The Linux Terminal Server Project: Thin clients and Linux

Jim McQuillan
Project leader - The Linux Terminal Server Project
jam@ltsp.org

Abstract

Linux makes a great platform for deploying diskless workstations that boot from a network server. The LTSP is an open source project to create the administration tools that will make setting up a diskless workstation easier. This document provides a description of how a diskless workstation works, describes how to obtain the ltsp distribution and how to install the LTSP.

1. Introduction

This project was born out of the need to solve a problem for a customer who needed a terminal that could communicate with both an IBM AS/400 and a Unix application server. It needed to run TCP/IP, it needed to be inexpensive, and it needed to be easy to maintain. As a plus, it should allow the user to browse the web, and allow them to read and send email.

We could have used PC's running windows, the software is certainly available, but the cost would have been high. Both in terms of the initial investment, and in supporting and maintaining the PC's over time.

We decided that a diskless workstation running the Linux kernel and X-Windows would fit nicely as a solution to the customer problem.

We didn't really invent anything new here, we searched the web and found the etherboot and netboot packages. Then we found an inexpensive ne2000 network interface card that had an eprom socket. We searched and found an affordable eprom burner, figured out what kind of eprom chips to buy, learned about bootp, xdm, nfs-root and all kinds of other things. Then, we put it all together. Initially we tried it on an old 486 PC we had lying around and it worked pretty well.

So, we installed a server and 11 workstations. The customer loved it. They ordered an additional 22 workstations, and they will probably add many more in the future. (16 June 2000 Update: They now have over 100 workstations)

After several months, and virtually no support problems with the workstations, we have decided to share our solution with the rest of the world.

The package we developed works very well for our uses. Hopefully, it will solve problems for others as well.

If you have any questions or comments about the package, please join the discuss mailing list at **http://www.ltsp.org/mailinglists.html**

2. Security Concerns

The installation of the LTSP software will enable some services that could make your system vulnerable to hack attempts.

Therefore, it is strongly recommended that the LTSP server be on the inside of a firewall, it should not be used as the gateway to the public internet.

Several services are required to be running on an LTSP server. These include:

- dhcpd or bootpd
- inetd
- mountd
- NIS
- portmap
- syslogd
- tftp
- xdm
- xfs

3. Description of a diskless workstation

A **Diskless Workstation** is a computer that doesn't need a hard disk, floppy or CDRom to boot from.

Once the workstation is booted, and the user logs in, any application programs that they invoke will be running on the server, while the output is being displayed on the workstation. This is a fundamental feature of X-Windows. The workstation is only running the Linux kernel, XFree86, Init and possibly a print server daemon for printing to the local printer.

Because there is very little running on the workstation, you can get away with a fairly inexpensive, low-power machine. We found in our initial testing that an i486 with 16mb of ram was actually pretty good at being an X terminal.

For the bootrom code, we used the Etherboot package available from: **http://etherboot.sourceforge.net/**. Included with the Etherboot package is the **mknbi-linux** utility that will prepare a tagged kernel for downloading to a workstation.

There are several HOWTO's that explain how to setup a server for diskless booting of a single workstation, but they don't discuss the problems involved in serving many workstations from a single server. The problem is that when the workstation is running, it needs to write to some files on the server, so each workstation needs to mount it's own unique root filesystem. If you had 50 workstations, you would need 50 directory trees exported. This can be a real pain to try to manage.

We have developed a method of setting up the root filesystem hierarchy that can be shared among all of the workstations. The kernel mounts the filesystem in read-only mode, then mounts a ramdisk as it's /tmp filesystem. The size of the ramdisk is configurable, and it defaults to 1mb. While the kernel and XFree86 are running, they like to update a few files, so we have placed those files on the ramdisk, and created symbolic links to them in the proper place within the hierarchy.

Additionally, we have created a configuration file and a program that runs as part of the workstation bootup sequence. Each workstation can include different hardware. Things such as Network board, Video Card and the type of mouse can be configured either as a default, or individually for each workstation.

Because there aren't any application programs running on the workstation, we don't need a swap device, although it is possible to configure the kernel to swap to an NFS filesystem.

