

Ubuntu Linux & LTSP Introductory Guide



Current version of this document can be found at: <http://www.logicalnetworking.net/other/UbuntuLinuxLTSPIntroductoryGuide.pdf>

Originally written by: Logical Networking Solutions



Table of Contents

| | |
|--|-----------|
| 1. License | 3 |
| 2. Introduction (a.k.a Useless Dribble) | 4 |
| 3. Linux System Administration | |
| a) User/Group Management | 5 |
| b) Ending Misbehaving User Processes | 7 |
| c) Using the Filesystem | 9 |
| 4. How LTSP Works | 11 |
| 5. Notes on Using the Shell | 12 |
| 6. Foreign Software and Linux | 13 |

1. License

The material in this document is released under the [Creative Commons Attribution-ShareAlike 3.0 License](#).



You are therefore free to share and adapt the material, provided that you do so under the same or similar license, and that you give credit to the original authors.

The full text of the license can be found on the [Creative Commons website](#).

When attributing, it is sufficient to refer to the authors of the document as a whole rather than individually, so "Logical Networking Solutions", although you should check the relevant page in case any specific attributions are required.

2. Introduction (a.k.a. Useless Dribble)

Ubuntu is one of the most popular distributions¹ of Linux today. It is based on Debian Linux, which is known mostly for its focus on security and stability. It is considered a very close sibling to Debian, keeping its strengths in mind, while adding things like improved hardware support, easy to navigate graphical interfaces, and up-to-date software. The two projects exchange code updates between each other on a regular basis, and the relationship has proven to be very successful.

LTSP (the *Linux Terminal Server Project*) is a suite of software that adds thin-client functionality to Linux. This allows many computers to act as terminals to the central LTSP server, which handles virtually *all* aspects of the clients – think of it like a bunch of monitors, keyboards and mice running from the server itself, that all utilize the power of the server for operations on the thin-clients. System administration is also completely centralized - users are added *once*. Software is added *once*. Updates are applied *once*. Printers are configured *once*. You get the picture.

Ubuntu Linux is the primary distribution of LTSP. This means the core software developers that work on LTSP primarily work toward Ubuntu compatibility and functionality *first*, and other developers follow suit with integration into other distributions.

All of this, along with the proven track-record of Linux stability, security, flexibility and now ease of use and amazing hardware compatibility, make Ubuntu and LTSP the perfect option for many thin-client situations, including computer labs and classrooms in educational and related environments.



¹ A branding - a collection of software that is considered unique

3. Linux System Administration

3a: User/Group Management

Managing users and groups is fairly easy once you get the hang of it. There are many different ways to manage users Linux (as you will find is the case with many tasks), but we will cover a couple of the most common.

Using the KUser GUI

KUser is a graphical interface to add, modify and delete users. It resembles Windows-based user managers, and is intuitive enough for anyone to get used to fairly quickly. If KUser is not installed, you must install it (**Applications** → **Add/Remove**).

- To add a new user using KUser, launch it (**Applications** → **System Tools** → **KUser**) and click **User** → **Add User**.
- Type in the desired username.
- When **User Properties** appears, you may type in the user's full name.
- Click on **Set Password** to give the user a password. Setting a password will automatically enable the user account. If you will be assigning the user to any groups (such as “students” or “teachers”) assign them by clicking on the **Groups** tab and checking each group they will be a member of.
- Click **OK** to finish the creation of the user. You may now log in with this new user!

Using the shell

| COMMANDS USED | DESCRIPTION OF COMMAND |
|-------------------|-----------------------------------|
| sudo | execute a command as another user |
| adduser, addgroup | add a user or group to the system |

To add a user using the shell, launch a terminal (**Applications** → **Accessories** → **Terminal**) and type:

sudo adduser <username>

You should see something similar to the following, and then a prompt to enter the password for the new user:

```
Adding user `<username>' ...
Adding new group `<groupname>' (1007) ...
Adding new user `<username>' (1004) with group `<groupname>' ...
Creating home directory `/home/<username>' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
```

Type in a password for the new user, and retype it when prompted. You should then see:

```
passwd: password updated successfully
Changing the user information for <username>
Enter the new value, or press ENTER for the default
    Full Name []:
```

Type in the user's full name (if desired). You may hit <Enter> through any prompts you do not wish to fill in. Hit **Y** to finish the creation of the new user when prompted if the information you entered was correct.

