Linux
Server
Configuration

Table of Contents

Chapter 1: Introduction to Linux System Administration

1.1 Introduction to UNIX and Linux

Linux is a true 32-bit operating system that runs on a variety of different platforms, including Intel, Sparc, Alpha, and Power-PC (on some of these platforms, such as Alpha, Linux is actually 64-bit).

Linux was first developed back in the early 1990s, by a young Finnish then-university student named Linus Torvalds. Linus had a "state-of-the-art" 386 box at home and decided to write an alternative to the 286-based Minix system (a small UNIX-like implementation primarily used in operating systems classes), to take advantage of the extra instruction set available on the then-new chip, and began to write a small bare-bones kernel.

The interesting thing about Linux is, it is completely free! Linus decided to adopt the GNU Copyleft license of the Free Software Foundation, which means that the code is protected by a copyright -- but protected in that it must always be available to others.

Free means *free* -- you can get it for free, use it for free, and you are even free to sell it for a profit (this isn't as strange as it sounds; several organizations, including Red Hat, have packaged up the standard Linux kernel, a collection of GNU utilities, and put their own "flavor" of included applications, and sell them as distributions. Some common and popular distributions are Slackware, Ubuntu, Red Hat, SuSe, and Debian)! The great thing is, you have access to source code which means you can customize the operating systems to your *own* needs, not those of the "target market" of most commercial vendors. Among most of the distributions Ubuntu is now very popular. It provides very simple gui facilities and a good command line interface. For the purpose of our demonstration examples we will use this operating system.

Linux can and should be considered a full-blown implementation of UNIX. However, it can not be called "Unix"; not because of incompatibilities or lack of functionality, but because the word "Unix" is a registered trademark owned by AT&T, and the use of the word is only allowable by license agreement. Linux is every bit as supported, as reliable, and as viable as any other operating system solution.

1.2 Linux command line

When Linus Torvalds introduced Linux and for a long time thereafter, Linux did not have a graphical user interface (GUI): I ran on character-based terminals only. All the tools ran from a command line. Today the Linux GUI is important but many people—especially system administrators—run many command line programs. Command line utilities are often faster, more powerful, or more complete than their GUI counterparts. Sometimes there is no GUI counterpart to a textual utility; some people just prefer the hands-on feeling of the command line. When you work with a command line interface, you are working with a shell.

A shell provides an interface between the user and operating system kernel. It is a command interpreter that takes commands from users and executes it.

Linux's most common command interpreter is called **bash**. **Bash** is the abbreviation of **Bourne-Again Shell**.

The shell is where commands are invoked. When started, the bash shell gives us a prompt and waits for a command to be entered. The command is typed at the shell prompt. The prompt usually ends in a dollar sign ($). After typing a command we need to press ENTER to invoke it. The shell will execute the command. Another prompt will then appear.

Shell commands consist of one or more words separated by spaces. The first word is the command to be run. Subsequent words are either options or arguments to the command. Options usually start with one or two hyphens.

Some examples of commands:

- List all the files in the current directory:

  ```
  $ ls
  ```

- List the files in the 'long format' (giving more information):

  ```
  $ ls –l
  ```

- List full information about some specific files:

  ```
  $ ls -l notes.txt report.txt
  ```

- List full information about all the .txt files:

  ```
  $ ls -l *.txt
  ```

- List all files in long format, even the hidden ones:

  ```
  $ ls -l -a
  $ ls -la
  ```

The dollar ($) represents the prompt here. We need not type it.

Most command take **parameters**. Some commands require them. Parameters are also known as **arguments**. For example the command echo simply displays its arguments.

```
$ echo

$ echo hello there
hello there
```

The first echo command outputs a blank line and the second echo command outputs its arguments.

Commands are usually case sensitive. Most of the commands are in lower case.

```
$ echo whisper
whisper
$ ECHO shout
bash: ECHO: command not found
```

Often it is desired to repeat a previously executed command. The shell keeps a **command history** for this purpose.