This method of booting a workstation is being used very successfully on a network with over 100 workstations all running from a single server running RedHat 6.0 on a 400mhz Pentium-II. Each workstation is a 166mhz Pentium machine with 32mb of ram. We could have used a smaller processor, but these days it is getting pretty tough to find the low-end cpus.

There is also an option that will allow applications to run directly on the workstation. We refer to this as **Local Apps**.

4. Theory of operation

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

1) When you turn on the workstation, it will go through it's power on self test (POST) and the bootcode in the eprom on the network card will begin executing.
2) The bootcode will scan the system for a network card. Once it detects the card, it will initialize it.
3) The bootcode will then broadcast a bootp request to the local network. The request will include the MAC address of the network card. (DHCP can also be used)
4) The **inetd** process on the server will see the broadcast and invoke the **bootpd** daemon to respond to the request.
5) The **bootpd** process will read it's configuration file, /etc/bootptab, and locate the entry that matches the MAC address that was sent in the request. Once the entry is found, it will be put into a reply packet and sent back to the workstation that requested the information. Several pieces of information will be passed back in the reply, the items that are important at this point are:

   IP address assigned to the workstation ('ip=') NETMASK setting for the local network ('sm=') Bootfile home directory ('hd=') The name of the kernel to download ('bf=')
6) The bootcode will receive the reply from bootp and it will configure the TCP/IP interface in the network card with the parameters that were supplied.
7) The bootcode will then send a TFTP request to the server to begin downloading the kernel from the server.
8) Once the kernel has been completely downloaded to the workstation, the bootcode will do a jump to the starting code in the kernel.
9) The kernel will then start executing, initializing the entire system and all of the peripherals.
10) The kernel will issue another bootp broadcast to the network, looking for all of it's networking parameters. Note, the bootcode does NOT pass the information to the kernel, the kernel MUST request the information for itself.
11) The server will respond with another reply packet, containing the information that the kernel needs to

continue.

12) The root filesystem will be mounted via NFS. This filesystem will be mounted read-only. We do this because we may have many workstations mounting the same filesystem, and we don't want any of the workstations to modify the contents of the root filesystem.

13) At this point, control will be passed from the kernel to the **init** process.

14) Init will read the **/etc/inittab** file and begin setting up the environment.

15) One of the first items in the inittab file is the **rc.local** command that will be run while the workstation is in the **sysinit** state.

16) The rc.local script will create a 1mb ramdisk to contain all of the things that need to be written to or modified in any way.

17) This ramdisk will be mounted as the **/tmp** directory. Any files that need to be written will actually exist in the **/tmp** directory, and there are symbolic links pointing to these files. For example, when the workstation is running, it will try to modify the permissions on the **/dev/tty0** device node. If the device node actually existed in the **/dev** directory, the permissions would not be modifiable, because the root filesystem is read-only. So, we have created symbolic links for all of the files and created the actual files/nodes in the **/tmp** directory (which is writeable).

18) The **/proc** filesystem will be mounted.

19) The **loopback** network interface will be configured.

20) Several directories will be created under the **/tmp** filesystem for holding some of the transient files that are needed while the system is running. Directories such as: **/tmp/compiled, /tmp/var, /tmp/var/run, /tmp/var/log, /tmp/var/lock** and **/tmp/var/lock/subsys** will all be created.

21) The **/etc/XF86Config** file will be generated based on entries in the **/tftpboot/lts/ltsroot/etc/lts.conf** configuration file. This is where information about the type of mouse, and other X parameters are combined to create the config file for X.

22) The **/tmp/start_ws** script will be created. This script will determine which X server to run, and the IP address of the server running **xdm.** This is Based on information found in the **/tftpboot/lts/ltsroot/etc/lts.conf** file.

23) The **/tmp/syslog.conf** file will be created. This file will contain information telling the **syslogd** daemon which host on the network to send the logging information to. The syslog host is specified in the **lts.conf** file.

24) The **syslogd** daemon is started, using the config file that was just created.

25) Control is passed back to **init.** Init will then look at the **initdefault** entry to determine which **runlevel** to enter.