To add the newly created user to a group simply type:

```
sudo addgroup <username> <groupname>
```

```
Adding user `<username>' to group `<groupname>' ...
Adding user <username> to group <groupname>
Done.
```

You can also simply add a new group to the system by typing:

```
sudo addgroup <groupname>
```

```
Adding group `<groupname>' (GID 1007) ...
Done.
```

3b: Ending Misbehaving User Processes

Sooner or later, you will have to deal with misbehaving processes. Misbehaving processes are programs, utilities or other software that stops, crashes or otherwise is not acting as it normally should (one common example is if you launch the Firefox web browser and it claims that it is already running, though there is no Firefox window present for that user). A Linux system administrator's job is to find and “kill” (official Unix terminology) these processes. The concept of killing misbehaving processes is the same across operating systems – the difference is that in Linux environments, misbehaving processes rarely bring the entire OS down with it.

We will explore two different ways to kill processes – one with a GUI called “System Monitor” and one using the shell.

Using the “System Monitor” GUI

System Monitor is a graphical application that allows you to view current running processes, system performance, and other aspects of the system. You can launch System Monitor by navigating **System** → **Administration** → **System Monitor**. The first thing you will want to do is enable the viewing of all user processes (Navigate to **Edit** → **Preferences** and on the **Processes** tab, **under Information Fields** click the check box next to **User**). Click on **Close** and sort the processes under the **Processes** tab by User. Find the misbehaving process of the user, *right-click* on it and select **End Process**. This will attempt to end the process with normal means. If this does not end the process, *right-click* on the process again and select **Kill Process**.

Using the shell

| COMMANDS USED | DESCRIPTION OF COMMAND |
|---------------|---|
| ps | report a snapshot of the current processes |
| sudo | execute a command as another user (root by default) |
| killall | Kill processes by name |
| kill | Send a signal to a process |

To kill a process(es) of a certain user using the shell, you must first identify the name of the process. In this example, we will assume the user's Firefox browser is misbehaving. Launch a terminal (**Applications** → **Accessories** → **Terminal**) and type:

ps w -u <username>

Your output will look similar (but definitely not the same) as this:

```
PID TTY          TIME CMD
5236 ?            00:00:05 gconfd-2
5336 ?            00:00:00 sshd
5337 pts/0        00:00:00 bash
5342 pts/0        00:00:00 sh
5345 ?            00:00:00 bash
5348 ?            00:00:00 ck-launch-sessi
```

```
5372 ?          00:00:00 ssh-agent
5376 ?          00:00:00 x-session-manag
5792 ?          01:03:19 firefox
```

As you can see in the process list, there is a process called *firefox* which is running. To kill this process, we use the 'killall' command.

sudo killall -u <username> firefox

Will kill all processes called 'firefox' running as the user <username>. We prepend this command with **sudo** so we run the command as the root user, which has the privileges necessary to kill other users' processes (other users cannot do this for security reasons). Alternately, we can use the 'kill' command to kill processes by process number:

sudo kill 5792

will kill process number 5792 (derived from the process list above). Process numbers are unique across the system, so you don't have to specify a user when running 'kill'.

If you use one of the examples above to end a process and it remains (verified by another process listing), you might have to use a switch to unmercifully kill the process (normal usage of 'kill' and 'killall' try to end the process politely – this is the recommended route unless the process is persistent and will not end by normal means). Simply append a switch to the 'kill' or 'killall' command to end it without remorse:

sudo killall -9 -u <username> firefox

or

sudo kill -9 5792

See the manual page for these commands to learn more about the switches used in these examples.

3c: Using the Filesystem

The Linux filesystem hierarchy is much like Apple's OS X[®] filesystem hierarchy, as both have roots with Unix filesystem concepts and practices.

For system administrators that are used to Microsoft-based filesystems, the first difference you will notice is that there is a single *root* of the filesystem (expressed as 'forward-slash', or /). The *root* is the top of the hierarchy, in which all internal and external devices, local and remote filesystems, directories and files are kept.

