Linux Terminal Server Project Administrator's Reference

A Guide to LTSP Networks

Ed. 1.2

Copyright © 2008, 2015 Scott Balneaves and other authors

Permission to use, copy, modify and distribute the software and its accompanying documentation for any purpose and without fee is hereby granted in perpetuity under the terms of the GNU GPL2, provided that a copy of the GNU GPL2 appears with it.

The copyright holder makes no representation about the suitability of this software for any purpose. It is provided "as is" without expressed or implied warranty. If you modify the software in any way, identify your software as a variant of LTSP.

COLLABORATORS

	TITLE : Linux Terminal Server Project Administrator's Reference	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Scott Balneaves, Jordan Erickson, Francis Giraldeau, Richard Johnson, David Johnston, Chuck Liebow, James McQuillan, Jonathan Mueller, Gideon Romm, Joel Sass, Robin Shepheard, Susan Stewart, Brian Tilma, David Van Assche, Carol Wiebe, and Vagrant Cascadian	Aug, 2015	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Linux Terminal Server Project - LTSP **3**

1.1 Introduction to LTSP and Thin Client Computing . . . 3
1.2 LTSP Security . . . 4
1.3 LTSP Manageability . . . 4
1.4 It's Green! . . . 4
1.5 Cost Effective . . . 4
1.6 Well Supported . . . 5
1.7 Built for Education, Government and business . . . 5

2 Basic Concepts: Networks and Networking **7**

2.1 Hardware . . . 7
2.1.1 Wired . . . 7
2.1.2 Wireless . . . 8
2.2 Software . . . 8
2.3 How LTSP Works . . . 9

3 LTSP Thin Client hardware requirements **11**

3.1 Hardware reuse and sizing . . . 11
3.2 Clients . . . 11
3.2.1 Older hardware . . . 11
3.2.2 CPU . . . 12
3.2.3 Network . . . 12
3.2.4 Thin client memory . . . 12
3.2.5 Video Card . . . 12

4 LTSP Server requirements **13**

4.1 Recommended specs . . . 13
4.1.1 Memory . . . 13
4.1.2 Processors . . . 13
4.1.3 Disks . . . 14

5 Network 15

6 The LTSP chroot environment 17

6.1 The boot process of a thin client . . . 17

7 Network booting the thin client 21

7.1 Boot ROM . . . 21

7.1.1 Etherboot . . . 21

7.1.2 PXE . . . 21

7.2 Local media . . . 21

7.2.1 Floppy disk . . . 22

7.2.2 Hard disk . . . 22

7.2.3 CD-ROM . . . 22

7.2.4 USB Memory device . . . 22

7.3 Installation . . . 22

8 Customizing thin client behaviour 23

8.1 Location of the lts.conf file . . . 23

8.2 Sample lts.conf file . . . 23

9 Format of the lts.conf file 25

9.1 Section headings . . . 25

9.2 Variable Assignments . . . 25

9.3 The LIKE keyword . . . 25

10 General thin client parameters 27

11 Local Device thin client parameters 31

12 Screen Scripts 33

13 The "rdesktop" screen script 35

13.1 Single Line . . . 35

13.2 Multiple Line . . . 35

13.3 RDP + Local Devices . . . 35

13.4 RDP + Local Sound . . . 36

14 Modules and startup scripts 37

15 Sound in LTSP 39

16 X-Windows parameters **41**
16.1 X.org Configuration . . . 41

17 XRANDR setting for managing displays **45**
17.1 New Xorg structure within LTSP . . . 45
17.2 Script structure . . . 45
17.3 XRandR parameters . . . 46

18 Printer configuration parameters **49**

19 Keyboard parameters **51**

20 Touchscreen configuration **53**

21 Local Applications **55**

22 The LDM display manager **57**
22.1 Introduction . . . 57
22.2 Theory of operation . . . 57
22.3 Encrypted versus unencrypted sessions . . . 58
22.4 Auto login features . . . 58
22.5 Load balancing features . . . 58
22.6 RC script capabilities . . . 58
22.7 LDM lts.conf parameters . . . 59
22.8 Multiple server setup . . . 62

23 Infrastructure setup **63**
23.1 Network topology . . . 63
23.2 Common authentication . . . 63
23.3 Shared home directories . . . 63
23.4 Shared printers . . . 64
23.5 Managing the SSH known hosts file . . . 64
23.6 Setting Network Forwarding or Masquerading . . . 64

24 Session dispatching **65**
24.1 Define the server list . . . 65
24.2 Dispatching method . . . 65
24.3 Network Swap . . . 66
24.4 Managing DHCP . . . 66

25 Adding static entries to the dhcpd.conf **69**

26 DHCP failover load balancing **71**

26.1 71

26.1.1 Lockdown with Sabayon (user profile manager) and Pessulus (lockdown editor) 71

26.2 Replication of desktop profiles 71

27 Managing the thin client **73**

27.1 Lockdown Editor 73

28 Updating your LTSP chroot **75**

29 Changing the IP of your LTSP server **77**

30 Appendix I **79**

30.1 Using NFS instead of NBD 79

30.2 Enabling dual monitors 80

A Guide to LTSP Networks

Chapter 1

Linux Terminal Server Project - LTSP

1.1 Introduction to LTSP and Thin Client Computing

One of the key technologies included in modern GNU/Linux operating systems is the Linux Terminal Server Project (LTSP) which allows you to boot thin clients from an LTSP server. For educational environments, LTSP lowers hardware costs by enabling the use of older or less powerful machines as thin clients, as well as reduced administration overhead by having only to install and maintain the software on the server. When a workstation fails, it can simply be replaced without data loss or re-installation of the operating system.

Thin client computing has been around for a long time in the UNIX world. Although the implementation has evolved quite a bit, the concept has remained the same:

1. The thin client only takes care of the basic functions like display, keyboard, mouse and sound.

2. The server does the heavy weightlifting. All the applications run on the server, and they simply display on the thin client.

Because the thin clients have a limited number of tasks to manage, the hardware for the thin client can be small and cheap. The thin clients themselves are basically maintenance free. They last longer because they have no storage with moving parts like hard disks. If they break no data is lost since nothing is stored on the client itself. Simply swap the client with another one and go back to work. If your thin client is stolen or put in the trash, no data ends up in the hands of unauthorized people.

The terminal server runs all applications and contains all the data. All the regular maintenance (software updates, administration) takes place on the terminal server. The number of thin clients that a terminal server can support is proportional to the power of the server. Because GNU/Linux makes efficient use of resources, you can support a surprising number of thin clients from a machine which might only be considered a powerful single user system running other operating systems. Please see for more details.

In a thin client computing environment, the stability of the server is important. It's important to make sure that your server has good power fallback facilities, like installing a UPS, and depending on how much availability is required, redundant power supplies may be called for. As well, users who have the resources may decide to invest in multiple disks for RAID support, and other options which may be needed in a High Availability environment. However, you certainly don't need them in all environments, and GNU/Linux's high quality means that in all but the most demanding environments, this won't be needed.

1.2 LTSP Security

Security has become a key challenge for administrators and LTSP both recognizes and handles this quite well. Often schools lack the specialized IT staff or time to lock and clean up computers.

Operating systems with LTSP included, being Linux-based operating systems, enjoy the security advantages of its Unix-like and open source heritages. This translates into higher quality code and no spyware and viruses, like they plague other operating systems.

In addition, it has a strict, proactive security policy which means that many common problems, such as open ports or misconfigured software, never make it into the released product. Finally, LTSP based systems are true multi-user operating systems, making it easy to allow users to complete their tasks without having a level of access that could compromise the system.

1.3 LTSP Manageability

With administrators and especially school IT departments deploying and administering an increasing number of computers, it is difficult to find time to manage individual machines. LTSP thin client technology, makes deployment and management simple and easy. A single server is all that is needed to set up, manage and administrate an entire network It is recognized that not every school's setup is the same, so LTSP (and the underlying operating system) has been made to easily customize your unique needs.

1.4 It's Green!

With the ongoing debate about climate change, questions are finally being asked and answered in the fields of IT, education and thin client technology in general. A recent study compared the energy and resource consumption of a regular PC and Thin Client setups. You can find that study here: http://it.umsicht.fraunhofer.de/TCecology/-index_en.html

They found that thin clients use half the energy of traditional workstations, which not only helps on the cost savings (calculate that a 40 terminal thin client lab will save approximately $500-$800 per year), but is ecologically effective in avoiding electronic waste and high carbon emissions. Thin client production, assembly and logistics costs far less and requires less energy than traditional PC manufacturing. The recycling of old machinery also helps the environment, making LTSP a green solution to the environmental and power saving issues many IT managers face today.

1.5 Cost Effective

With ever-increasing demands on school budgets, expensive technology is often last on the list. LTSP can help you offer what your students increasingly require from computer technology, without breaking the bank. GNU/Linux is and always will be free to acquire, use and modify, including the underlying LTSP structure that holds it all together.

Need to set up another machine? Or another 100? Just install them! With GNU/Linux you'll have no more expensive OS upgrades and licenses, and having specialized programs on only some computers will become a thing of the past. When you build your network on Open Source software, you are freed to seek support for your computers from whomever you wish.

GNU/Linux with LTSP can also help you save hardware costs, by allowing you to redeploy older machines as thin clients using LTSP technology. Whether you choose to set up many smaller labs with various LTSP servers or one giant setup with a load balanced LTSP setup (various servers working together to manage the users and applications logging on) the cost savings are always enormous.

1.6 Well Supported

LTSP support is available from the LTSP community. Many of the authors of the software included in LTSP, including the respective developers of the various LTSP GNU/Linux distribution implementations themselves, can be contacted directly via mailing lists and IRC channels.

There are many forms of support available, including mailing lists, Wiki websites, IRC channels, and bug trackers. There are also special support groups for using LTSP and GNU/Linux.

The official IRC support channel is found on freenode.org at #ltsp

The official LTSP mailing list is found here:

https://lists.sourceforge.net/lists/listinfo/ltsp-discuss

In fact, some of the money that would have gone to purchasing software can instead be spent to hire local experts to help train you, and to help you support your network. LTSP can help you take more control over your network while also benefiting your local economy. With LTSP based systems, these choices are yours.

1.7 Built for Education, Government and business

LTSP based distros come with translations for many languages and localization features that allow people from all over the world to enjoy their computing experience. Accessibility features strive to provide a pleasant, high-quality computing experience to disabled users.

The Free nature of GNU/Linux means that the applications a user is used to at their school or workplace is also available to them at home. Users can install their favorite GNU/Linux distribution at home, and have all the same functionality they are used to. In fact, many GNU/Linux distributions have Live CD's, which allow users to try, or even fully use the distribution at home, without even installing it on their home machine.

The LTSP server software allows administrators, IT managers and teachers to create a low cost computer lab so that users can have access to the opportunities that GNU/Linux and the Internet can provide.

Since setups can be adjusted to many situations, each thin client lab can be uniquely tailored to fit the business, agency or educational facility in question.

Chapter 2

Basic Concepts: Networks and Networking

There are two components of a network: hardware and software. This section will give an introduction to both.

2.1 Hardware

Networking works by breaking files and other data into little packets of information. These packets are transferred over a network. The difference between various types of networks is how they transfer packets.

There are two types of networking hardware: wired and wireless.

An important fact to remember is that a network will be only as fast as the slowest part. Making sure that your network setup matches your intended use case is an important consideration in an LTSP network.

2.1.1 Wired

Wired networking transfers packets over a cable that resembles a telephone cord, but with more wires. Wired networks can transfer packets at one of three possible speeds: 10 Mbit/sec, 100 Mbit/sec, or (Gigabit) 1000 Mbit/sec.

A network is only useful if it can connect multiple computers. There are some pieces of hardware that allow multiple computers to be connected in a network. They look alike, but they function differently and, likewise, operate at different speeds.

Hub

A hub is the simplest way to connect multiple computers. A hub has a lot of ports in the front and usually has several small lights corresponding to each port. The hub takes a message it receives on one port and re-sends it to all the ports. As a result, only one port can talk at a time.

Switch

A switch looks a lot like a hub; it has a lot of ports in the front and usually has several small lights corresponding to each port. However, a switch is unlike a hub because it only makes a connection between the ports it needs to. A switch can have multiple connections at the same time. This allows a switch to be faster than a hub.