- We use UP and DOWN to scroll through the list of previously executed commands and then press ENTER to execute the desired command.
- Commands can also be edited before being run. The LEFT and RIGHT cursor keys navigate across a command.
- Extra characters can be typed at any point. BACKSPACE deletes characters to the left of the cursor. DEL and CTRL+D delete characters to the right.

Typically successful commands do not give any output. However, messages are displayed in the case of errors.

1.3 Files And Directories

A directory is a collection of files and/or other directories. Because a directory can contain other directories, we get a directory **hierarchy**. The top level of the hierarchy is the **root directory**. Files and directories can be named by a **path.** The root directory is referred to as /. Other directories are referred to by the **path**. The **path** consists of names separated by /. A file can also be referred to by the **path**. If it is directory, then the **path** may end with a /.

An **absolute path** starts at the root of the directory hierarchy and names directories or files under it. For example:

```
/etc/hostname
```

The above refers to a file hostname which is in the etc directory under the root (/) directory.

1.3.1 List The Names of Files In A Director: ls

We can use ls command to list files in a specific directory by specifying the specific directory:

```
$ ls /usr/share/doc/
```

The above command lists all he files and folders under the directory /usr/share/doc. If the first argument to ls is not given, then ls lists the files in current working directory of he user.

The -l option to ls gives more information, including the size of files and the date they were last modified:

```
$ ls -l
drwxrwxr-x 2 fred users 4096 Jan 21 10:57 Accounts
-rw-rw-r-- 1 fred users 345 Jan 21 10:57 notes.txt
-rw-r--r-- 1 fred users 3255 Jan 21 10:57 report.txt
```

1.3.2 Viewing And Changing Current Directory: pwd, cd

The shell has a **current directory –** the directory in which currently the logged user is working in shell. Usually after firs login, the current directory should be the home directory of the user. Some commands like ls use the current directory if none is specified. We use pwd command to see what the current directory is:

```
$ pwd
/home/fred
```

We can change the current directory with the command cd :

```
$ cd /mnt/cdrom
$ pwd
/mnt/cdrom
```

The symbol tlide (~) is an abbreviation for home directory. So for user fred the following are equivalent:

```
$ cd /home/fred/documents/
$ cd ~/documents/
```

The following are the same for user fred:

```
$ cd
$ cd ~
$ cd /home/fred
```

Paths do not have to start from the root directory. A path which does not start with the / is a **relative path**. It is relative to some other directory usually current directory. Relative paths specify files in the same way as the absolute ones. For example the following sets of directory changes end up in the same directory

```
$ cd /usr/share/doc
```

```
$ cd /
$ cd usr
```

```
$ cd share/doc
```

Every directory contains two special filenames which help making relative paths.

The directory .. points to the parent directory. **ls ..** will list files in the parent directory

For example if we start from /home/fred:

```
$ cd ..
$ pwd
/home
$ cd ..
$ pwd
/
```

The directory . points to the directory it is in. sp ./foo is the same file as foo.

The special .. and . directories can be used in paths just like any other directory names:

```
$ cd ../other-dir/
```

The above means the directory other-dir in the parent directory of the current directory. It is common to see .. to go back several directories from the current directory. The dot directory is most commonly used on its own to mean the current directory.

1.3.3 Creating Directory: mkdir

The mkdir command makes new directory under an existing directory. For example to create a directory for storing music files:

```
$ mkdir musics
```

To delete an empty directory we use rmdir command.

```
$ rmdir OldMusics
```

We use rm with –r option to delete directories and all the files (recursively) they contain.

```
$ rm -r OldMusics
```

1.3.4 Viewing Hidden Files And Directories: ls -a

The special . and .. directories don't show up when we do ls. They are **hidden files**
Files whose name starts with a dot (.) are considered hidden.

Make ls to list all files, even the hidden ones, by giving the –a option:

```
$ ls -a
. .. .bashrc .profile report.doc
```

1.4 Working With Files

This section describes utilities that copy, move, print, search through, display, sort, and compare files.