NOTE: The root of the filesystem should not be confused with the root user account, which is the system administrator's account and has complete control over the system).

In Microsoft-land, hard disks, CD/DVD-ROM drives, USB and other external media, and even mapped network shares are commonly expressed in drive letters (A: B: C: etc). In Linux, OS X and Unix, all of these medias are *mounted* from within the root (again, /) of the filesystem. For example, if you insert a CD-ROM disc into a Linux system's CD-ROM drive, it will most likely be *mounted* under a directory such as **/media/cdrom**. Commonly in GUIs like Gnome, a CD-ROM icon will also appear on the user's desktop for easy access – but don't be confused, as this is basically just a pointer to the **/media/cdrom** directory. All of the files contained in the CD-ROM will appear under this directory. This is the case for most all media in Linux. This approach to managing your files and filesystems will hopefully prove to become much more flexible and stable than others that have obvious limitations and fallacies (such as shifting drive letters).

NOTE: Keep in mind that the forward-slash separator (/) is opposite the Microsoft counterpart, which is normally a backslash (\).

Some of the most common data locations that you'll want to know about are:

The /home directory:

All user data (besides the *root* user) are normally kept in the */home* directory. If you have user *joe*, his default home directory (unless you specified otherwise) will be */home/joe*. User *joe* will have full permissions to create files and directories under */home/joe* but not necessarily */home/jane*. This is typically synonymous to the "Documents and Settings" concept in Windows 2000+. The *root* user is normally the only exception - */root* is this user's home directory.

The /media (and /mnt) directories:

The */media* directory is where you will find device filesystems such as CD/DVD-ROMs and USB drives after you insert them (for example, under */media/cdrom*). Devices that appear under */media* are normally 'auto-mounted' by the system. */mnt* is normally used as a temporary 'mount point' (such as manually mounting a digital camera under */mnt* that isn't automatically mounted by the system).

The /var directory:

Data kept in */var* is normally "variable" data such as system logs, printer/e-mail spool files and other transient and temporary files. */var/log* is a commonly visited directory as it is where all system logs are kept - when you are troubleshooting a problem, this is a useful place to start. Another example is */var/spool/cups*, where all printer spool files go when a user prints a document to a printer

through the CUPS printing subsystem.

/bin, /sbin, /usr, /lib and /opt

All of these directories contain executable programs and their counterparts. It is not normally needed to peruse through these directories unless you are looking for something specific.

/dev, /proc, /tmp and /lost+found:

These are special directories. */dev* contains *device files* that reference hardware such as modems, hard disks and mice. */proc* contains an actual hierarchy of the system's memory. */tmp* is the default directory that all programs' temporary files are kept. */lost+found* is a directory that contains files that are recovered by the system when a crash occurs (such as files that were in use if the system crashed unexpectedly).

For a much more technical and encompassing description of the Linux filesystem hierarchy, please visit the following page: <http://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/>

4. How LTSP Works

LTSP stands for the “Linux Terminal Server Project”. It is a collection of software that turns a normal Linux installation into a terminal server. This allows low-powered, low-cost thin-clients (or legacy hardware already in possession) 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 (such as Microsoft Windows® Terminal Services) 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 PXE² 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:

1. Thin-clients boot via a protocol called PXE (Pre-eXecution Environment)
2. PXE requests an IP address from a local DHCP server
3. The DHCP server passes additional parameters to the thin-client and downloads a Linux filesystem image via TFTP³ into a RAM disk on the client itself.
4. The thin-client then boots the downloaded Linux image, detects hardware, and connects to the LTSP server's X⁴ session (normally handled by LDM⁵).

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.