Router

A router is used to make a connection between two networks. Routers are also commonly used to connect a LAN (local area network) to the Internet.

2.1.2 Wireless

Some people may wish to try using LTSP in a wireless environment, for various reasons. This represents some challenges.

Wireless networks typically have more latency than wired networks, which generally makes interactive programs feel slow and unresponsive. As well, wireless adaptors cannot directly PXE boot, as you need to set things such as ESSID, keys, etc., which wouldn't be there in a PXE capable card.

However, for those wishing to use LTSP wirelessly, it is still possible, but requires more hardware. Wireless bridge boxes are available, which contain both an ethernet and a wireless network connection. One can typically connect to them like a small Internet router box, and program them with the information pertinent to your network. You can then use a standard wired network card connected directly to the bridge, and the bridge itself will handle the wireless part.

This method has been used with success by users of LTSP in the past. The latency of wireless makes the experience slower, however, depending on the application you wish to use, you may find it acceptible.

2.2 Software

The most common network infrastructure services include:

- DHCP (Dynamic Host Configuration Protocol)

  Each computer on a network needs a unique identifier called an IP address. The IP address allows packets to be directed to the computer, much like a street address allows mail to be delivered to the correct house. An IP address follows a specific form: four groups of digits forming a number from 0 to 255. For example, a local IP address might be 192.168.2.50.

  For convenience, a computer's IP address can be given by a server running the Dynamic Host Configuration Protocol (DHCP) service. DHCP automatically provides network settings to the computers on the network. With DHCP, there is no need to keep track of each computer's IP address.

- DNS (Domain Name System)

  DNS is a service that runs on a server, and it is like a phone book for computers, except that it stores IP addresses instead of phone numbers. Your computer talks to a DNS server every time you refer to another computer system with a name instead of an IP address. For example: www.ltsp.org, wikipedia.org, and google.com are all DNS hostnames.

- NTP (Network Time Protocol)

  NTP is a service that runs on a server and allows other computers to synchronize their clocks. The server synchronizes with an extremely accurate atomic clock, and then the clients synchronize with the server.

- Web Server

  A Web server answers queries using protocols such as HTTP, and sends content such as web pages back to clients. Your Web browser almost exclusively talks to Web servers.

- Web Proxy

  A Web proxy is a service that runs on a server and accesses Web sites on behalf of the clients. A proxy can cache some data to allow faster repeated access to commonly accessed pages. This is not really needed in essence for ltsp thin clients, since nothing runs on them, it all runs on the server. But in order to allow for content filtering, a proxy is required. In the case of a mixed network, where some clients are independent from the the thin client network, a proxy server is useful. The most common and recommended proxy solution is called Squid, which can be easily installed through your distro's package manager.

- Content Filter or Net Guardian

  A typical network requires a filtering policy to be implemented, which can easily be done by software like dansguardian, squidguard or squid-filter. This allows an administrator to block and control unwanted traffic like:

  1. banner ads,
  2. user behaviour tracking via cookies,
  3. animated pictures,
  4. JavaScript, VBScript, ActiveX (dangerous as well as annoying).

- Firewall & Port Blocker

  A firewall is usually a service on the server, but often DSL routers have the basic functionality of a firewall too. A firewall can protect your server (and clients) by restricting or allowing computers on the Internet from initiating connections into your server or network. There are many programs available for different distros. On Ubuntu and Debian we recommend using gufw (uncomplicated firewall), while Fedora has Fedora Firewall GUI, and SuSE has Yast2 Firewall. If they are not already installed, you can simply install them with your distro's package manager

2.3 How LTSP Works

1. LTSP is a collection of software that turns a normal GNU/Linux installation into a terminal server. This allows low-powered, low-cost thin-clients (or legacy hardware you already own) to be used as terminals to the thin-client server. LTSP is unique from other thin-client systems in that it is considered by many as the easiest to maintain. Other thin-client systems require each client to have software that boots the system to a point to be able to connect to the terminal server. This could be a full-blown operating system, or a minimal OS that simply provides an interface to connect to the server. Systems such as this generally require more maintenance and administration, as the local software that boots the thin-clients may become corrupt or contain bugs that require attention. LTSP, on the other hand, requires no client-side software. It requires only a PXE capable network interface, which many thin-clients and PCs have built-in already. This means that you need absolutely no physical storage media (hard disk, compact-flash, etc.) for your thin-client to boot to LTSP. This significantly reduces the amount of administration required to keep your network running. The process of booting a thin-client to an LTSP server is as follows:
2. Thin-clients boot via a protocol called PXE (Pre-eXecution Environment)
3. PXE requests an IP address from a local DHCP server
4. The DHCP server passes additional parameters to the thin-client and downloads a Linux initramfs filesystem image via TFTP into a RAM disk on the client itself.
5. The thin-client then boots the downloaded Linux initramfs image, detects hardware, and connects to the LTSP server's X session (normally handled by LDM).

From here, all operations such as authenticating your username and password, launching applications, and viewing websites are actually handled on the LTSP server rather than the thin-client. The LTSP server transfers all graphical information to the thin-client over the network. This allows very low powered thin-clients to utilize the power of the server for all operations. It also allows for large client deployments with reduced overall resource utilization, as 50 thin-clients all running the popular OpenOffice suite under different sessions generally only require enough RAM for a single instance of OpenOffice (excluding per-user configuration which is minimal). The server shares memory between user sessions, so libraries for applications are only loaded once and referenced for each user session.

Chapter 3

LTSP Thin Client hardware requirements

A lot of LTSP deployments are in classroom environments, and usually, in these situations, the primary goal is to re-use existing hardware that the school already owns. However, specifically designed thin clients can be used also.

3.1 Hardware reuse and sizing

A person setting up a LTSP thin client environment for the first time, typically asks two questions:

1. Will my existing machines work as terminals, or, what should I buy to use as a terminal?
2. How big a server do I need?

Chances are, hardware that you already have is more than sufficient for terminals. One of the great advantages of an LTSP Server is that you can set up a high quality lab of terminals for your students to use, by leveraging the machines you already have. As for servers, usually, it's very easy to turn any high-end single user desktop machine into a terminal server capable of handling many thin clients. We'll present some guidelines that should help in making the most of your resources.

3.2 Clients

3.2.1 Older hardware

There are three things to consider when trying to re-use existing hardware:

1. CPU
2. Network
3. Thin client memory
4. Video card

3.2.2 CPU

For using the default, secure mode of LTSP, you'll need to have a slightly faster CPU. Any 533 MHz or better CPU should provide acceptable performance.

If you have slower clients, in the range of 233 MHz to 533 MHz, you may be able to use them, if you're willing to reduce the security of your thin client network. More information on this is available in the chapter on LDM.

3.2.3 Network

A thin client boots over the network, using a small program called a network boot loader. This network boot loader is sometimes located on the card itself, or, for older cards without one, the user can provide one on a floppy or CDRom which can be used to boot the thin client.

Three common network boot loaders which can be used are:

1. *PXE:* This one is the most common, and many network cards and motherboards with built-in network cards support this. If you have one of these, you'll be able to boot without any problems.
2. *Etherboot/gPXE:* For older cards that don't have PXE included on them, you can use the Free Software equivalent, Etherboot, or it's newer replacement, gPXE. This excellent alternative to PXE can either be booted from a floppy, memory stick, or CDRom, or, if you're handy with electronics, be burned onto a EPROM if your card has a socket for one. More information on the project can be found at http://www.etherboot.org, and you can download ready-to-use Etherboot images at http://www.rom-o-matic.org.
3. *Yaboot:* For Macintosh PowerPC machines (iMac's and later), you can use the built in Yaboot network boot.

3.2.4 Thin client memory

The bare minimum for a thin client to work is about 48MB, but it will be unusably slow, so it is recommended to install at least 128MB Ram, with 256MB Ram if you can spare it. This will really help speed up thin clients.

3.2.5 Video Card

Typically, any video card that uses the PCI bus and has 16 MB or more of memory, should make a reasonable client.

Chapter 4

LTSP Server requirements

An LTSP thin client network is quite scalable; a moderately powerful machine can serve several thin clients, and if you need to add more thin clients, you can either expand the capabilities of the existing server, or, simply add more servers.

4.1 Recommended specs

Server sizing in an LTSP network is more art than science. Ask any LTSP administrator how big a server you need to use, and you'll likely be told "It depends". How big a server you need does depend largely on what it is you're planning on doing with your thin client network. The server requirements needed for a network where the only use will be a little light web-browsing, with no Java or Flash, will be greatly different from a network where you want to do heavy graphics, interactive games, and Flash animation. Here are some common guidelines that should fit most "average" cases.

4.1.1 Memory

A GNU/Linux based operating system makes efficient use of memory. The usual formula that's used for adding memory to a thin client server is:

256 + (192 * users) MB

So, if your target is to have a server with 20 terminals, you'll need:

256 + (192 * 20) = 256 + 3840 = 4096 MB

So, you'll need 4 1 Gig memory sticks. Making sure you've got enough memory is the single most important thing you can do to help the performance of an LTSP thin client server. If you do not have enough memory in your server, you'll find your server will have to use the hard drive as an overflow "virtual" memory. Hard drives are much slower than memory, so you'll find things getting very slow if this happens.

If you intend to make heavy use of graphics work in your curriculum, you may want to add even more, perhaps doubling the previous estimate.

4.1.2 Processors

How fast a processor you need is entirely dependant on what programs you plan to use. Interactive games require a bit more than say, a word processor. If you plan to use Java and Flash plug-ins in your web browser, these can consume

a lot of processing power. For a "mixed" model, i.e. some people playing TuxMath, a few people browsing the web, and a few people typing in OpenOffice.org, a 2GHz or better processor should be able to adequately handle 20 people with some minor delays. A 3GHz processor would be better.

For larger networks, moving to an SMP (Symmetric Multi Processing), or multiple CPU server may be advantageous. If you plan to handle 30 or more clients, a newer dual-core Xeon server or dual-core Opteron will provide good results.

Remember, if you need to serve a large number of clients, it will be worth your while to configure multiple LTSP servers, each handling some of the terminals.

4.1.3 Disks

It's advisable to use some form of RAID in the terminal servers. Besides saving your data when a single disks fails, it improves the performance (especially read performance, which is the most common type of file access). For people on a budget, setting up software RAID 1, with 2 SATA disks with NCQ (Native Command Queueing) will provide good results. If you have a bigger budget, and a bigger network, setting up your server with RAID 10 along with 10,000 RPM western digital VelociRaptors will give you the best speeds possible. This will provide you with top notch performance and reliability.

Chapter 5

Network

If you have more than 20 users, it is recommended to use Gigabit Ethernet connected to a gigabit port on a switch for your LTSP servers. Although normal usage ranges from 0.5 to 2mbit, clients can peak quite high (70mbit), especially when watching multimedia content.

Booting a thin client involves several steps. Understanding what is happening along the way will make it much easier to solve problems, should they arise.

There are four basic services required to boot an LTSP thin client. They are:

1. DHCP
2. TFTP
3. NFS or NBD
4. SSH

Chapter 6

The LTSP chroot environment

In order to turn a computer into a thin client, we need to run a mini version of GNU/Linux on the workstation. It needs to boot this mini version of GNU/Linux over the network, since it probably won't have a hard drive on it's own. This mini GNU/Linux installation needs to live somewhere, and the best place for it is on the server.

This scaled-down GNU/Linux installation, customized so that it's efficient to boot over the network, is called a chroot environment. You can have several of them, based upon several different CPU architectures.

They'll normally live under `/opt/ltsp` on the server, with sub directories for each of the architectures. For instance, if you have a lab full of old Power PC Macs, and older PC's, you'll have an `/opt/ltsp/ppc` and an `/opt/ltsp/i386` directory on the server.

This is the LTSP project's preferred area to store the chroot, however, different distros that support LTSP are free to change this. Check with your distro's specific LTSP documentation to see where the LTSP chroot is stored.

The reason why it is called a chroot environment is that to install it, the GNU/Linux command chroot is called to actually set the installation root to `/opt/ltsp/<arch>`. From there, a scaled-down version of the distribution is installed. What this means is that for you to manage the chroot, performing such things as updates, all you need to do is use the chroot command to change the root of your installation. Then you can use all your tools like you normally would.