1.4.1 Display A Text File: cat

The cat utility displays the contents of a text file. The name of the command is derived from catenate, which means to join together, one after the other. A convenient way to display the contents of a file to the screen is by giving the command cat, followed by a SPACE and the filename. Figure 1.5.1 shows cat displaying the contents of practice. This figure shows the difference between the ls and cat utilities: The ls utility displays the name of a file, whereas cat displays the contents of a file.

1.4.2 Delete A File: rm

The rm (remove) utility deletes a file. Figure 1 shows rm deleting the file named practice. After rm deletes the file, ls and cat show that practice is no longer in the directory. The ls utility does not list its filename, and cat says that no such file exists.
Use rm carefully.

```
$ ls
practice
$ cat practice
This is a small file that I created
with a text editor.
$ rm practice
$ ls
$ cat practice
cat: practice: No such file or directory
$
```

1.4.3 Display A Text File One Screen At A Time: less, more

When you want to view a file that is longer than one screen, you can use either the less utility or the more utility. Each of these utilities pauses after displaying a screen of text. Because these utilities show one page at a time, they are called pagers. Although less and more are very

similar, they have subtle differences. At the end of the file, for example, less displays an EOF (end of file) message and waits for you to press q before returning you to the shell. In contrast, more returns you directly to the shell. In both utilities you can press h to display a Help screen that lists commands you can use while paging through a file. For example:

```
$ more target-file(s)
```

displays the contents of *target-file(s)* on the screen, pausing at the end of each screenful and asking the user to press a key (useful for long files). It also incorporates a searching facility (press '/' and then type a phrase that you want to look for).

1.4.4 Copy A File: cp

The cp (copy) utility (Figure 1.6.1) makes a copy of a file. This utility can copy any file, including text and executable program (binary) files. You can use cp to make a backup copy of a file or a copy to experiment with. The cp command line uses the following syntax to specify source and destination files:

```
cp source-file destination-file
```

The ***source-file*** is the name of the file that cp will copy. The ***destination-file*** is the name that cp assigns to the resulting (new) copy of the file.

```
$ ls
memo
$ cp memo memo.copy
$ ls
memo memo.copy
```

The cp command line in Figure 1.6.1 copies the file named memo to memo.copy. The period is part of the filename—just another character. The initial ls command shows that memo is the only file in the directory. After the cp command, second ls shows two files in the directory, memo and memo.copy.

1.4.5 Changes The Name Of A File : mv

The mv (move) utility can rename a file without making a copy of it. The mv command line specifies an existing file and a new filename using the same syntax as cp:

```
mv existing-filename new-filename
```

The command line in Figure 1.6.2 changes the name of the file memo to memo.0130.

The initial ls command shows that memo is the only file in the directory. After you give the mv command, memo.0130 is the only file in the directory. Compare this result to that of the earlier cp example.

```
$ ls
memo
$ mv memo memo.0130
$ ls
memo.0130
```

1.4.6 Search For A String In A File: grep

The grep utility searches through one or more files to see whether any contain a specified string of characters. This utility does not change the file it searches but simply displays each line that contains the string.

```
$ cat memo
Helen:
In our meeting on June 6 we
discussed the issue of credit.
Have you had any further thoughts
about it?
                Alex
$ grep 'credit' memo
discussed the issue of credit.
```

The grep command in Figure 1.6.3 searches through the file memo for lines that contain the string credit and displays a single line that meets this criterion. If memo contained such words as discredit, creditor, or accreditation, grep would have displayed those lines as well because they contain the string it was searching for. The –w option causes grep to match only whole words. Although you do not need to enclose the string you are searching for in single quotation marks, doing so allows you to put SPACEs and special characters in the search string.