26) If runlevel **3** is specified, then a shell will be started on the console. This is good for doing trouble shooting.

27) If runlevel **5** is specified, then the **/tmp/start_ws** script will be invoked, which will either bringup X-windows, or start a telnet session to the server, depending on the value of the configuration entry **'UI_MODE'.**

28) If GUI mode is chosen, then when X is started, it will send and XDMCP query to the server, which will cause the login dialog box to be displayed.

29) Once the user logs in, they will be running processes on the server. That is, if they bring up an **xterm** session, it will be running on the server, and it will be displaying it's output on the workstation.

Basically, we now have either an X station or a telnet terminal.

5. Local apps versus Remote apps

In an LTSP environment you have a choice of running the applications locally on the workstation, or remotely on the server.

The easiest way to setup LTSP is to run the apps on the server. That is, the client application runs on the server, using the servers' CPU and memory, while it displays it's output on the workstation and uses the workstations' keyboard and mouse.

This is a fundamental capability of X Windows. The workstation works just like a standard X-Windows terminal.

5.1. Issues with support for local apps

Setting up the ability to run apps locally requires much more.

- Greater demand on the workstation. It needs more RAM and a more powerful CPU. 64mb of ram on the workstation is a pretty good starting point.

- NIS - To run the apps on the workstation, you first must log into the workstation. This requires some form of password authentication. NIS can be used as the method of authenticating users over the network.

- Additional directories need to be exported for the workstation to mount via NFS.

- Slower startup of applications because they need to be read via NFS, causing increased network activity. Also, because each copy of the program is running on it's own CPU, you won't get the advantage of the Linux/Unix ability to share code segments between multiple instances of the same program, which would reduce the time it takes for second and succeeding launches of a program.

5.2. Benefits of running apps locally

- Reduces the load on the server. In large networks with memory intensive applications such as Netscape, running the app on the workstation can provide better performance, as long as the workstation is powerful enough to handle it.

- Runaway apps will not affect other users.

- Sound support is much easier to configure when the application that plays the sound is running on the workstation.

6. Setting up the server

We have created RPM's and TGZ's containing all of the pieces you will need to setup a Linux system as the server.

The distributions are known to work with RedHat 6.0, 6.1 or 6.2 systems, but have also been reported to work on other distributions of Linux.

You can download all of the software from the **LTSP download page** at *http://www.ltsp.org/download/index.html* or you can ftp the files from the ftp site at *ftp://ftp.ltsp.org/pub/download*

6.1. Planning the IP-Address scheme

Each machine in your network needs a unique IP address. We have chosen one of the reserved Class-C networks, **192.168.0.0.** Of course, you are free to choose any addresses you want.

For the server, we chose **192.168.0.254,** and for the workstations, we started with **192.168.0.1** and climb up from there, this gives us the possibility of having 253 workstations associated with a single server. If you need more workstations than that, you can either setup another class-c on the server, perhaps **192.168.1.254** and have the additional workstations use the addresses between **192.168.1.1** and **192.168.1.253.** Or, you can go to a Class-B address, and have 65533 workstations all running on the same network. (Sounds pretty cool eh?)

We kept the names of the workstations simple, starting with **'ws001'** and going up. Again, you are free to choose any names you want for the workstation. Just make sure they are setup in **/etc/hosts** or in **DNS.**

6.2. Choose the appropriate kernel

When the workstation boots, it pulls a kernel from the server and loads it into memory. The kernel that it loads needs to be configured specifically for network booting, and it needs to have the proper network card driver included. It cannot load a network card driver module at this point. We have prepared a few kernels that are ready to go, you will just need to pick the correct one for your network card, download it and install it.

You can create your own kernel. When setting kernel parameters, you MUST make sure you specify the following:

- Support for your specific network card
- RAM disk support
- IP kernel level autoconfiguration
- BOOTP support
- /proc filesystem support
- NFS filesystem support
- Root file system on NFS
- Support for Parallel and/or serial ports for printers

Once the kernel is built, it must be converted into a tagged image format, using the **mknbi-linux** command, which is part of the Etherboot package.

```
mknbi-linux
      --output=/tmp/vmlinuz.ne2000
      --append="ramdisk=1024"
      --rootdir=/tftpboot/lts/ltsroot
      /usr/src/linux/arch/i386/boot/bzImage
```

The kernel then needs to be placed into the **/tftpboot/lts** directory.