6.1 The boot process of a thin client

1. Load the Linux kernel into the memory of the thin client. This can be done several different ways, including:
2. Each of the above booting methods will be explained later in this chapter. But for now, it should be noted that it makes sense in almost all cases to use a PXE based network card during booting for the fastest, and easiest to setup method.
3. Once the kernel has been loaded into memory, it will begin executing.
4. The kernel will initialize the entire system and all of the peripherals that it recognizes.
5. This is where the fun really begins. During the kernel loading process, an initramfs image will also be loaded into memory.
6. Normally, when the kernel is finished booting, it will launch the new task launcher **upstart**, which will handle starting up a server or workstation. But, in this case, we've instructed the kernel to load a small shell script instead. This shell script is called **/init**,and lives in the root of the initramfs.

7. The **/init** script begins by mounting `/proc` and `/sys`, starts **udev** to discover and initialize hardware, especially the network card, which is needed for every aspect of the boot from here on. As well, it creates a small ram disk, where any local storage that is needed (to configure the `xorg.conf` file, for instance) can be written to.

8. The *loopback* network interface is configured. This is the networking interface that has *127.0.0.1* as its IP address.

9. A small DHCP client called **ipconfig** will then be run, to make another query from the DHCP server. This separate user-space query gets information supplied in the dhcpd.conf file, like the nfs root server, default gateway, and other important parameters.

10. When **ipconfig** gets a reply from the server, the information it receives is used to configure the Ethernet interface, and determine the server to mount the root from.

11. Up to this point, the root filesystem has been a ram disk. Now, the **/init** script will mount a new root filesystem via either NBD or NFS. In the case of NBD, the image that is normally loaded is `/opt/ltsp/images/<arch>.img`. If the root is mounted via NFS, then the directory that is exported from the server is typically `/opt/ltsp/<arch>`. It can't just mount the new filesystem as `/`. It must first mount it to a separate directory. Then, it will do a **run-init**, which will swap the current root filesystem for a new filesystem. When it completes, the filesystem will be mounted on `/`. At this point, any directories that need to be writable for regular start up to occur, like `/tmp`, or `/var`, are mounted at this time.

12. Once the mounting of the new root filesystem is complete, we are done with the **/init** shell script and we need to invoke the real **/sbin/init** program.

13. The **init** program will read the `/etc/event.d` directory and begin setting up the thin client environment. From there, upstart will begin reading the start up commands in `/etc/rcS.d`.

14. It will execute the **ltsp-client-setup** command which will configure many aspects of the thin client environment, such as checking if local devices need starting, loading any specified modules, etc.

15. Next, the **init** program will begin to execute commands in the `/etc/rc2.d` directory

16. One of the items in the `/etc/rc2.d` directory is the **ltsp-client-core** command that will be run while the thin client is booting.

17. The `lts.conf` file will be parsed, and all of the parameters in that file that pertain to this thin client will be set as environment variables for the **ltsp-client-core** script to use.

18. If Sound is configured at this point, the **pulseaudio** daemon is started, to allow remote audio connections from the server to connect and play on the thin client.

19. If the thin client has local device support enabled, the **ltspfsd** program is started to allow the server to read from devices such as memory sticks or CD-Roms attached to the thin client.

20. At this point, any of the screen sessions you've defined in your `lts.conf` will be executed.

    Screen sessions are what you want to launch on all of the virtual screens on your terminal. These are the standard virtual screens that all GNU/Linux distributions usually have, i.e. Alt-F1, through Alt-F10.

    By default, a standard character based getty will be run on screen 1 (`SCREEN_01` in the `lts.conf` file).

    As well, if nothing else is specified in the `lts.conf` file, an **ldm**ldm; screen script is run on `SCREEN_07`. The LTSP Display Manager (**ldm**ldm;) is the default login manager for LTSP.

21. If `SCREEN_07` is set to a value of **ldm**ldm;, or **startx**, then the X Windows System will be launched, giving you a graphical user interface.

    By default, the Xorg server will auto-probe the card, create a default `/etc/X11/xorg.conf` file on the ram-disk in the terminal, and start up xorg with that custom config.

22. The X server will either start an encrypted **ssh** tunnel to the server, in the case of **ldm**ldm;, or an an XDMCP query to the LTSP server, in the case of **startx**. Either way, a login box will appear on the terminal.

23. At this point, the user can log in. They'll get a session on the server.

    This confuses a lot of people at first. They are sitting at a thin client, but they are running a session on the server. All commands they run will be run on the server, but the output will be displayed on the thin client.

Chapter 7

Network booting the thin client

Getting the thin client to boot over the network can be accomplished in a variety of ways:

1. Boot ROM
2. Local media

7.1 Boot ROM

Depending on your network card, it may already contain a boot ROM, or you may be able to use an EPROM programmer to create your own. Check the hardware documentation for the network card in your thin client for details.

7.1.1 Etherboot

Etherboot is a very popular open-source bootrom project. It contains drivers for many common network cards, and works very well with LTSP.

ROM images suitable for booting from floppy, CD-ROM, etc., can be obtained from http://www.rom-o-matic.org

Linux kernels must be tagged with the **mknbi-linux**, which will prepare the kernel for network booting, by prefixing the kernel with some additional code, and appending the initrd to the end of the kernel.

The kernels that are supplied with LTSP are already tagged, and ready to boot with Etherboot.

7.1.2 PXE

Part of the 'Wired for Management' specification from the late 1990's included a specification for a bootrom technology known as the *Pre-boot Execution Environment* commonly abbreviated as *PXE*.

A PXE bootrom can load at most a 32 kilo-byte file. A Linux kernel is quite a bit larger than that. Therefore, we setup PXE to load a 2nd stage boot loader called **pxelinux**, which is small enough to be loaded. It knows how to load much larger files, such as a Linux kernel.

7.2 Local media

If your network card in the thin client doesn't have a boot ROM built in, and you don't have access to an EPROM burner, have no fear! Chances are, that old machine has a floppy drive, or CD-ROM in it. If so, then you can use local media to boot the thin client.

7.2.1 Floppy disk

Booting Etherboot from a floppy is an excellent way of booting an LTSP thin client that doesn't have a boot ROM. Etherboot is loaded in the boot sector of the floppy. Then, it will act just like a bootrom. The boot code will be executed, the network card will be initialized, and the kernel will be loaded from the network server.

7.2.2 Hard disk

The hard disk can be used with LILO or GRUB, to load the Linux kernel and initrd. You can also load the Etherboot bootrom image from the hard disk, and it will act like a bootrom.

7.2.3 CD-ROM

A bootable CD-ROM can be loaded either with a Linux kernel, or an Etherboot image.

7.2.4 USB Memory device

Just like a CD-ROM, Floppy disk and Hard disk, you can use a USB Memory device to boot an Etherboot module.

7.3 Installation

With the integration of LTSP into distributions, installation of LTSP is now usually as easy as adding the LTSP packages in your distro's package manager. Consult your distribution's documentation for details on how to install LTSP on your particular system.

However, as a general guideline, usually after you've installed your distributions' LTSP packages, and configured your network interfaces, and some kind of DHCP3 server, you'd run (as root):

```
sudo ltsp-build-client
```

If you are on a 64-bit system but your clients have another architecture use the --arch option e.g.

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

After that, you should be able to boot your first thin client.

Chapter 8

Customizing thin client behaviour

By default, most thin clients will automatically configure themselves correctly, and just work when they're plugged in. However, sometimes you may wish to customize their behaviour. You would do this by editing the `lts.conf` file.

8.1 Location of the lts.conf file

In order to speed up LTSP, by default, we're using NBD (Network Block Devices) rather than NFS. To do this, we'd had to move the the lts.conf file out of the chroot and into the TFTP directory, in /var/lib/tftpboot/ltsp/<arch>, where <arch> is the architecture you are working on (usually i386, but could be something else, like amd64 for example). This means you can make changes to the file immediately, and simply reboot the terminal, without recompiling the image.

8.2 Sample lts.conf file

Here is an example of the lts.conf file:

```
################
# Global defaults for all clients
# if you refer to the local server, just use the
# "server" keyword as value
# see lts_parameters.txt for valid values
################

[default]
    X_COLOR_DEPTH=16
    LOCALDEV=True
    SOUND=True
    NBD_SWAP=True
    SYSLOG_HOST=server
    XKBLAYOUT=de

################
#[MAC ADDRESS]: Per thin client settings
################

[00:11:25:84:CE:BA]
```

```
    XSERVER = vesa
    X_MOUSE_DEVICE=/dev/ttyS0
    X_MOUSE_PROTOCOL=intellimouse

###############
# A Thin Client Print server
# (switch off X by pointing tty7 to shell,
# to save resources)
###############

[00:11:25:93:CF:00]
    PRINTER_0_DEVICE=/dev/usblp0
    SCREEN_07=shell

###############
# A workstation that executes a specific
# command after login
###############

[00:11:25:93:CF:02]
    LDM_SESSION=/usr/bin/myloginscript
```

Chapter 9

Format of the lts.conf file

When LTSP was designed, one of the issues that needed to be dealt with was varying hardware configurations for the thin client. Certainly, whatever combination of processor, network card and video card available today would not be available in 3 months, when you want to add more thin clients to the network. So, LTSP devised a way of specifying the configuration of each thin client. The configuration file is called lts.conf and it lives in the `/var/lib/tftpboot/ltsp/<arch>/` directory.

The format of the lts.conf allows for 'default' settings and individual thin client settings. If all of your thin clients are identical, you could specify all of the configuration settings in the `'[Default]'` section. The file must have a first line containing `'[Default]'` in any case.

9.1 Section headings

Section headings begin with an identifier in the form [Default] which is used for all computers as mentioned above, and [MAC address] for individual workstations, in the form of [XX:XX:XX:XX:XX:XX], where X is the digits 0-9, or A-F.

You can usually read the MAC address for a network card from a sticker on the card itself, or use some kind of network tool to discover it. The best way to check for the MAC address of the machine is by starting it up, checking its IP number and doing a **arp -an** on the server, this should then tell you which IP number has which MAC address.

9.2 Variable Assignments

After the section heading, you can then define variables. Variables are ether boolean values, requiring a True/False or Y/N answer. Note that you can either use True or False, Yes or No, or Y or N. Whichever you prefer. Other variables may simply be strings, supplied after the = sign. The general format of an assignment looks like:

```
VARIABLE = value
```

Comments can be inserted into the file for your documentation purposes. Comments start with a # character, and everything after the # for the rest of the line is considered a comment.

9.3 The LIKE keyword

The `LIKE` keyword allows you to define a general set of parameters under a unique identifier, and then assign individual workstations that set of parameters using the `LIKE` keyword. An example will illustrate it's use.

Let's assume you have 3 kinds of thin clients on your network. One set, which are used in the lab, have older video cards, and must use 16 bit colour at 1024x768 resolution. Another set need to have their video ram set to 8 megs, and a third set which auto-detect everything correctly. We don't need to specify anything for the third set, but having some symbolic names for the first two would help us to maintain the `lts.conf` file.

Here's an example `lts.conf` that illustrates how this would be done:

```
[Lab]
    X_COLOR_DEPTH = 16
    X_MODE_0 = 1024x768

[Lowram]
    X_VIDEO_RAM = 8096

[00:40:32:71:77:A1]
    LIKE = Lab

[00:70:84:BB:27:52]
    LIKE = Lowram
```

As you can see, using the `LIKE` keyword can make your `lts.conf` more readable, by allowing you to group related parameters together into a single symbolic name.

Chapter 10

General thin client parameters

There are several variables that one can define in the lts.conf file which control how the thin client interacts with the server. These are:

CONFIGURE_FSTAB boolean, default *True*

/etc/fstab is generated by boot scripts

FSTAB_0..FSTAB_9 string, default *unset*

Complete lines to add to /etc/fstab, for example:

```
FSTAB_1="server:/home    /home           nfs     defaults,nolock 0       0"
```

CRONTAB_01..CRONTAB_10 string, default *unset*

A crontab line to add for a thin client.

DNS_SERVER IP address, default *unset*

A valid IP for domain name server Used to build the client's resolv.conf file. Not needed by default.

SEARCH_DOMAIN string, default *unset*

sets a valid search domain in the clients's resolv.conf file. Used to build the resolv.conf file. Not needed by default.

Needed if `DNS_SERVER` is set

HOSTNAME string, default *unset*

This parameter sets the host name for the thin client, for situations when if no DNS is available. A hostname is auto-generated if no hostname is set.

