside the DHCP range dished out by your router. Domain name is optional. Leave it on default if you want.

Domain-name-servers has to be the same as the IP address of your router.

Set option broadcast-address so the first three sets of numbers are the same as the subnet setting and end with 255.

Option routers should be the same as the option domain-name-servers. All other settings can remain the same. Save the file with its original filename. Restart the DHCP server with:

```
sudo service isc-dhcp-server restart
```

Snag 1, And A Cure

Once you've set all this, it would be reasonable to assume that the DHCP server would start when the server was booted. It's supposed to, but on my system it didn't. Starting manually is fine for me, but I want a system I can install and forget. Eventually, I installed WebMin (**www.webmin.com**). This is probably the best ever Linux setup tool that works on any distro, beating the distro's own tools. Once it's installed, open any web browser and enter:

```
localhost:10000
```

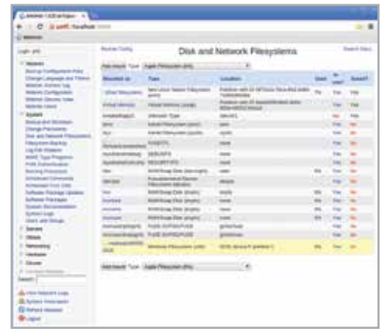
Going to System > Boot up and Shutdown, I discovered that although both the standard DHCP and IPv6 DHCP servers were set to run on boot, neither was actually running. Attempting to start DHCP6 manually from WebMin failed, so I removed the 'start at boot' option from it. Now the standard (IPv4) server runs as it should. If your network supports IPv6, this issue may not arise, but webmin is useful anyway for other admin duties, especially as the LXDE tools are pretty limited.

Build A Client

Build in this case refers to creating a place on the server into which clients will be welcomed. We'll get to building client hardware soon.

Using Webmin to select services to run at boot time

Webmin's configuration page



Using Webmin to enable users to access USB devices

Once again, the command line seems the easiest way to do this. The following command starts the process:

```
sudo ltsp-build-client --arch i386
```

It can take a long time, as LTSP needs to download and install some extra utilities and libraries. Now you can build the client hardware.

Client Hardware

A Raspberry Pi model B with Ethernet connection, case to put it in, a monitor, power supply, keyboard, mouse, video cable and SD card. Pis are currently £32.73 from Farnell, including P&P and VAT.

The case came from eBay. There are several people offering laser cut Perspex cases, but one, 'groovybananas', offers customised versions. For £7.15, it made me one a bit longer than normal so the SD card fits inside the case and users can't mess with it.

I found some 19" LCD monitors on eBay for £35. They're not widescreen, but they are 1280x1024 and they have a built-in USB hub that just needs a short USB to micro-USB cable to power the Pi. You need a monitor with a digital connection. It doesn't have to be HDMI, though; an HDMI - DVI cable (from guess where) costs about £1.50.

I found a USB keyboard and mouse set at the same place for £7.50, which are remarkably usable, and an SD card under a fiver. Total cost, about £90, not counting the sticky pads I used to fix the case to the back of the monitor. You can get Pi cases with Vesa fixing holes, but this monitor uses the Vesa mounts to attach the stand, so that wouldn't be convenient.

Client Software

Download the latest BerryTerminal. It arrives as a .zip file. Stick the SD card in your PC (any OS will do) and format the card with FAT32 file system. Unzip the download and transfer to the card. (On most Linux desktops right-click the file and 'extract to' then pick the card.) Insert the card in the Pi. Done.

Snag 2, And A Cure

It's DHCP again. Boot the server, log in and create another user. Boot the Pi. With luck, it will find the server and display a login screen. Mine didn't; it found the DHCP server on my router and displayed a basic BerryTerminal login. According to the BerryTerminal site, you can set a server address manually by opening cmdline.txt on the SD card and appending "server= [your server's IP]" and saving the file. It didn't work for me; DHCP overrules it.

What did work was switching off the router, or disconnecting it from the switch. With only one DHCP server online, the BerryTerminal finds it and displays an LTSP login. Log in with the second username and password. Reconnect the router. Once you've logged in, the terminal remembers the server location, so next time you switch on the terminal, it will connect automatically. Now you see why I recommend a separate router and switch? If you don't have a separate switch, investigate your router's settings. You can probably switch off DHCP temporarily while the terminal discovers the server.

Using BerryTerminal

Switch on the server. On my modest hardware, it boots to a login screen in under 20 seconds. Switch on the terminal - that's even faster. With only one terminal, I can't honestly say how the system will behave under load, but using the terminal is exactly like sitting at the server. Flash websites work okay and sound works via the Pi's headphone socket.

The only problem I foresee in a public setting is that the terminal cannot read USB local storage. People who arrive wanting to view photos off a camera or files off a memory stick will be disappointed. In the longer term, BerryTerminal may well release a version that can do this - LTSP clients on other hardware already can. In the short term, there is a workaround of sorts.

Insert a USB stick or camera/phone connector into a port on the server. When the pop-up appears, opt to view the device in a file manager; this will mount the file system. On the server, run WebMin. Go to System > Disk and Network Filesystems. The USB gadget should be at the end of the list. Click the blue (link) text description on the left. Midway down the page, find the option 'Allow users to mount this filesystem'. Click the 'yes' option then 'Save'.

Exit Webmin. 'Safely remove' the USB gadget. Now, with the terminal, running insert any USB gadget into the server and in a second or so a dialogue box will open on the terminal asking for authentication. At this point, you need to give the server's admin password, so in a public setup this isn't ideal. If you (the admin) go to the terminal and enter it yourself and don't just hand out the password to all and sundry, it will probably be okay. It's a good idea to change the admin password regularly, though. **mm**