6.3. Choose the X server

You will need to pick the correct X server that is compatible with your video card. The XF86_SVGA server works with most video chipsets, but you may get better performance by using the server made specifically for your chipset.

You can download one of X servers from ftp.ltsp.org that will automatically install themselves in the proper location.

6.4. Edit the configuration files

The installation of the lts_core package will add entries to several configuration files, but these entries may need to be modified for your specific needs.

6.4.1. /etc/inetd.conf

By default, the lines in the inetd.conf file for the bootp and tftp servers are commented out. The LTSP installation procedure will uncomment the tftp daemon line, but it will NOT uncomment the bootps line.

6.4.2. /etc/X11/xdm/Xservers

If you already have the server setup in runlevel 5, then you are already getting the graphical login screen when the server is booted. In this case, the installation script will leave the **Xservers** file untouched. If you are running the server in runlevel 3, then the installation script will modify the **Xservers** file by commenting out the entry for the server's console. This is so that when the installation script changes the default runlevel to 5, the console won't switch into graphical mode unexpectedly. If you do want the server to run in graphical mode, then you can either start X-Windows by running the **startx** command or uncomment the line in the Xservers file.

6.4.3. /etc/X11/xdm/Xaccess

This file controls whether a remote workstation can communicate with the xdm program on the server. The installation script will uncomment the wildcard entry to allow the remote workstations to get a login screen.

6.4.4. /etc/X11/xdm/xdm-config

By default, in Redhat 6.2, there is an entry that must be removed (or commented out).

The installation script will take care of commenting the line for you.

The line that is commented out looks like:

DisplayManager.requestPort: 0

6.4.5. /etc/inittab

The server needs to have xdm running. This is usually started from the **/etc/inittab** file. There are multiple xdm type servers available on a Redhat system. (xdm, gdm and kdm)

We recommend using xdm.

The installation script will setup xdm to run for you, and it will change the default runlevel to 5.

6.4.6. /etc/bootptab

If you are using bootp, then you will need to configure this file to provide the IP information to the workstation.

6.4.7. /etc/dhcpd.conf

If you are using **DHCP** instead of bootp, then you will need to configure the **/etc/dhcpd.conf** file so that DHCP can hand out IP information to the workstation.

An example dhcpd.conf file is installed as part of the LTSP installation. The file is called **/etc/dhcpd.conf.example** It can be copied or renamed to **/etc/dhcpd.conf** and dhcpd can be started.

6.4.8. /etc/hosts

Once you have added the workstation to the bootptab file or dhcpd.conf file, make sure you add the IP-Address and name of the workstation to either the **/etc/hosts** file, or set it up in your DNS tables either on the server, or somewhere on your network. The NFS server MUST be able to resolve the IP address to a name.

6.4.9. /etc/hosts.allow

The LTSP installation script will add entries to the **/etc/hosts.allow** file that will allow bootpd, portmapper and tftp to function properly.

The entries added look like:

```
bootpd:   0.0.0.0
in.tftpd: 192.168.0.
```

portmap: 192.168.0.

The above entries are assuming that you are using the private 192.168.0.0 class-C. If you are using a different class-C then substitute accordingly.

6.4.10. /etc/exports

The default entry that is put into this file should be Ok. Unless you are using a Class-C other than **192.168.0.0.** If you are using a different class-c, then you need to modify this file to show it.

6.4.11. /etc/rc.d/init.d/syslog

The LTSP installation script will modify the **/etc/rc.d/init.d/syslog** startup script for you, to enable remote workstations to send their syslog messages to the server.

6.4.12. /tftpboot/lts/ltsroot/etc/lts.conf

This is the config file for the workstations. Most of the configurable parameters for the workstations can be specified here.

The config file is comprised of sections, each section represents a workstation, and there is a **Default** section.

Section headers contain the name of the workstation or the word 'Default', surrounded by square brackets ('[' and ']'). If all of the workstations are identical, then you could get away with just having a Default section.