1.5 Process Management

The kernel considers each program running on our system to be a **process**. A process 'lives' as it executes with a lifetime that may be short or long. A process is said to die when it terminates. The kernel identifies each process by a number known as process id, or **pid.** A process has a user id(**uid**) and a group id(**gid**) which together specifies what permissions it has. A process has a parent process id (**ppid**) – the **pid** of the process that has created it.

Each process has its own working directory initially inherited from its parent process. There is an environment for each process. A collection of named environment variables and their associated values. The environment is usually inherited from the parent process.

1.5.1 Process Monitoring: ps

The ps command gives a snapshot of the processes running on the system at a given moment in time. It normally shows a brief summary of each process. The command ps has many options. Some of the most commonly used are:

- -a – Show processes owned by other users
- -f – display process ancestors in a tree-like format
- -u – use the user output format, showing user names and process start times

1.5.2 Process Monitoring: pstree

It also displays a snapshot of currently running processes. It always uses a tree like display similar to ps –f . Some of the most commonly used options for pstree are:

- -a – displays command's arguments
- -c – don't compact identical subtrees
- -G – attempts to use terminal specific line-drawing characters
- -h – highlights the ancestors of the current process
- -n – sort processes numerically by pid, rather then alphabetically by name
- -p – includes pid in the output

1.5.3 Process Monitoring: top

The top command shows full-screen, continuously updated snapshots of process activity. It wais for a short period of time between each snapshots to give the illusion of real-time monitoring. Processes are displayed in descending order of how much processor time they are using. It also displays system uptime, load average, cpu status and memory information. Some of the most commonly used options for top are:

- -b – Batch mode — send snapshots to standard output
- -n num – Exit after displaying num snapshots
- -d delay – Wait delay seconds between each snapshot
- -i – Ignore idle processes
- –s – Disable interactive commands which could be dangerous the superuser

1.5.4 Signaling Processes

A process can be sent a signal by the kernel or by another process. Each signal is a very simple message: A small whole number with a mnemonic name. Signal names are all-capitals like INT. they are often written with SIG as part of the name for example: SIGINT. There are about 30 signals available not all of which are useful.

The following are the most commonly used signals:

Name	Number	Meaning

INT	2	Interrupt — stop running. Sent by the kernel when you press Ctrl+C in a terminal.
TERM	15	“Please terminate.” Used to ask a process to exit gracefully.
KILL	9	“Die!” Forces the process to stop running; it is given no opportunity to clean up after itself.
TSTP	18	Requests the process to stop itself temporarily. Sent by the kernel when you press Ctrl+Z in a terminal.
HUP	1	Hang up. Sent by the kernel when we log out, or disconnect a modem. Conventionally used by many daemons as an instruction to re-read a configuration file.

1.5.5 Sending Signals: kill

The kill command is used to send a signal to a process It is a normal executable command, but many shells also provide it as a built-in. For example to send a SIGHUP signal to a process we use either of the following two:

```
$ kill -HUP pid or
$ kill -s HUP pid
```

If we omit out the signal name in the kill command, by default kill will send a SIGTERM to the process. We can specify more than one pid to signal multiple processes at the same time.

1.5.6 Sending Signals to Daemons: pidof

On UNIX systems, long-lived processes that provide some service are often referred to as daemons. Daemons typically have a configuration file (usually under /etc) which affects their behavior. Many daemons read their configuration file only at startup. If the configuration changes, you have to explicitly tell the daemon by sending it a SIGHUP signal. We can sometimes use pidof to find the dæmon’s pid: for example, to tell the inetd dæmon to reload its configuration, we can run:

```
$ kill -HUP $(pidof /usr/sbin/inetd)
```

1.5.7 Process Priorities: nice

Not all tasks require the same amount of execution time. Linux has the concept of execution priority to deal with this. Process priority is dynamically altered by the kernel. We can view the

current priority by looking at top or ps -l and looking at the PRI column. The priority can be biased using nice. The current bias can be seen in the NI column in top.