HOSTNAME_BASE string, default *ltsp*

This parameter sets the base for the autogenerated host name for the thin client.

HOSTNAME_EXTRA string, default *ip*

This parameter determines weather autogenerated host names are appended with information based on the ip address or mac address. Values are "ip" or "mac".

NBD_SWAP boolean, default *False*

Set this to **True** if you want to turn on NBD swap.

If unspecified, it's automatically enabled for thin clients with less than 300 MB RAM and for fat clients with less than 800 MB RAM.

NBD_SWAP_PORT integer, default *10809*

The port on which NBD swapping will occur. An nbd-server export named `swap` is normally used.

NBD_SWAP_SERVER IP address, default *SERVER*

The NBD swap server can exist on any server on the network that is capable of handling it. You can specify the IP address of that server. The default is whatever the value of `SERVER` set to.

NBD_SWAP_THRESHOLD integer, default *300*

Automatically enable `NBD_SWAP` if the client has less RAM than the specified. For FAT clients, it defaults to 800.

RM_SYSTEM_SERVICES string, default *unset*

A space separated list of services that shouldn't start on the clients even if they're installed, for example:

```
RM_SYSTEM_SERVICES="apache2 dnsmasq mysql nbd-server nfs-kernel-server"
```

RM_THIN_SYSTEM_SERVICES string, default *unset*

Same as `RM_SYSTEM_SERVICES`, but it only affects thin clients.

KEEP_SYSTEM_SERVICES string, default *unset*

Some services are deleted by default when an LTSP client boots, either to save RAM, or because they don't make sense for netbooted machines. If you need some of them you can list them in `KEEP_SYSTEM_SERVICES`, for example:

```
KEEP_SYSTEM_SERVICES="acpid avahi-daemon bluetooth cups"
```

SERVER IP address, default *unset*

This is the server that is used for the `XDM_SERVER`, `TELNET_HOST`, `XFS_SERVER` and `SYSLOG_HOST`, if any of those are not specified explicitly. If you have one machine that is acting as the server for everything, then you can just specify the address here and omit the other server parameters. If this value is not set, it will be auto detected as the machine that the thin client booted from.

SYSLOG_HOST IP address, default *unset*

If you want to send logging messages to a machine other than the default server, then you can specify the machine here. If this parameter is NOT specified, then it will use the `SERVER` parameter described above. Starting from LTSP 5.4.1 and on, this parameter must be specified to enable remote logging.

You have to configure your server to accept remote logging as well.

USE_LOCAL_SWAP boolean, default *False*

If you have a hard drive installed in the thin client, with a valid swap partition on it, this parameter will allow the thin client to swap to the local hard drive.

TIMEZONE string, default *unset*

The timezone code for the thin client to use.

TIMESERVER IP address, default *unset*

The address of an NTP time server that the thin client can set it's time from. If unset, the thin client just uses the BIOS time.

SHUTDOWN_TIME string, format hh:mm:ss in 24 hour format, default *unset*

Time at which thin client will automatically shut down.

LTSP_FATCLIENT boolean, default *unset*

Enable Fat Client support. It's automatically enabled if any sessions exist in /usr/share/xsessions.

FAT_RAM_THRESHOLD integer, default *300*

Disable fat client support if less RAM is present.

Chapter 11

Local Device thin client parameters

Local devices such as USB sticks, CD-ROM drives, or even floppy disks need special configuration in order to be accessed from the thin client. The following values allow to enable or disable the use of various local devices:

LOCALDEV boolean, default *True*

This parameter enables local devices support, like CD's and USB sticks. Users plugging them in should see them on the desktop, after they've been allowed to access the FUSE subsystem on the server. Check your distibutions docs to see how this is done on your distribution.

LOCALDEV_DENY_CD boolean, default *False*

This parameter disables local device support for CD and DVD-rom devices.

LOCALDEV_DENY_FLOPPY boolean, default *False*

This parameter disables local device support for floppy devices.

LOCALDEV_DENY_INTERNAL_DISKS boolean, default *True*

This parameter disables local device support for internal ATA and SCSI hard disk devices.

LOCALDEV_DENY_USB boolean, default *False*

This parameter disables local device support for USB devices.

LOCALDEV_DENY string, default *unset*

This parameter disables local device support for devices matching certain patterns. Values are specified as a comma-separated list of sysfs attributes, which can be obtained by using udevadm info (or udevinfo). for example:

```
udevadm info -q env -n /dev/hda
ID_TYPE=disk
ID_BUS=ata
```

should return a list of the attributes relevent to /dev/hda. to exclude this disk and disks like it using LOCALDEV_DENY: LOCALDEV_DENY="ID_BUS:ata+ID_TYPE:disk" would match devices that were on the ata bus that were disks.

Chapter 12

Screen Scripts

Screen scripts are how LTSP determines what type of login will run on what virtual screen. Most GNU/Linux machines have 12 virtual consoles, which you can access by pressing Control-Alt-F1, through Control-Alt-F12. On some distributions there is a text based getty that is started on screen 1, but you normally can't log into it, as there are no local users on the thin client.

However, for debugging purposes, you may want to set up root to log in on the thin client. You may need to do this if you're debugging problems with local devices, for example. Fortunately, it's easy to do: on the server, as root, just chroot into the LTSP chroot, and set the password with passwd.

```
chroot /opt/ltsp/<arch>
passwd
```

By default, if there's nothing else mentioned in `lts.conf`, an LDM session will be started on screen 7.

`SCREEN_01...SCREEN_12` string, default *ldm*

Up to 12 screen scripts can be specified for a thin client. This will give you up to 12 sessions on the thin client, each accessible by pressing the Ctrl-Alt-F1 through Ctrl-Alt-F12 keys.

Currently, possible values include: kiosk, ldm, menu, rdesktop (deprecated), shell, ssh, startx (deprecated), telnet, xdmcp, xfreerdp, xterm

Look in the `$CHROOT/usr/share/ltsp/screen.d` directory for more scripts, or write your own, and put them there.

`TELNET_HOST` IP address, default *unset*

If the thin client is setup to have a character based interface, then the value of this parameter will be used as the host to telnet into. If this value is NOT set, then it will use the value of `SERVER` above.

Chapter 13

The "rdesktop" screen script

LTSP includes an **rdesktop** screen script that can be used to bring up a full screen RDP connection using **rdesktop** on the thin client. This screen script can be invoked in one of two ways.

13.1 Single Line

For example, by adding the following `lts.conf` parameter:

```
SCREEN_07 = "rdesktop -a 16 192.168.0.253"
```

That is, calling it with the arguments you normally would on the command line. This method of invocation is useful in that you can have multiple screens pointing to different RDP servers with different arguments and switch between them.

13.2 Multiple Line

For example, by adding the following parameters to `lts.conf`:

```
SCREEN_07 = rdesktop
RDP_SERVER = 192.168.0.253
RDP_OPTIONS = "-a 16"
```

This method will apply the same arguments and server to all screens.

13.3 RDP + Local Devices

When you run an **rdesktop** screen script, **rdesktop** runs on the thin client. The thin client, by default, has no way of automounting removable devices, and the normal localdev approach used in an **ldm**ldm; session in which local devices invoke a call over ssh to have the Linux server mount the device obviously won't work.

To address this issue in a controlled way, we have chosen to use **ltspfs** as a local automounter for RDP sessions. To add local device support, you must:

1. Install ltspfs in the chroot.

2. Use folder redirection in **rdesktop** to map the local /media/root folder created by ltspfs to the server as a shared drive. For example, you could add the following **rdesktop** argument:

   ```
   -r disk:Drives=/media/root
   ```

   With this redirection, you should get a "Drives" share in Windows under My Computer. Inside the "Drives" share, a folder will appear for each local device. The local device will be mounted with ltspfs, so they can just be removed when the device is not being written to without "unmounting".

13.4 RDP + Local Sound

You should be able to add sound to your RDP session with the following **rdesktop** argument:

```
-r sound:local
```

Chapter 14

Modules and startup scripts

For the most part, LTSP does a very good job of detecting what hardware's on your thin client. However, it's possible that you may want to manually specify a kernel module to load after boot. Alternatively, you may have a script you've written that you've put in the chroot, and want to make sure gets run at startup. LTSP provides some hooks to allow you to do this.

`MODULE_01...MODULE_10` string, default *unset*

Up to 10 kernel modules can be loaded by using these configuration entries. The entire command line that you would use when running insmod can be specified here. For example:

```
MODULE_01 = uart401.o
MODULE_02 = "sb.o io=0x220 irq=5 dma=1"
MODULE_03 = opl3.o
```

If the value of this parameter is an absolute path name, then **insmod** will be used to load the module. Otherwise, **modprobe** will be used.

In normal circumstances, you shouldn't need to specify anything here, as most hardware will be auto-detected.

`RCFILE_01...RCFILE_10` string, default *unset*

Commands to be executed from `/etc/rc.local` when the client boots.

Chapter 15

Sound in LTSP

Sound in LTSP is handled by running the **pulseaudio** daemon on the thin client, which sits on top of the ALSA kernel drivers. The thin client's kernel should detect the thin client sound hardware via the usual udev mechanisms, and enable the sound card. At boot time, the **pulseaudio** daemon is run, which allows the thin client to receive audio streams via network connections.

On login, the LDM sets both the `PULSE_SERVER` and `ESPEAKER` environment variables for the X windows session, to allow the server to re-route the sound over a TCP/IP socket to the thin client.

SOUND boolean, default *True*

This parameter enables sound for the thin client.

SOUND_DAEMON string, default *pulse*

This parameter sets which sound daemon to use on the thin client. Values are esd, nasd, and pulse (default).

VOLUME integer, default *90*

This represents an integer percentage of the volume, ranging from 0 to 100%.

HEADPHONE_VOLUME integer, default *unset*

This represents an integer percentage of the headphone volume, ranging from 0 to 100%.

PCM_VOLUME integer, default *unset*

This represents an integer percentage of the PCM volume, ranging from 0 to 100%.

CD_VOLUME integer, default *unset*

This represents an integer percentage of the CD input volume, ranging from 0 to 100%.

FRONT_VOLUME integer, default *unset*

This represents an integer percentage of the front speaker volume, ranging from 0 to 100%.

MIC_VOLUME integer, default *unset*

This represents an integer percentage of the microphone input volume, ranging from 0 to 100%.

Chapter 16

X-Windows parameters

Setting up X windows on the thin client's normally a pretty easy operation. The thin client uses X.org's own auto configuration mode to let X determine what it thinks is installed in the box.

However, sometimes, this doesn't always work. Either due to strange/buggy hardware, or buggy drivers in X.org, or because X detects default settings that you don't want. For instance, it may detect that your monitor is capable of doing 1280x1024, but you'd prefer it to come up in 1024x768 resolution. Fortunately, you can tweak individual X settings, or, alternatively, simply provide your own `xorg.conf` to use.

16.1 X.org Configuration

`USE_XFS` boolean, default *False*

Instructs the thin client to look at the `XFS_SERVER` option, and use XFS for serving fonts.

`XFS_SERVER` IP address, default *unset*

If you are using an X Font Server to serve fonts, then you can use this entry to specify the IP address of the host that is acting as the font server. If this is not specified, it will use the default server, which is specified with the `SERVER` entry described above.

`CONFIGURE_X` boolean, default *False*

If you want to be able to configure the individual settings of the X configuration file, without having the X automatically configure the graphics card for you, you must enable this option. By default this option is turned off. To turn it on do:

```
CONFIGURE_X = True
```

You don't need this option just for keyboard and mouse settings. It corresponds to the graphic card and monitor options only.

`X_CONF` string, default *unset*

If you want to create your own complete X.org config file, you can do so and place it in the `/opt/ltsp/<arch>/etc/X11` directory. Then, whatever you decide to call it needs to be entered as a value for this configuration variable. For example: X_CONF = /etc/X11/my-custom-xorg.conf Note that for the thin client, you reference it from `/etc/X11`.

X_RAMPERC integer, default *100*

Percentage of RAM for X server. Some programs allocate a large amount of ram in the X.org server running on your thin client. Programs like **Firefox** and **Evince** can use up so much ram, that they eventually exhaust all your physical ram, and NBD swap, causing your thin client to crash. If you find your clients being booted back to a login prompt, or freezing up when viewing certain PDF's or web pages, this may be the problem.