Example of a **/tftpboot/lts/ltsroot/etc/lts.conf** file:

```
[Default]
    XSERVER             = XF86_SVGA
    SERVER              = 192.168.0.254
    X_MOUSE_PROTOCOL    =
    X_MOUSE_DEVICE      =
    X_MOUSE_RESOLUTION  = 400
    X_MOUSE_BUTTONS     = 3
    USE_XFS             = N
    UI_MODE             = GUI

[ws001]
    XSERVER             = XF86_SVGA
    X_MOUSE_PROTOCOL    =
    X_MOUSE_DEVICE      =
    X_MOUSE_RESOLUTION  = 50
    X_MOUSE_BUTTONS     = 3
    X_MOUSE_BAUD        = 1200

[ws002]
    XSERVER             = XF86_Mach64

[ws003]
    XSERVER             = XF86_SVGA
    X_COLOR_DEPTH       = 24
    USE_XFS             = N

[ws004]
    UI_MODE             = CHAR
```

See the list of the available parameters later in this document.

6.5. Configuring the server for Local apps

Several additional changes must be made on the server to enable it to provide the ability to run local applications on the workstations.

6.5.1. Network Information System - NIS

NIS is a very complex system to administer, and this document does not attempt to explain it in great depth. For a more complete reference, there is an O'Reilly book available called **Managing NFS and NIS**. Refer to the list of references at the end of this document for more information about the book.

There is also a very good HOWTO document on the LDP called **The Linux NIS(YP)/NYS/NIS+ HOWTO**. Refer to the list of references at the end of this document for the url.

There are a few simple settings that can be done to get it running with LTSP:

- Turn off shadow passwords. The NIS HOWTO explains why.

- Edit the yp Makefile.

  Change into the **/var/yp** directory, edit the Makefile.

  Look for the **MERGE_PASSWD** setting, on or near line 34. Change it to **false.**

  Look for the **MERGE_GROUP** setting, on or near line 38. Change it to **false.**

  Look for the line that starts with **all:**, on or near line 96. Comment out the **netgrp** entry by putting a '#' in front of the word.

Start ypserv by running:

/etc/rc.d/init.d/ypserv start

You should run **ntsysv** to set it to start everytime the server is booted.

Set the NIS domain so you can run ypinit.

domainname ltsp

Initialize NIS by running ypinit:

/usr/lib/yp/ypinit -m

At this point, NIS should be ready to go.

7. Setting up the workstation

The workstation needs a network card with a bootrom.

If you have access to an eprom programmer, you can download the etherboot image from the LTSP website. It is based on the Etherboot project, and the source code is available at http://etherboot.sourceforge.net if you want to compile the image yourself.

Another alternative is to download the bootrom image from the LTSP website, along with the floppyload.bin file. Then, copy the two files to the boot sector of a floppy, then boot from the floppy. It will probe for the network card, then begin the boot process just as if the code came from an eprom on the network card. This is very useful for testing purposes.

The command for copying the files to the floppy is:

cat floppyload.bin ne.rom >/dev/fd0

This will place the files into the first sector of the floppy.

8. Testing

Reboot the server after making all of the above changes. Then, just turn on the workstation and you should see it query the network for it's IP information, then you should see a tftp transfer of the kernel and the kernel will begin booting. You should then see X-Windows come up, and finally, a login window should appear.

If it fails somewhere along the way go back through the instructions above and double check the setup. If you are still having trouble, join the LTSP discussion mailing list, available at http://www.ltsp.org/mailinglists.html

9. Tuning the server

Once you get the server and workstations setup, and you begin to increase the size of the network, you may need to perform some tuning of the server.

The following items may need to be tuned:

9.1. Maximum number of open files

By default, the maximum number of files that can be opened is set to 4096. You can check what the max is on your system by running:

cat /proc/sys/fs/file-max

It will report what the limit is.

You can change this limit by echoing a new value into that file, like this:

echo 8192 >/proc/sys/fs/file-max

This will change the limit immediately. A reboot of the server is NOT necessary.

You should put this command in the **/etc/rc.d/rc.local** file so that it runs each time the server is rebooted.