The nice command starts a program with a given priority bias. Peculiar name: 'nicer' processes require fewer resources. Niceness ranges from +19 (very nice) to −20 (not very nice). Non-root users can only specify values from 1 to 19; the root user can specify the full range of values. Default niceness when using nice is 10.

To run a command at increased niceness (lower priority):

```
$ nice -10 long-running-command &
$ nice -n 10 long-running-command &
```

To run a command at decreased niceness (higher priority):

```
$ nice --15 important-command &
$ nice -n -15 important-command &
```

1.5.8 Modifying Priorities: renice

The command renice changes the niceness of existing processes. Non-root users are only permitted to increase a process's niceness. To set the process with pid 2984 to a higher niceness (lower priority):

```
$ renice 15 2984
```

The niceness is just a number: no extra – sign. To set the process with pid 3598 to a lower niceness (higher priority):

```
$ renice -15 3598
```

You can also change the niceness of all a user's processes:

```
$ renice 15 -u mikeb
```

1.6 Installation of Software in Linux

There are several different types of installation files for Linux, and few of them are as easy to install as the EXE installation files found on Windows. For Linux we find several different types of files: .deb, .rpm, .bin, .tar.gz, INSTALL, .sh, etc. These different files all have a different method of execution. Below are instructions on installing these filetypes. The following section assumes that we are running Ubuntu Linux system.

1.6.1 Using apt-get

Ubuntu has something called *apt-get*, which allows you to draw from a set of online repositories (stored in the /etc/apt/sources.list file) that house packages (i.e., programs/software). The *apt-get* command does several things at once—it downloads the appropriate files, downloads all their dependencies, and installs all of them. A single command installs the software. You don't have to download a separate installer file or unzip or go through a wizard or reboot. For example, if I wanted to install Thunderbird, I'd type these commands in a terminal:

```
$ sudo apt-get update
$ sudo apt-get install thunderbird
```

The first command looks both at what I have installed and what's available in the repositories. The second command downloads the packages needed for Thunderbird and installs them.

Another great thing about *apt-get* is the ability to install several different packages at once. For example, if I wanted to install not only Thunderbird but Firefox, GIMP, Inkscape, Juk, and Wine, I could type in these commands:

```
$ sudo apt-get update
$ sudo apt-get install thunderbird firefox gimp inkscape
juk wine
```

And all of those packages would download and install themselves.

This is the best way of installing software in Ubuntu because it automatically resolves all dependencies and installs them.

1.6.2 Configuring the sources.list File

The *sources.list* file resides in the */etc/apt* directory. Like most other Linux configuration files, it can be revised by using an ordinary text editor, such as ae.

The file contains a series of lines, each specifying a source for packages. The lines are consulted serially, so it's usually advantageous to place lines that specify local sources - such as a CD-ROM - ahead of lines that specify remote sources. Doing so can save many minutes of download time.

Each line has the form:

```
deb
 uri distribution components
```

The *uri* is a universal resource identifier (URI) the specifies the computer on which the packages reside, the location of the packages, and the protocol used for accessing the packages. It has the following form:

protocol://
host/
path

Four protocols - sometimes called URI types - are recognized:

cdrom	A local CD-ROM drive.
file	A directory of the local filesystem.
http	A Web server.
ftp	An FTP server.

The *host* part of the URI and the preceding pair of slashes (//) are used only for the *http* and *ftp* protocols. There, the *host* part of the URI gives the name of the host that contains the packages.

The *path* part of the URI always appears, with the preceding slash (/). It specifies the absolute path of the directory that contains the packages.

Here are some examples of typical URIs:

```
cdrom:/cdrom
cdrom:/mnt/cdrom
file:/mnt
file:/debian
http://www.us.debian.org/debian
http://non-us.debian.org/debian-non-US
ftp://ftp.debian.org/debian
ftp://nonus.debian.org/debian-non-US
```

The distribution part of a *sources.list* line specifies the distribution release that contains the packages. Typical values include:

- **stable** : The latest stable release; that is, one that is commonly regarded as having sufficiently few serious bugs for everyday use.