The `X_RAMPERC` variable stands for X RAM PERCent, and is a number between 0 and 100 that specifies how much of the free space on your thin client X.org is allowed to consume. You'll generally want to set it at something lower than 100 percent, if you're having problems. Experimentation has shown a value between 80 and 90 will usually keep the terminal alive. What will then happen is the program consuming the memory will die, as opposed to the thin client itself. If you're having unexplained terminal problems, specifying:

X_RAMPERC = 80

in your `lts.conf` file may improve things.

X_VIRTUAL string, default *unset*

If you want to have a virtual screen which is larger than the physical screen on your thin client, you would configure that by providing a string of the form "width height" in this parameter, similar to the `xorg.conf` format.

XDM_SERVER IP address, default *unset*

If you're using the older **startx** screen script, and need to specify a different XDMCP server, then you can specify the server here. If this parameter is NOT specified, then it will use the `SERVER` parameter described above.

XSERVER string, default *unset*

You can use this parameter to override which X server the thin client will run. For PCI and AGP video cards, this parameter should not be required. The thin client should normally be able to auto-detect the card.

If, for some reason you do need to manually set it, here are some valid values:

ark, ati, atimisc, chips, cirrus_alpine cirrus, cirrus_laguna, cyrix, dummy, fbdev fglrx, glint, i128, i740, i810, imstt, mga, neomagic, newport, nsc, nv, r128, radeon, rendition, riva128, s3, s3virge, savage, siliconmotion, sis, sisusb, tdfx, tga, trident, tseng, v4l, vesa, vga, via, vmware, voodoo

X_MOUSE_DEVICE string, default *unset*

This is the device node that the mouse is connected to. If it is a serial mouse, this would be a serial port, such as `/dev/ttyS0` or `/dev/ttyS1`. This is not needed for PS/2 or USB mice, as they are auto-detected.

X_MOUSE_PROTOCOL string, default *unset*

Should be auto-detected. However, valid entries include:

sunkbd, lkkbd, vsxxxaa, spaceorb, spaceball, magellan, warrior, stinger, mousesystems, sunmouse, microsoft, mshack, mouseman, intellimouse, mmwheel, iforce, h3600ts, stowawaykbd, ps2serkbd, twiddler, twiddlerjoy

X_MOUSE_EMULATE3BTN boolean, default *unset*

Normally unset, may need to be set to *Y* for certain 2 button mice.

X_NUMLOCK boolean, default *False*

If this variable is set to *True*, then the numlock key will be defaulted to on when the terminal boots. Note that the **numlockx** command must be installed in the chroot for this to work.

X_COLOR_DEPTH Integer, default *unset*

This is the number of bits to use for the colour depth. Possible values are **8**, **16**, **24** and **32**. 8 bits will give 256 colours, 16 will give 65536 colours, 24 will give 16 million colours and 32 bits will give 4.2 billion colours! Not all X servers support all of these values. The default for thin clients is 16 in order to minimize network bandwidth, while for fat clients the X server default is used.

X_SMART_COLOR_DEPTH boolean, default *True*

If set, thin clients no longer default to 16 bit colour depth but use the X server default instead.

X_HORZSYNC min-max, default *unset*

This sets the X.org **HorizSync** configuration parameter. This should be auto-detected for your monitor, however, if you want to force a lower resolution, use this parameter to do so.

X_VERTREFRESH min-max, default *unset*

This sets the X.org **VertRefresh** configuration parameter. This should be auto-detected for your monitor. If you need to force a lower resolution, use this parameter to do so.

X_VIDEO_RAM string, default *unset*

This sets the X.org **VideoRam** configuration parameter. The setting is in kilobytes. This should be auto-detected for your monitor. If you need to force a different video ram setting, use this parameter to do so.

X_OPTION_01...X_OPTION_12 string, default *unset*

A valid Device option. This allows you to specify **Option** settings in the `xorg.conf` file, to add options to the video driver. A common use for this will be to test turning off acceleration in your driver, if you're having trouble. An example usage would be:

```
X_OPTION_01 = "\"NoAccel\"" X_OPTION_02 = "\"AnotherOption\" \"True\""
```

You probably won't need these except in special circumstances.

X_MONITOR_OPTION_01...X_MONITOR_OPTION_10 string, default *unset*

A valid Monitor option, that would normally be used in an xorg.conf file.

X_MODE_0...X_MODE_2 string, default *unset*

These set the X.org `ModeLine` configuration. For example, if your thin client comes up in a higher resolution than what you want, say, 1280x1024, specifying:

```
X_MODE_0 = 1024x768
```

should get your desired resolution on startup.

X_MODE_* require XRANDR_DISABLE=True to work. For drivers that support XRANDR, the XRANDR_MODE_ variabled are preferred. See the XRANDR section.

X_BLANKING integer, default *unset*

When set, `X_BLANKING` will cause DPMS standby to activate after the number of seconds provided. If the monitor does not support DPMS, then the blanking screensaver will activate. If `X_BLANKING` is set to 0, the monitor will remain on indefinitely. NOTE: This does not apply to the **xdmcp** or **startx** screen script. Also, server-side Xclients such as power managers and screensavers may override this setting.

Chapter 17

XRANDR setting for managing displays

The new Xorg Xserver has the ability to figure out (for the most part, to the extent that the driver helps in the process) the best mode for the videocard. Moreover, with the new dependency upon hal and Xrandr, it is recommended to add input devices with hal and modify video modes with Xrandr 1.2 calls. In essence, the xorg.conf becomes a place really to fix deficiencies in poorly written drivers or to force certain abnormal driver behavior in a particular environment in a way that can not be otherwise done through hal or Xrandr.

17.1 New Xorg structure within LTSP

To accommodate this, Xorg now understands partial xorg.conf files. Meaning you only add the sections that you need to force. Otherwise, it discovers everything. That's why you might see minimalist xorg.conf files in your LTSP chroot.

The `screen-session.d/` directory (located in the chroot's `/usr/share/ltsp directory`) is a structure of shell scripts all of which are sourced in order (similar to `Xsession.d/` or `rc.d/` that you may be familiar with). These scripts are executed upon the beginning of each session but before the Xserver (if the session runs an Xserver) is launched. You can make whatever script you want that may need to run at that point. For LTSP, one thing we use it for is to set up *how* the Xserver will be launched. This entails not just generating a `xorg.conf` file as needed, but also configuring the parameters that the Xserver should be launched with. The nice thing about a collection of sourced scripts is that it gives flexibility to the distribution or to the administrator to add additional scripts that may be required for that distribution or for a particular network environment that will not modify existing files (and therefore require more maintenance to care for updates in the upstream code).

17.2 Script structure

Each script is named with a prefix letter, then an order number, then a name. The prefix letter determines when the scripts of that prefix are executed and the order number determines in what order.

PREFIXES: Prefixes that may be used include:

S - Is a script that runs at the beginning of a session (screen script) K - Is a script that runs at the end of a session (screen script) XS - Is a script that is only run at the beginning of screen scripts that run an Xserver

All of the scripts that generate a xorg.conf or modify the Xserver arguments are XS* scripts.

These scripts are mostly organized by the particular lts.conf parameter or function that they affect. For example, XS85-xvideoram adds the ability to specify the X_VIDEO_RAM parameter in lts.conf and force the amount of video ram used by the driver.

If you are going to create your own script, I recommend looking at other scripts to understand the structure. Since many hacks may impact the same xorg.conf sections, each section has a function of hacks assigned to it, and in your script, you would create a function and add it to the list of functions for that section. For example, if you add something to the Monitor section (that cannot already be added through existing functions) you would create a function in your script and add it to the monitor_hacks function list. Again, easier to read the code and look at examples to understand how to write a new script.

Also, please note that one of the lts.conf parameters you can specify is: `CONFIGURE_X_COMMAND` This should be set to a path to a script. So, if you have the old configure-x.sh and like it better, simply copy it into the chroot, to say, **/opt/ltsp/<arch>/usr/share/ltsp/configure-x.sh** and then in `lts.conf`, specify:

```
CONFIGURE_X_COMMAND =
                    "/usr/share/ltsp/configure-x.sh"
```

and you will be back to where you were.

17.3 XRandR parameters

XRANDR_COMMAND_0 ...XRANDR_COMMAND_9 string, default *unset*

Full xrandr command to run when X starts. They're useful to define and add custom modes, for example:

```
XRANDR_COMMAND_0="xrandr --newmode 1024x600 49.00 1024 1072 1168 1312 600 603 ↩
    613 624 -hsync +vsync"
XRANDR_COMMAND_1="xrandr --addmode VGA1 1024x600"
XRANDR_COMMAND_2="xrandr --output VGA1 --mode 1024x600"
```

You can use **cvt** to find the correct timings for new modes.

XRANDR_DISABLE boolean, default *False*

Disables XRANDR output handling so that the older `X_MODE_0` way of setting resolution works. This is useful on older Xorg drivers that don't support XRANDR.

XRANDR_OUTPUT_0 ...XRANDR_OUTPUT_8 string, default *unset*

Define xrandr output - can also be used for multihead positioning

XRANDR_MODE_0 ...XRANDR_MODE_8 string, default *unset*

Valid video mode resolution. Sets mode for corresponding output.

XRANDR_NEWMODE_0 ...XRANDR_NEWMODE_8 string, default *unset*

Specifies a valid modeline for a corresponding output.

XRANDR_RATE_0 ...XRANDR_RATE_8 string, default *unset*

Sets refresh rate for the corresponding output.

XRANDR_DPI_0 ...XRANDR_DPI_8 string, default *unset*

Sets the DPI for the corresponding output.

XRANDR_ROTATE_0 ...XRANDR_ROTATE_8 string, default *unset*

Sets the rotation for the corresponding output.

XRANDR_REFLECT_0 ...XRANDR_REFLECT_8 string, default *unset*

Sets the reflection for the corresponding output.

XRANDR_SIZE_0 ...XRANDR_SIZE_8 string, default *unset*

Sets the resolution for the corresponding output (for xrandr <1.2).

XRANDR_ORIENTATION_0 ...XRANDR_ORIENTATION_8 string, default *unset*

Sets the orientation for the corresponding output (for xrandr <1.2).

Chapter 18

Printer configuration parameters

Sometimes, it's convenient to hang a printer off of a thin client in a lab, so that the computer lab has access to local printing resources. Fortunately, LTSP can accommodate printing on the workstation.

LTSP can connect up to 3 printers per workstation to the network via a small daemon called JetPipe. Both parallel and USB printers are supported. JetPipe makes the printer look like a standard HP Jet Direct printer interface. You can then create any cups printer on your server, and point it at the printer via a Jet Direct connection.

In your `dhcpd.conf` file that controls your thin client IP assignments, you'll want to assign a static IP for the terminal with the printers, to guarantee that it gets the same IP address every time it boots. Otherwise, your printing won't work if the terminal leases a different IP address.

`PRINTER_0_DEVICE` string, default *unset*

The device name of the printer. Valid device names such as `/dev/lp0`, or `/dev/usblp0` are allowed.

`PRINTER_0_PORT` integer, default *9100*

The TCP/IP Port number to use for the print server.

`PRINTER_0_TYPE` string, default *unset*

Can either be set to P (for parallel), U (for USB) or S (for serial). Autodetected in most cases (except for serial).

`PRINTER_0_WRITE_ONLY` boolean, default *False*

Some parallel printers may need this set in order for the thin client to communicate to them properly. If you have problems with a parallel printer only printing part of the print job, try setting this to *True*.

`PRINTER_0_SPEED` integer, default *9600*

Should be set to the baud rate of the printer (serial printers only).

`PRINTER_0_FLOWCTRL` string, default *unset*

Should be set to the flow control desired for the printer (serial printers only).

`PRINTER_0_PARITY` boolean, default *False*

Specifies whether parity should be enabled for the printer (serial printers only).

`PRINTER_0_DATABITS` integer, default *8*

Specifies how many data bits for the printer (serial printers only).

PRINTER_0_OPTIONS string, default *unset*

Specifies specific options for the printer (serial printers only).

LDM_PRINTER_LIST string, default *unset*

Comma separated list of printers that will be displayed for that thin client (requires patched cups, included in Debian and Ubuntu).

LDM_PRINTER_DEFAULT string, default *unset*

Default printer for the thin client.