2 Pre-Execution Environment, or, more commonly, "Network Boot"

3 Trivial File Transfer Protocol

4 X Window System, the default Linux graphical windowing system

5 LTSP Display Manager, the default login manager for LTSP

5. Notes on Using the Shell

- I. Many Linux system administrators prefer using the shell (also referred to as the terminal, command-line or command prompt) for common administration tasks, as it takes much less time compared to GUI operations once you are familiar and comfortable with the environment. It is believed by many seasoned Linux system administrators that the shell is the most powerful and versatile way to interact with the operating system itself because of the close attention paid to flexibility during most shell tools' development.
- II. Linux system administrators generally also have at least **some** knowledge of how to create and modify “shell scripts”, which are files that contain multiple shell commands, variables, and other elements that create a fruitful environment to accomplish the most basic, to most complex tasks in Linux. Shell scripts are similar to Microsoft-based “batch files”, but are generally much more powerful, given the amount of development and attention to principal usage that has gone into shell environments in Linux operating systems compared to Microsoft-based command-line environments.
- III. There are many different “shells” available to Linux. The default shell is called “Bash” (Bourne Again SHell). Most system administrators generally stick to Bash on Linux, but other shells provide alternate functionality that some might find useful.
- IV. There are many command-line utilities to do things like what we have discussed above. If you are curious about how to use any commands, the **man** command gives you information about any given command on the system. For example, if you type **man adduser**, a manual page for the **adduser** command will appear in your terminal and you can read up on exactly how it works. You generally hit **q** to leave the manual page you are reading.
- V. **The Linux shell is very powerful.** Make sure you understand the commands and command switches you are using (especially when using escalated privileges and/or deleting/moving files/directories) as mistakes are very unforgiving. The Linux shell will rarely ask you if you're sure you want to do something – it assumes you know what you're doing, so make sure you do!
- VI. There are countless resources on the Internet regarding the Linux shell. It is generally advised by Linux administrators that if you have questions, you RTFM (Read The Fine Manual) before asking someone. Countless hours has been spent creating detailed, easy to read documentation for those interested in learning how the Linux shell, as well as the rest of the operating system, works. Take a look at <http://www.tldp.org> for some of the best documentation regarding many common Linux tasks. There are also countless other sites – do some Googling and find resources that you enjoy the most.



6. Foreign Software and Linux

Q. "Can I use Linux based application X with Ubuntu/LTSP?"

A. *Probably!* Generally, any software made for Linux will work with Ubuntu in a thin-client environment. There *are* exceptions, such as extremely graphics-intensive software. This is the case in any thin-client environment, as every pixel is transmitted over the network from the server to the thin-client. With 3D applications and games, having a full computer lab with everyone on a particularly graphics-intensive application might slow things down. There is no reason not to try any application, though, unless it is already known to cause problems in a thin-client environment. The Ubuntu and LTSP communities would be glad to hear success/failure stories for any particular application, so if you decide to try something, let them know if it worked! They might even be able to give you some pointers.

Q. "Can I use Microsoft Windows® or DOS based application X with Ubuntu/LTSP?"

A. *Maybe.* Getting software designed for a different operating system to work with Linux is possible, but it really depends on the particular application. There are Windows API compatibility-layer programs such as "Wine" and "Crossover Office" (which is based on Wine) that allow you to install Windows-based programs in Linux, but because of the nature of installing software that is designed for a completely different OS in Linux, it is usually hit-and-miss. You can browse <http://www.winehq.org> to see if your particular application has been tested, and if so, how well it works.

Thin-client environments complicate things since, again, everything flows over the network. If the application in question doesn't work well under Windows Terminal Services, you can probably assume that it won't work under LTSP. Most times this is due to how the application is originally programmed. Again, however, there is no problem with trying, and reporting your findings back to the community.

Apart from using Wine or Crossover Office, there is the option of installing Windows Terminal Server inside a Virtual Machine (or VM) on the Linux server. This allows you to have a native Windows environment for legacy software compatibility. You can connect to the virtualized Windows session via a Linux-based Windows RDP client such as 'tsclient' and have a Windows-native session from your Linux thin-client. When legacy Windows application support in an LTSP environment is required, many opt for this choice, unless their program works with Wine/CXOffice. Of course, in doing this, you must maintain the Windows virtual machine just as you would an actual server - this means Windows and application software updates, anti-virus/anti-spyware maintenance, separate user and file management facilities, etc.

Q. Can I use Apple OSX-based application in Linux?

A. Some online research shows a PowerPC emulator for Windows/Linux called PearPC, which might provide facilities for OSX applications in Linux or Windows. You can visit their page at: <http://wiki.pearpc.net>. As with any emulator, however, compatibility and stability are not guaranteed.