9.2. Maximum number of inodes

By default, the maximum number of open inodes is 16384. You can check what the max is on your system by running:

cat /proc/sys/fs/inode-max

It will report what the limit is.

You can change this limit by echoing a new value into that file, like this:

echo 32768 >/proc/sys/fs/inode-max

This will change the limit immediately. A reboot of the server is NOT necessary.

You should put this command in the **/etc/rc.d/rc.local** file so that it runs each time the server is rebooted.

9.3. Maximum number of processes

On Redhat 6.0, and in kernels downloaded from kernel.org, the limit on the total number of processes is 512, and the limit on the number of tasks per user is half of that.

Since Redhat 6.1, using the standard redhat supplied kernel, the total number of process is 2560, and the max per user is 2048.

If you need to increase the limits, you will need to modify the **/usr/src/linux/include/tasks.h** file. The parameters to change are **NR_TASKS** and **MAX_TASKS_PER_USER.**

In all cases, the maximum value for these parameters is **4092** (**4090** if APM is enabled).

If you change these parameters, you will need to build a new kernel.

10. LTSP on other versions of Linux and Unix

The LTSP is currently written to work on Redhat Linux 6.0, 6.1 and 6.2, although there is nothing specific about it that requires Redhat.

Various people have reported success with making it work on other distributions, such as Debian and SuSE.

We have received a report from David Anders (dave123@abcsinc.com) who was able to install LTSP on SCO OpenServer 5.0.5!

At some point, we would like to include the instructions for the other distributions, so if you have had some experience making LTSP work on platforms other than Redhat, please send us some info.

11. Available lts.conf parameters

11.1. General parameters

Comments
: Comments start with the hash '#' sign and continue through the end of the line.

SERVER
: 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, **192.168.0.254** will be used.

SYSLOG_HOST
: 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

UI_MODE
: This will determine whether X-Windows will run, or a telnet session will run. The available choices are:

  GUI Runs X-windows on the workstation

  CHAR Runs a telnet session to the server

  SHELL Runs a shell on the workstation. Useful for debugging

  Currently, this is only used when **NOT** running local apps. The default value is **GUI.**

TELNET_HOST
: If the workstation 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.

DNS_SERVER
: Used to build the resolv.conf file.

SEARCH_DOMAIN
: Used to build the resolv.conf file.

MODULE_01 thru MODULE_10
: Upto 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:

RAMDISK_SIZE
: When the workstation boots, it creates a ramdisk and mounts it on the /tmp directory. You can control the size of the filesystem with this parameter. Specify it in units of kbytes (1024 bytes). To create a ramdisk of 2 megabytes, specify **RAMDISK_SIZE = 2048**

  If you change the size of the ramdisk here, you will also need to change the size of the ramdisk within the kernel. This can be compiled in, or if you are using Etherboot or Netboot, you tell the kernel the ramdisk size when you tag the kernel with mknbi-linux.

The default value for this is 1024 (1 mb)

11.2. X-Windows parameters

XDM_SERVER
: If you want to point XDM to a machine other than the default server, then you can specify the server here. If this parameter is NOT specified, then it will use the 'SERVER' parameter described above.

XSERVER
: This defines which X server the workstation will run. Possible values include: **XF86_SVGA** and **XF86_Mach64.** Any other XFree86 X server should work, as long as it has been installed in the **/tftpboot/lts/ltsroot/ltsbin** directory. The default value for this is **XF86_SVGA.**

X_MOUSE_PROTOCOL
: Any value that will work for the XFree86 Pointer Protocol keyword can be put here. Typical values include "Microsoft" and "PS/2". The default value for this is

X_MOUSE_DEVICE
: 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.** If it is a PS/2 keyboard mouse, this value would be **/dev/psaux.** The default value for this is **/dev/psaux.**

X_MOUSE_RESOLUTION
: This is the 'Resolution' value in the **XF86Config** file. A typical value for a serial mouse is **50** and a typical value for a PS/2 mouse would be **400.** The default value for this is **400.**

X_BUTTONS
: This tells the system how many buttons the mouse has. Usually set to **2** or **3.** The default value for this is **3**