SCANNER boolean, default *unset*

This parameter enables scanners for the thin client.

Chapter 19

Keyboard parameters

All of the keyboard support files are copied into the `/opt/ltsp/<arch>` hierarchy, so configuring international keyboard support is simply a matter of configuring X.org. There are several configuration parameters for this.

The values for the above parameters are from the X.org documentation. Whatever is valid for X.org is valid for these parameters.

We would like to add documentation to show what values are needed for each type of international keyboard. If you work with this and can configure your international keyboards, feedback to LTSP would be greatly appreciated.

CONSOLE_KEYMAP string, default *en*

A valid console keymap. Allows you to specify a valid console keymap for TELNET_HOST sessions.

XKBLAYOUT string, default *unset*

A valid xkb layout. Consult the X.org documentation for valid settings.

XKBMODEL string, default *unset*

A valid xkb model. Consult the X.org documentation for valid settings.

XKBVARIANT string, default *unset*

A valid xkb variant. Consult the X.org documentation for valid settings.

XKBRULES string, default *unset*

A valid xkb rules specifier. Consult the X.org documentation for valid settings.

XKBOPTIONS string, default *unset*

A valid xkb options specifier. Consult the X.org documentation for valid settings.

Chapter 20

Touchscreen configuration

Description to be added later.

USE_TOUCH Enable touchscreen

default <unset>, Enable touchscreen

X_TOUCH_DEVICE	Path to device	/dev/ttyS0	set device for touchscreen
X_TOUCH_DRIVER	Touchscreen driver	elographics	set driver for touchscreen
X_TOUCH_MAXX	integer	3588	Xmax
X_TOUCH_MAXY	integer	3526	Ymax
X_TOUCH_MINX	integer	433	Xmin
X_TOUCH_MINY	integer	569	Ymin
X_TOUCH_UNDELAY	integer	10	Untouch delay
X_TOUCH_RTPDELAY	integer	10	Repeat touch delay

Chapter 21

Local Applications

Description to be added later.

LOCAL_APPS boolean, default *True*

Enables support for running local apps on the thin client.

LOCAL_APPS_EXTRAMOUNTS string, default *unset*

This parameter enables extra mount points to be mounted on the thin client with sshfs. This require a commas separated list of directory.

LOCAL_APPS_MENU boolean, default *False*

Enables overriding of menu items from remote (server) applications. If this is set to *True*, local applications in the users menu will be used instead of the applications on the server.

LOCAL_APPS_MENU_ITEMS string, default *unset*

This item should contain a comma separated list of application names as they appear on their .desktop files.

LOCAL_APPS_WHITELIST string, default *unset*

Used to allow only specified space-separated commands to be run as local apps, allow all is default if unset. Full-paths are required for each command. No spaces in the names are allowed.

Chapter 22

The LDM display manager

22.1 Introduction

The LTSP Display Manager, or **ldm**ldm; is the display manager specifically written by the LTSP project to handle logins to a GNU/Linux server. It is the default display manager for LTSP thin clients running under LTSP, and has a lot of useful features:

1. It is written in C, for speed and efficiency on low end clients.
2. It supports logging in via either a greeter (a graphical login application) or autologin.
3. It can be configured to encrypt X Windows traffic, for increased security, or leave it unencrypted, for better performance on slower clients.
4. It contains a simple load-balancing system, to allow the system administrator to allow load balancing across several servers.

We'll go over the `lts.conf` entries you'll need to control these features below.

22.2 Theory of operation

To help understand the following sections, a bit of an explanation of how **ldm**ldm; does it's work is needed. Most thin client display managers tend to run up on the server. The **ldm**ldm; display manager is unique in that it runs on the thin client itself. This allows the thin client to have a lot of choice as to how it will set up the connection. A typical login session goes as follows:

1. **ldm**ldm; launches and starts up the X Windows display on the thin client.
2. **ldm**ldm; starts up the greeter, which is a graphical program which presents the user with a nice login display and allows them to select their session, language, and host they'd like to log into.
3. **ldm**ldm; collects the information from the greeter, and starts an ssh session with the server. This ssh connection is used to create an ssh master socket, which is used by all subsequent operations.
4. Now, the users selected session is started via the master socket. Depending on whether or not an encrypted connection has been requested, via the `LDM_DIRECTX` parameter, the session is either connected back to the local display via the ssh tunnel, or via a regular TCP/IP connection.

5. During the session, any memory sticks, or other local devices that are plugged in, communicate their status to the server via the ssh control socket.

6. When the user exits the session, the ssh connection is closed down, the X server is stopped, and **ldm**ldm; restarts itself, so everything starts with a clean slate.

22.3 Encrypted versus unencrypted sessions

By default, LTSP5 encrypts the X session between the server. This makes your session more secure, but at the cost of increased processing power required on the thin client and on the server. If processing power is a concern to you, it's very easy to specify that the connection for either an individual workstation, or the default setting should use an unencrypted connection. To do so, simply specify:

```
LDM_DIRECTX = True
```

in your `lts.conf` file in the appropriate section.

22.4 Auto login features

This new version of LDM supports auto login of accounts, if specified in the `lts.conf` file. Simply create a config section for each of the terminals you want to log in automatically (you can use either MAC address, IP address, or hostname) and specify the variable `LDM_USERNAME` and `LDM_PASSWORD`. Note that you must have created these accounts on the server, with the passwords specified. An example will serve to illustrate how to use this:

22.5 Load balancing features

In this version of LTSP, there's a simple load-balancing solution implemented that allows administrators to have multiple LTSP servers on the network, and allow the thin client to pick which one of the servers it would like to log into.

The host selection system is simple and flexible enough to allow administrators to implement their own policy on how they want the load balancing to happen: either on a random, load-based, or round robin system. See for details.

22.6 RC script capabilities

LDM has a very good system for handling user-supplied rc.d scripts. This allows people looking to add site-specific customizations to their LTSP setups an easy way to integrate this functionality into LTSP.

These rc.d scripts can be placed in `$CHROOT/usr/share/ldm/rc.d/`. They are executed in the usual rc.d type method, so you must make sure that any script you write will not make a call to **exit**.

The files start with the letter I, S, K, or X, and have two digits after them, allowing you to place them in order of execution. The letters stand for:

- *I* scripts are executed at the start of LDM, before the greeter has been presented.
- *S* scripts are executed after the user has logged in, but before the X session is run.
- *X* scripts are executed while the X session is being executed.

- *K* scripts are executed after the X session has ended, but before the user logs out entirely.

Your scripts can make use of the following environment variables in the S, X, and K scripts:

LDM_USERNAME This is the username the user supplied at login.

LDM_SOCKET The path to the ssh control socket that LDM has open for communication with the server.

LDM_SERVER The current server that LDM is connected to.

LDMINFO_IPADDR The IP address of the thin client.

You can use these variables to create scripts that customize behaviors at login time. For instance, lets say you were running the GNOME desktop environment, and wanted to force your users to have blank-only mode for their screen savers, to save network bandwidth.

Since the script is actually running *on the thin client itself*, you want this script to set this up on the server, where the Gnome session is running. That's where you can make use of the `LDM_SOCKET` and `LDM_SERVER` environment variables to run an **ssh** command on the server, using the control socket that LDM has set up. Here's an example script. You could install this into `$CHROOT/usr/share/ldm/rc.d/X01-set-blankonly`:

```
#
# sourced with .
#
# Script to automatically switch gnome screensaver to blank
# only mode
#
ssh -S ${LDM_SOCKET} ${LDM_SERVER} "/usr/bin/gconftool-2 --set --type string /apps ↩
   /gnome-screensaver/mode blank-only"
```

Using this mechanism, it's easy to customize your LTSP setup to your needs.

22.7 LDM lts.conf parameters

LDM_AUTOLOGIN boolean, default *False*

This option allows the thin client to login automatically without the need for a username and password. To set it set

```
LDM_AUTOLOGIN = True
```

for the corresponding thin client. This will attempt to log in the thin client with username = hostname and password = hostname. You can also set a user and password with `LDM_USERNAME` and `LDM_PASSWORD` variables.

LDM_DEBUG_TERMINAL boolean, default *False*

Opens a local terminal after login for debugging purposes.

LDM_DIRECTX boolean, default *False*

This is arguably the most important LDM option, as it allows you to turn off the encrypted X tunnel via SSH, and instead run a less secure, but much faster unencrypted tunnel. Users who have slower thin clients will want to set this to *True*. It is set to *True* by default in Fedora.

LDM_GUESTLOGIN boolean, default *False*

This option places a GUEST LOGIN button underneath the entry field for username and password. To set it set

```
LDM_GUESTLOGIN = True
```

for the corresponding thin client. You can also set a user and password with

```
LDM_USERNAME = John
```

and

```
LDM_PASSWORD = secret
```

, although not setting these will default to the hostname of the thin client.

LDM_GUEST_SERVER string, default *unset*

This is a space-separated list of available servers where guest logins are available. The first server in the list will be the default guest login server unless the user selects another from the preferences menu at login time.

LDM_USER_ALLOW string, default *unset*

This option allows you to give access to certain thin clients based on the username set in `/etc/passwd`. For example, thin client A should only be used by Jane, Bob, and Fred, while thin client B is to be used by Harry only. By adding these options to the corresponding mac addresses you allow or deny access to the thin clients in question. Example:

```
[thin:client:A:mac:address]
LDM_USER_ALLOW = Jane,Bob,Fred
[thin:client:B:mac:adddress]
LDM_USER_ALLOW = Harry
```

LDM_LOGIN_TIMEOUT integer

This lets LDM automatically login after the set amount of time in seconds. If you specify this option, then do not specify LDM_AUTOLOGIN. Use it in this format:

```
[thin:client:mac:address]
LDM_LOGIN_TIMEOUT = 25
```

Note that you will need to also set LDM_GUESTLOGIN=True for LDM_LOGIN_TIMEOUT to be useful.

LDM_USERNAME string, default *unset*

This is the username that LDM will use for autologin.

LDM_PASSWORD string, default *unset*

This is the password that LDM will use for autologin.

LDM_SYSLOG boolean, default *False*

Normally, LDM logs to a simple file on the thin client, namely `/var/log/ldm.log`. This has the advantage of being fast, but the disadvantage of being hard to read for the administrator in the event of a problem, as the administrator must either spawn a shell screen session, or enable root login in tty1. By setting this option to "True", you can log up to the server *if you've enabled your server's syslog for remote logging.*

LDM_SERVER string, default *unset*

This is a space-separated list of available servers for LDM to log into. The first server in the list will be the default server unless the user selects another from the preferences menu at login time.

LDM_LANGUAGE string, default *unset*

This allows the system administrator to override the default locale settings on the server by setting the environment variables LANG, LANGUAGE and LC_ALL at login.

Use the LANG variable to set the default locale for LDM's user interface.

LDM_FORCE_LANGUAGE string, default *unset*

Same as `LDM_LANGUAGE`, but it overrides any previously user selected language and it even hides the LDM language selection menu.

LDM_SSHOPTIONS string, default *unset*

Allows you to specify custom options to the ssh sessions started between LDM and the server.

SSH_OVERRIDE_PORT integer, default *unset*

If you run your ssh server different from the defult, you may set the port the thin client will use with this parameter.

SSH_FOLLOW_SYMLINKS boolean, default *unset*

Causes `sshfs` mounted filesystems for local applications to follow symlinks. By default it's false for symlinks under $HOME and true for any `LOCAL_APPS_EXTRAMOUNTS`.

LDM_SESSION string, default *unset*

Used to chose the default session on the server, for example:

```
LDM_SESSION="gnome-fallback"
```

You can find the list of services that your server provides in `/usr/share/xsessions`.

If the user has selected a specific session though LDM or another DM in the past, that's stored in his `~/.dmrc`, and it overrides the default session.

LDM_FORCE_SESSION string, default *unset*

Same as `LDM_SESSION`, but it overrides any previously selected sessions by the user and it even hides the LDM session selection menu.

LDM_XSESSION string, default *Xsession*

Allows you to specify custom script on the server for LDM to run, rather than the server's standard script for starting an X session (usually **Xsession**).

LDM_LIMIT_ONE_SESSION boolean, default *False*

Only allow a given user to log into one thin-client at a time.