X_MOUSE_BAUD
: For serial mice, this defines the baud rate. The default value for this is **1200.**

X_COLOR_DEPTH
: This is the number of bits to use for the color depth. Possible values are **8, 15, 16, 24** and **32.** 8 bits will give 256 colors, 16 will give 65536 colors, 24 will give 16 million colors and 32 bits will give 4.2 billion colors! Not all X servers support all of these values. The default value for this is **16.**

USE_XFS
: You have a choice of running the X Font Server (XFS) or reading the fonts through the NFS filesystem. The font server should provide a simple way of keeping all of the fonts in one place, but there has been some problems when the number of workstations grows past about 40. The 2 values for this option are **Y** and N. The default value is **N.** If you do want to use a font server, then you can use the **XFS_SERVER** entry to specify which host will act as the font server.

XFS_SERVER
: 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.

X_HORZSYNC
: This sets the XFree86 <bf/HorizSync/ configuration parameter. It defaults to

X_VERTREFRESH
: This sets the XFree86 <bf/VertRefresh/ configuration parameter. It defaults to

X_MODE_1024x768
: This sets the XFree86 <bf/Modeline/ entry for the **1024 x 768** resolution. There is already a default for the 1024x768, 800x600 and 640x480. If you set Any of the **X_MODE_** entries then the 3 default entries will NOT be used and only the entries that you set explicitly will be used.

X_MODE_800x600
: This sets the XFree86 **Modeline** entry for the **800 x 600** resolution. See the note on the 1024x768 entry for an explanation about the defaults.

X_MODE_640x480
: This sets the XFree86 **Modeline** entry for the **640 x 480** resolution. See the note on the 1024x768 entry for an explanation about the defaults.

XF86CONFIG_FILE
: If you want to create your own complete XF86Config file you can do so and place it in the **/tftpboot/lts/ltsroot/etc** directory. Then, whatever you decide to call it needs to be entered as a value for this configuration variable.

11.3. Touch screen parameters

USE_TOUCH
: If you are connecting a touch screen to the workstation, you can enable it by setting this entry to **Y.** If enabled, additional configuration entries will configure specific aspects of the touch screen. The default value is **N.**

X_TOUCH_DEVICE
: A touch screen works like a mouse and usually is interfaced with the workstation through a serial port. You can specify which serial port with this entry. For example, you could set it equal to **/dev/ttyS0.** There is no default value for this entry.

X_TOUCH_MINX
: Callibration entry for an EloTouch touch screen. Defaults to **433.**

X_TOUCH_MAXX
: Callibration entry for an EloTouch touch screen. Defaults to **3588.**

X_TOUCH_MINY
: Callibration entry for an EloTouch touch screen. Defaults to **569.**

X_TOUCH_MAXY
: Callibration entry for an EloTouch touch screen. Defaults to **3526.**

X_TOUCH_UNDELAY
: Callibration entry for an EloTouch touch screen. Defaults to **10.**

X_TOUCH_RPTDELAY
: Callibration entry for an EloTouch touch screen. Defaults to **10.**

11.4. Local apps parameters

LOCAL_APPS
: If you want the ability to run applications locally on a workstation, set this variable to **Y.** Several additional steps must be taken on the server to enable local apps. See the 'Local Apps' section in the LTSP manual for more information. The default value is **N.**

LOCAL_WM
: If you do setup LOCAL_APPS, then you have a choice of where you want the window manager to run. It can either run on the workstation or on the server. If you set **LOCAL_WM** to **Y** then the window manager will run on the workstation. If you set it to **N** then it will run on the server. Local apps are much easier to setup if you run the window manager locally on the workstation. Then, by default, any programs that are launched are also run locally. The default value is **Y.**

NIS_DOMAIN
: If you do setup LOCAL_APPS, then you need to have an NIS server on the network. The NIS_DOMAIN entry is where you specify the NIS domain name. It needs to match a domain name that has been defined on the NIS server. This is NOT the same thing as an internet DOMAIN. The default value is **ltsp.**

NIS_SERVER
: Set this to the IP address of your NIS server if you don't want it to send a broadcast looking for an NIS server.

11.5. Keyboard parameters

All of the keyboard support files are now copied into the ltsroot hierarchy so configuring internation keyboard support is now a matter of configuring XFree86. Several configuration parameters are available to make this possible.

XkbTypes
: The default value for this is the word **'default'**

XkbCompat
: The default value for this is the word **'default'.**

XkbSymbols
: The default value for this is **'us(pc101)'.**

XkbModel
: The default value for this is **'pc101'.**

XkbLayout
: The default value for this is **'us'.**

The values for the above parameters are from the XFree86 documentation. Whatever is valid for XFree86 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 the ltsp core group would be greatly appreciated.

11.6. Printer configuration parameters

Upto three printers can be connected to a diskless workstation. A combination of serial and parallel printers can be configured via the following entries in the **lts.conf** file:

PRINTER_0_DEVICE
: The device name of the first printer. Names such

as **/dev/lp0, /dev/ttyS0** or **/dev/ttyS1** are allowed.

PRINTER_0_TYPE
: The type of the printer. Valid choices are **'P'** or for Parallel, and **'S'** for Serial.

PRINTER_0_PORT
: The TCP/IP Port number to use. By default, it will use **'9100'**

PRINTER_0_SPEED
: If the printer is serial, this is the setting that will select the baud rate. By default, **'9600'**will

PRINTER_0_FLOWCTRL
: For serial printers, the flow control can be specified. Either **'S'** for Software (XON/XOFF) flow control, or **'H'** for Hardware (CTS/RTS) flow control. If neither is specified, **'S'**will

PRINTER_0_PARITY
: For serial printers, the Parity can be specified. The choices are: **'E'**-Even, **'O'**-Odd or **'N'**-None. If not specified, **'N'** will be used.

PRINTER_0_DATABITS
: For serial printers, the number of data bits can be specified. The choices are: **'5', '6', '7'** and **'8'.** If not specified, **'8'**will

PRINTER_1_DEVICE
: Second printer device name

PRINTER_1_TYPE
: Second printer type

PRINTER_1_PORT
: Second printer tcp/ip port

PRINTER_1_SPEED
: Second printer baud rate (serial)

PRINTER_1_FLOWCTRL
: Second printer flow control (serial)

PRINTER_1_PARITY
: Second printer parity (serial)

PRINTER_1_DATABITS
: Second printer data bits (serial)

PRINTER_2_DEVICE
: Third printer device name

PRINTER_2_TYPE
: Third printer type

PRINTER_2_PORT
: Third printer tcp/ip port

PRINTER_2_SPEED
: Third printer baud rate (serial)

PRINTER_2_FLOWCTRL
: Third printer flow control (serial)

PRINTER_2_PARITY
: Third printer parity (serial)

PRINTER_2_DATABITS
: Third printer data bits (serial)

12. Additional references

12.1. Online references

- **Linux Terminal Server Project (LTSP)**
  http://www.ltsp.org

- **Diskless-Nodes HOW-TO document for Linux**
  http://www.linuxdoc.org/HOWTO/Diskless-HOWTO.html

- **Etherboot Home Page**
  http://etherboot.sourceforge.net

- **XFree86-Video-Timings-HOWTO**
  http://www.linuxdoc.org/HOWTO/XFree86-Video-Timings-HOWTO.html

- **The Linux NIS(YP)/NYS/NIS+ HOWTO**
  http://www.linuxdoc.org/HOWTO/NIS-HOWTO.html

- **DisklessWorkstations.com home page**
  http://www.DisklessWorkstations.com

12.2. Print Publications

- **Managing NFS and NIS**
  Hal Stern
  O'Reilly & Associates, Inc.
  1991
  ISBN 0-937175-75-7

- **TCP/IP Illustrated, Volume 1**
  W. Richard Stevens
  Addison-Wesley
  1994
  ISBN 0-201-63346-9

- **X Window System Administrator's Guide**
  Linda Mui and Eric Pearce
  O'Reilly & Associates, Inc.
  1993
  ISBN 0-937175-83-8
  (Volume 8 of the The Definitive Guides to the X Window System)