LDM_LIMIT_ONE_SESSION_PROMPT boolean, default *False*

Prompt to kill processes of other logins when other logins are detected. Requires LDM_LIMIT_ONE_SESSION to be set.

LDM_THEME string, default *unset*

Specify the name of the LDM theme. It can reference a directory in /usr/share/ldm/themes, or be specified as a full path to the theme dir (both relative to the chroot).

To use the theme in `/opt/ltsp/i386/usr/share/ldm/themes/MYTHEME`, you'd specify:

```
LDM_THEME=MYTHEME
```

In your `lts.conf` file.

Alternately, To use the theme in `/opt/ltsp/i386/etc/MYTHEME`, you'd specify:

```
LDM_THEME=/etc/MYTHEME
```

In your `lts.conf` file.

`LDM_PASSWORD_HASH` boolean, default *False*

When set to *True*, this will create a proper shadow entry on the client, allowing for screen locking, and other things which require authentication to work. Note, this allows you to change your password locally, or possibly other actions such as sudo, but these changes are only temporary and will not persist on reboot.

versions: LDM 2.2.14+, LTSP 5.5.2+

22.8 Multiple server setup

A multiple server setup is useful for larger thin client networks. Instead of using one big server, it makes it possible to use smaller servers, and dispatch users on them. You can adjust computing resources as the demand grows simply by adding a new server. To make sure that every server behaves the same from the users point of view, new services and configurations that are required will be discussed. In addition, some configurations specific to thin clients will be presented.

Chapter 23

Infrastructure setup

23.1 Network topology

The network topology is the same as a standalone server setup, except that there are more than one server on the thin client LAN.

You will need to select one server to behave as the primary server. This server will be used to run additional services, hold users files, and network boot thin clients.

Secondary servers will be used only to run desktop sessions. They are simpler, and will be configured to use the central services from the primary server.

23.2 Common authentication

A user should be able to start a session with the same login and password, no matter which server it connects to. For this purpose, a central authentication mechanism must be used. There are many possibilities offered. Here are the major technologies:

1. LDAP authentication: On the master server, setup an OpenLDAP server. Configure each servers to use this LDAP server as the authentication base.
2. NIS authentication: On the master server, setup a NIS server. Configure each server to use this NIS server for the authentication.
3. Winbind authentication: Useful if you already have an Active Directory server.

For detailed instructions, see their respective manuals.

23.3 Shared home directories

Shared home directories are easy to setup using an NFS server on eithe the primary LTSP server, or even better, a standalone NFS server. Other more modern, faster (and consequently more expensive) options include a SAN, and maybe even moving to a fibre-channel raid SAN. Consult your distribution's documentation for details and suggestions for setting up an NFS server.

23.4 Shared printers

For printers to be accessible on each server, the cups server must be configured to share printers. Refer to the CUPS manual for your distribution for detailed setup instructions.

Note that an LTSP thin client acting as a print server resembles a JetDirect print server.

23.5 Managing the SSH known hosts file

For security reasons, a thin client won't connect to an untrusted server. You must add the keys of secondary servers inside the client root on the primary server. To do this, first export the key file of the secondary server using LTSP's tools. As root, run:

```
ltsp-update-sshkeys --export ssh_known_hosts.myhostname
```

Then, copy the file `ssh_known_hosts.myhostnam` to the primary server, in the directory `/etc/ltsp/` and run **ltsp-update-sshkeys** on the primary server. Then, thin clients will trust the freshly added server, and will be able to connect to it.

If a secondary server changes it's IP address, then this procedure must be repeated.

23.6 Setting Network Forwarding or Masquerading

The purpose of IP Masquerading is to allow machines with private, non-routable IP addresses on your network to access the Internet through the machine doing the masquerading. Traffic from your private network destined for the Internet must be manipulated for replies to be routable back to the machine that made the request. To do this, the kernel must modify the *source* IP address of each packet so that replies will be routed back to it, rather than to the private IP address that made the request, which is impossible over the Internet. Linux uses *Connection Tracking* (conntrack) to keep track of which connections belong to which machines and reroute each return packet accordingly. Traffic leaving your private network is thus "masqueraded" as having originated from your gateway machine. This process is referred to in Microsoft documentation as Internet Connection Sharing.

IP Forwarding with IP Tables

1. to enable IPv4 packet forwarding by editing /etc/sysctl.conf and uncomment the following line:

   ```
   net.ipv4.ip_forward=1
   ```

2. If you wish to enable IPv6 forwarding also uncomment:

   ```
   net.ipv6.conf.default.forwarding=1
   ```

3. Next, execute the sysctl command to enable the new settings in the configuration file:

   ```
   sudo sysctl -p
   ```

4. IP Masquerading can now be accomplished with a single iptables rule, which may differ slightly based on your network configuration:

   ```
   sudo iptables -t nat -A POSTROUTING -s 192.168.0.0/16 -o eth0 -j MASQUERADE
   ```

   The above command assumes that your private address space is 192.168.0.0/16 and that your Internet-facing device is eth0. The syntax is broken down as follows:

Chapter 24

Session dispatching

24.1 Define the server list

LDM is a login manager for thin clients. Users can select a server from the available ones in the host selection dialogue box.

The displayed server list is defined by the `LDM_SERVER` parameter. This parameter accepts a list of server IP address or host names, separated by space. If you use host names, then your DNS resolution must work on the thin client. If defined in the `lts.conf` file, the list order will be static, and the first server in the list will be selected by default.

You can also compute a new order for the server list, by creating the script **/opt/ltsp/<arch>/usr/share/ltsp/get_hosts** . The parameter `LDM_SERVER` overrides the script. In consequence, this parameter must not be defined if the **get_hosts** is going to be used. The **get_hosts** script writes on the standard output each server IP address or host names, in the chosen order.

24.2 Dispatching method

You can change this behaviour by using a script to rearrange the list. The simplest way to do it is by randomizing the list. First, define a custom variable in the file `lts.conf` , for example `MY_SERVER_LIST`, that will contain the list of servers, the same way as `LDM_SERVER` Then, put the following script in **/opt/ltsp/<arch>/usr/share/ltsp/get_hosts**

```
#!/bin/bash
# Randomize the server list contained in MY_SERVER_LIST parameter
TMP_LIST=""
SHUFFLED_LIST=""

for i in $MY_SERVER_LIST; do
    rank=$RANDOM
    let "rank %= 100"
    TMP_LIST="$TMP_LIST\n${rank}_$i"
done

TMP_LIST=$(echo -e $TMP_LIST | sort)
for i in $TMP_LIST; do
    SHUFFLED_LIST="$SHUFFLED_LIST $(echo $i | cut -d_ -f2)"
done
echo $SHUFFLED_LIST
```

More advanced load balancing algorithms can be written. For example, load balancing can be done by querying ldminfod for the server rating. By querying ldminfod, you can get the current rating state of the server. This rating goes from 0 to 100, higher is better. Here is an example of such a query:

```
nc localhost 9571 | grep rating | cut -d: -f2
```

24.3 Network Swap

Just like on a full fledged workstation, it helps to have swap defined for your thin client. "Swap" is an area of disk space set aside to allow you to transfer information out of ram, and temporarily store it on a hard drive until it's needed again. It makes the workstation look like it has more memory than it actually does. For instance, if your workstation has 64 Megabytes of ram and you configure 64 Megabytes of swap, it's theoretically possible to load a 128 Megabyte program.

We say, "theoretically", because in practice, you want to avoid swapping as much as possible. A hard drive is several orders of magnitude slower than ram, and, of course, on a thin client, you don't even have a hard drive! You have to first push the data through the network to the server's hard drive, thus making your swapping even slower. In practice, it's best to make sure you have enough ram in your thin client to handle all your average memory needs.

However, sometimes that's not possible. Sometimes, you're re-using old hardware, or you've simply got a program that isn't normally used, but does consume a lot of ram on the thin client when it does. Fortunately, LTSP supports swapping over the network via NBD, or Network Block Devices. We include a small shell script called nbdswapd, which is started via inetd. It handles creating the swap file, and setting up the swapping, and removing the swap file when it's no longer needed, after the terminal shuts down.

By default, swap files are 64 Megabytes in size. This was chosen to give your workstation a little extra ram, but not use up too much disk space. If you get some random odd behaviour, such as Firefox crashing when viewing web pages with a lot of large pictures, you may want to try increasing the size of the swap files. You can do so by creating a file in the directory `/etc/ltsp` on the LTSP server, called `nbdswapd.conf`. In it, you can set the SIZE variable to the number of Megabytes you wish the file to be sized to. For instance, to create 128 Megabyte files, you'll want: SIZE=128 in the `nbdswapd.conf` file.

Please note that this is a global setting for all swap files. If your server has 40 thin clients, each using 128 Megs of memory, you'll need 128 * 40 = 5120, or a little over 5 Gigabytes of space in your `/tmp` directory, where the swap files are stored.

24.4 Managing DHCP

DHCP stands for Dynamic Host Configuration Protocol and is the very first thing your thin client uses to obtain an IP address from the network, in order to allow it to start booting. In LTSP, the dhcpd file is located in `/etc/ltsp`. Any changes you want to make to booting behaviour should be made there.

By default, LTSP ships a `dhcpd.conf` that serves thin clients in a dynamic range (i.e. it will hand out ip addresses to anyone who asks for them) from 192.168.0.20 to 192.168.0.250. The default dhcpd.conf file looks like:

```
#
# Default LTSP dhcpd.conf config file.
#

authoritative;

subnet 192.168.0.0 netmask 255.255.255.0 {
```

```
    range 192.168.0.20 192.168.0.250;
    option domain-name "example.com";
    option domain-name-servers 192.168.0.1;
    option broadcast-address 192.168.0.255;
    option routers 192.168.0.1;
#    next-server 192.168.0.1;
#    get-lease-hostnames true;
    option subnet-mask 255.255.255.0;
    option root-path "/opt/ltsp/i386";
    if substring( option vendor-class-identifier, 0, 9 ) = "PXEClient" {
        filename "/ltsp/i386/pxelinux.0";
    } else {
        filename "/ltsp/i386/nbi.img";
    }
}
```

This `dhcpd.conf` should handle most situations.

By default, LTSP will detect an unused network interface and configure it to be 192.168.0.254. LTSP's recommended single server installation is to use a separate network interface for the thin clients. If, however, you're not using two network interfaces, or you already have an interface in the 192.168.0 range, then you might have to configure the thin client interface differently, which means you may have to adjust the `dhcpd.conf` accordingly.

If the network interface that you're going to connect the thin clients to has, say, a TCP/IP address of 10.0.20.254, you'll want to replace every occurrence of 192.168.0 with 10.0.20 in the `dhcpd.conf` file.

Always remember, you'll need to re-start the dhcp server if you make any changes. You can do this by issuing the command:

```
sudo invoke-rc.d dhcp3-server restart
```

(at the command prompt.)

Chapter 25

Adding static entries to the dhcpd.conf

Sometimes, you may need to have a certain terminal boot with a guaranteed fixed TCP/IP address every time. Say, if you're connecting a printer to the terminal, and need to make sure the print server can find it at a fixed address. To create a fixed address, use a low number in the range of 2-19, or otherwise, if you change the range statement in the `dhcpd.conf`.

To create a static entry, simply add the following after the "option root-path" line:

```
host hostname {
    hardware ethernet MA:CA:DD:RE:SS:00;
    fixed-address 192.168.0.2;
}
```

Substitude the MAC address for the mac address of the thin client you wish to fix the address of. The fixed-address will be the TCP/IP address you want, and "hostname" is the name you wish to give the host. This kind of setup is relatively complex and the admin should have a full understanding of how DHCP works before attempting such a setup. For more information, check the Internet.

Chapter 26

DHCP failover load balancing

Another common method of load balancing is to use DHCP load balancing. There's an excellent writeup on the topic at: https://wiki.edubuntu.org/EdubuntuDHCPload-balancingFailover

26.1

26.1.1 Lockdown with Sabayon (user profile manager) and Pessulus (lockdown editor)

A common requirement in both schools and businesses is having the ability to lock down the desktop and provide certain default configurations.

In LTSP, the applications you'll want to use are Sabayon and Pessulus. You'll want to add them from the package manager.

The Sabayon user profile editor looks like a window that contains a smaller sized picture of your desktop. Within this window, you can create a default layout: add icons to panels and the desktop, lock down the panels so they can't be modified, remove access to the command line, etc.

Once you're done, you can save your profile. You have the option of applying your profile to either individual users, or all users on the system. Please consult the manual included with Sabayon for all the details.

More information is available here:

http://live.gnome.org/PythonSabayon

http://www.gnome.org/~seth/blog/sabayon

http://www.gnome.org/projects/sabayon/

26.2 Replication of desktop profiles

If you customize user's desktop, then custom desktop profiles should be copied to every server. Gnome desktop profiles created with Sabayon are located in `/etc/desktop-profiles`

Chapter 27

Managing the thin client

Previously, there was a program called TCM or thin client manager, which was responsible for checking what was happening on the various thin terminals, messaging between them, locking, or generally offering support from a master terminal. This has now been replaced by the use of Italc, which must be separately installed depending on your distribution.

27.1 Lockdown Editor

By choosing a single user and right clicking on that users name, you will open up the context menu. From here you can choose "Lockdown", which will allow you to set options to restrict a particular user. Clicking this menu item will invoke the "Pessulus" program, which is the Gnome lockdown editor. Ticking and unticking options in Pessulus will enable and disable certain functions for that particular user. There is a padlock next to each option in Pessulus. Ticking this will make the option unchangeable by the user. This is called a mandatory setting. For further help with Pessulus, please refer to the Pessulus documentation.

Chapter 28

Updating your LTSP chroot

At some point in the future, updates will become available for your LTSP server. You must remember that although you may have applied all the updates to the server itself, as in the instructions....HERE it is likely that the LTSP chroot will also need updating. To do this you must open up a terminal and use the following commands.

First make sure the Client environment has the same Package lists as the Server, to achieve that, you will copy the /etc/apt/sources.list (on Debian and Ubuntu) or the /etc/yum.repos.d/fedora.repo file from the Server to the Client environment.

Now issue the command below.

```
sudo chroot /opt/ltsp/<arch>
```

(replace <arch> with the architecture you are working with.)

This will change your root directory to be the LTSP clients root directory. In essence, anything you now do inside here, will be applied to the LTSP clients root. This is a separate small set of files that are used to boot the clients into a usable, and enable them to contact the LTSP server. Once inside this shell, we must type the following command to obtain the latest list of packages from the apt/yum servers.

```
apt-get get update
```

on Debian and Ubuntu

You need to mount `/proc` in the chroot before beginning, as some of the packages you install may need resources in `/proc` to install correctly.

```
mount -t proc proc /proc
```

To be sure no deamons are started do the following:

```
export LTSP_HANDLE_DAEMONS=false
```

Once this has completed you will have to upgrade the software in the chroot by running the following command:

```
apt-get upgrade
```

(on Debian and Ubuntu)

or

```
yum update
```

(on Fedora)

Just in case `/proc` is still mounted when you exit the chroot, unmount it first by doing:

```
umount /proc
```

Once you're done, you must leave the chroot by either typing *exit* or by using the key combination Ctrl+D. This will return you to the root of the server.

If your kernel has been upgraded you must run the LTSP kernel upgrade script, to ensure that your LTSP chroot uses the latest version. This is performed by running the command below:

```
ltsp-update-kernels
```

All of your clients will now use the latest kernel upon their next reboot.

Finally, you must remember to rebuild the NBD boot image from your chroot with the following command:

```
ltsp-update-image
```

(add architecture using -arch= addition)

Be advised that this may take a few minutes, depending on the speed of your server.

Chapter 29

Changing the IP of your LTSP server

At some point in time, it may become necessary to change the IP address of your LTSP server. Normally this does not present an issue, but LTSP servers and clients communicate over and encrypted channel and require all SSL certificates to be updated. Without this update, *no LTSP clients will be able to log in*. This is done by simply opening a terminal and running the following command.

```
sudo ltsp-update-sshkeys
sudo ltsp-update-image
```

Chapter 30

Appendix I

Here you can find some solutions to common questions and problems.

30.1 Using NFS instead of NBD

Using NBD instead of NFS has several advantages:

1. Using a squashfs image we can now merge that together in a unionfs to get writeable access which is a lot faster during bootup.
2. A squashed root filesystem uses less network bandwidth.
3. Many users and administrators have asked us to eliminate NFS, for reasons of site policy. Since the squashed image is now served out by nbd-server, which is an entirely userspace program, and is started as the user nobody, this should help to eliminate concerns over NFS shares.

However, some people still want to use NFS. Fortunately, it's easy to switch back to NFS, if it's so desired:

1. On the server, use the chroot command to maintain the LTSP chroot:
   ```
   sudo chroot /opt/ltsp/<arch>
   ```
2. Now edit /etc/default/ltsp-client-setup and change the value of the root_write_method variable to use bind mounts instead of unionfs, it should look like this afterwards:
   ```
   root_write_method="bind_mounts"
   ```
3. Next, create the file `/etc/initramfs-tools/conf.d/ltsp` and add the following line (set the value of the BOOT variable to nfs):
   ```
   BOOT=nfs
   ```
4. Regenerate the initramfs:
   ```
   update-initramfs -u
   ```
5. Hit CTRL-D to exit the chroot now. Make sure LTSP uses the new initramfs to boot:
   ```
   sudo ltsp-update-kernels
   ```

30.2 Enabling dual monitors

First, I am going to start with a couple assumptions:

- I will assume that you are operating thin clients with an NBD file system in this write-up.
- I will assume that you are running Ubuntu 8.04.1
- I will assume that you are running LTSP 5
- I will assume that you are replacing a running image that has been properly tested, and is working.

Create a new image to ensure your configuration is congruent with my successfully tested configuration.

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

(note the --arch i386 command is required for my system because it's running an amd64 kernel. It may not be required for individuals running a 32-bit kernel)

Download the pertinent VIA unichrome driver for your chipset from this web site: http://linux.via.com.tw/support/-downloadFiles.action Be sure to select the proper OS as well. The installation script is set up specifically for the directory structure of each OS, and will error out if the wrong OS release is installed. Next we need to move the downloaded file to the image directory

```
cp /home/<username/Desktop/chrome9.83-242-sl10.1.tar.gz /opt/ltsp/i386
```

After that, we need to chroot to the same image directory.

```
sudo chroot /opt/ltsp/i386/
```

Unpack the driver in the root directory

```
tar -zxvf chrome9.83-242-sl10.1.tar.gz
```

After unpacking, enter the directory:

```
cd chrome9.83-242-sl10.1/
```

Run the file contained inside to start the driver installation

```
./vinstall ..................done! Original X config file
was saved as /etc/X11/xorg.conf.viabak
```

(The following error: "VIAERROR:The /etc/X11/xorg.conf is missing!" Can be ignored. We will be replacing the xorg.conf anyway, and the drivers are still installed properly.)

Next we need to put a proper xorg.conf in to the proper directory.

```
gedit /etc/X11/xorg.conf
```

Now paste the following in to the empty file:

```
Section "Module"
    Load "extmod"
    Load "dbe"
    Load "dri"
    Load "glx"
```

```
    Load "freetype"
    Load "type1"
EndSection

Section "Files"
    RgbPath "/usr/X11R6/lib/X11/rgb"
    FontPath "/usr/X11R6/lib/X11/fonts/misc/"
    FontPath "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
    FontPath "/usr/X11R6/lib/X11/fonts/75dpi/"
    FontPath "/usr/X11R6/lib/X11/fonts/Type1"
    FontPath "/usr/X11R6/lib/X11/fonts/TTF"
EndSection

Section "ServerFlags"
    Option "Dont Zoom"
    Option "AllowMouseOpenFail" "Yes"
    Option "BlankTime" "20"
    Option "StandbyTime" "0"
    Option "SuspendTime" "0"
    Option "OffTime" "0"
    Option "Xinerama" "on"
EndSection

Section "InputDevice"
    Identifier "Keyboard1"
    Driver "Keyboard"
    Driver "keyboard"
    Option "AutoRepeat" "500 30"
    Option "XkbRules" "xfree86"
    Option "XkbModel" "pc105"
    Option "XkbLayout" "en_US,en_US"
    Option "XkbOptions" "grp:alt_shift_toggle,grp_led:scroll" EndSection
EndSection

Section "InputDevice"
    Identifier "USBMouse"
    Driver "mouse"
    Option "Protocol" "IMPS/2"
    Option "Device" "/dev/input/mice"
    Option "ZAxisMapping" "4 5"
    Option "Buttons" "5"
EndSection

Section "InputDevice"
    Identifier "Mouse1"
    Driver "mouse"
    Option "Protocol" "Auto"
    Option "Device" "/dev/psaux"
    Option "ZAxisMapping" "4 5"
    Option "Buttons" "5"
EndSection

Section "Monitor"
    Identifier "Monitor0"
    HorizSync 31.5-48.5
    VertRefresh 60
```

```
    Option "DPMS"
EndSection

Section "Device"
    Identifier "CN700"
    Driver "via"
    VideoRam 16384
    Screen 0
    Option "NoDDCValue" "1"
    Option "Simultaneous"
    Option "DPMS" "on"
    BusID "PCI:1:0:0"
EndSection

Section "Screen"
    Identifier "Screen0DVI"
    Device "CN700DVI"
    Monitor "Monitor0DVI"
    DefaultDepth 24
    Subsection "Display"
        Depth 8
        Modes "1280x1024"
        ViewPort 0 0
    EndSubsection
    Subsection "Display"
        Depth 16
        Modes "1280x1024"
        ViewPort 0 0
    EndSubsection
    Subsection "Display"
        Depth 24
        Modes "1280x1024"
        ViewPort 0 0
    EndSubsection
EndSection

Section "Monitor"
    Identifier "Monitor0"
    HorizSync 31.5-48.5
    VertRefresh 60
    Option "DPMS"
EndSection

Section "Device"
    Identifier "CN700DVI"
    Driver "via"
    VideoRam 16384
    Screen 1
    Option "NoDDCValue" "1"
    Option "Simultaneous"
    Option "DPMS" "on"
    BusID "PCI:1:0:0"
EndSection

Section "Screen"
    Identifier "Screen0"
```

```
    Device "CN700"
    Monitor "Monitor0"
    DefaultDepth 24
    Subsection "Display"
        Depth 8
        Modes "1280x1024"
        ViewPort 0 0
    EndSubsection
    Subsection "Display"
        Depth 16
        Modes "1280x1024"
        ViewPort 0 0
    EndSubsection
    Subsection "Display"
        Depth 24
        Modes "1280x1024"
        ViewPort 0 0
    EndSubsection
EndSection

Section "ServerLayout"
    Identifier "Simple Layout"
    Screen "Screen0" 0 0
    Screen 1 "Screen0DVI" LeftOf "Screen0"
    InputDevice "Mouse1" "CorePointer"
    InputDevice "Keyboard1" "CoreKeyboard"
    InputDevice "USBMouse" "AlwaysCore"
EndSection
```

IMPORTANT NOTE IN THE ABOVE SECTION PASTED INTO THE XORG.CONF, NOTICE THAT THERE ARE RESOLUTIONS SPECIFIED PER MONITOR. PLEASE ENSURE THAT YOU HAVE THE PROPER RESOLUTIONS FOR YOU YOUR MONITOR ENTERED ON THOSE AREAS. Be sure to save the file as xorg.conf and exit out of your chroot'd image. <CTRL+D or "exit"> next we need to put in an addendum in the lts.conf

```
gedit /var/lib/tfpboot/ltsp/i386/lts.conf
```

Feel free to comment out anything that you need to in the lts.conf. I will include my full lts.conf as an example:

```
[DEFAULT]
    #X_COLOR_DEPTH = "16"
    #X_MODE_0 = "1680x1050"
    #X_VERTREFRESH = "43-61"
    #X_HORZSYNC = "28-85"
    #X_OPTION_01 = "\"ForcePanel\" \"True\""
    #X_OPTION_01 = "\"NoPanel\" \"true\""
    #SCREEN_02 = shell
    #SCREEN_07 = ldm
    X_CONF = /etc/X11/xorg.conf
```

Feel free to copy-paste this in it's entirety if you want, but you will only need the last line. After you add the X_CONF line, save & exit. Now we need to make the changes that we have made take effect in the image

```
# sudo ltsp-update-image --arch i386
```

(again, the --arch i386 will not be required for most, but I am putting it in just in case a user has a x64 installation on their server.) That should do it! Boot up the client and you should be good to go.