- **unstable** : The latest unstable release. This release sometimes contains serious bugs and should not be installed by users who require high levels of system availability or reliability.

The components part of a *sources.list* line specifies the parts of the distribution that will be accessed. Typical values include:

- main: The main set of packages.
- contrib.: Packages not an integral part of the distribution, but which may be useful.
- non-free: Packages that contain software distributed under terms too restrictive to allow inclusion in the distribution, but which may be useful.

A typical *sources.list* file might contain the following entries:

```
deb file:/cdrom stable main contrib
deb http://www.us.debian.org/debian stable main contrib non-
free
deb http://non-us.debian.org/debian-non-US stable non-US
```

This configuration allows rapid access to the distribution packages contained on the local CD-ROM. It also allows convenient access via the network to other packages and more recent package versions stored on web servers.

1.6.3 Using apt-get

Once you've configured sources.list, you can use apt-get to update information on available packages, to install a package, or to upgrade installed packages.

1.6.3.1 Updating Information on Available Packages

To update information on available packages, issue the following command:

```
$ sudo apt-get update
```

1.6.3.2 Installing a Package

To install a specified package, issue the following command:

```
$ sudo apt-get install <package>
```

where *package* specifies the name of the package to be installed.

1.6.3.3 Upgrading Installed Packages

To automatically upgrade all installed packages to the latest available version, issue the following command:

```
$ sudo apt-get upgrade
```

1.6.7 Installing DEB files

A .deb file is the easiest file to install on Ubuntu--if you are given an option for the type of file you want to download, choose this option. Save the file to your Desktop. Once it is there, simply

double click on the file and the system package installer will open. Click the button in the top right corner that says "Install Package", and wait for it to say finished. Close the window. Your application is now installed and ready to use.

1.6.8 Installing RPM files

DEB files are the default installation file for Ubuntu--if at all possible, you should choose a .deb file over any other file type. However, sometimes an application is only available in one or two formats, none of which are Ubuntu-flavored. RPM is one such file type. In order to install this file, you will need to convert it into something Ubuntu knows how to install--a .deb file!

To do this, open the Terminal and type:

```
$ sudo apt-get install alien
```

You will be prompted to enter your password. After entering, press the return key. You will see some code scroll by quickly, and then you will be presented with the option to continue or quit the installation. Type 'Y' and press the return key.

You will see the Alien application installing; this app will be used to convert your RPM file into a DEB file. Installation could take several minutes depending on your Internet and computer speeds.

Once finished, move the RPM file to your Desktop and open the Terminal. Type: cd Desktop. This will point your Terminal to your Desktop directory where you have the RPM file saved.

Now, to install the RPM file, in the Terminal, type:

```
$ sudo alien -k filename.rpm
```

Replace filename.rpm with the actual name of the RPM file, then press the return key. It will convert the RPM file to a DEB file. Once finished, install the DEB file using the method above.

1.6.9 Install BIN files

A BIN file is similar to an RPM file, in that Ubuntu can't understand how to install it until you convert it into a different format. To do this, follow these instructions.

Download and save the BIN file to your systems Desktop. Once saved, open the Terminal and type:

```
$ cd Desktop
```

Once you've cd'ed to the Desktop, type the following line into the Terminal:

```
$ sudo chmod +x filename.bin
```

Change filename.bin to the name of your BIN file and press the return key. Nothing will show up in the Terminal, nothing will be copied to the Desktop--it will appear as if nothing at all happened. This is not the case, however, so do not worry. Type ./filename.bin and press the return key.

The program will install from within the Terminal.

Chapter 2: Compressing And Archiving Files

Large files use a lot of disk space and take longer than smaller files to transfer from one system to another over a network. If you do not need to look at the contents of a large file very often, you may want to save it on a CD, DVD, or another medium and remove it from the hard disk. If you have a continuing need for the file, retrieving a copy from a CD may be inconvenient. To reduce the amount of disk space you use without removing the file entirely, you can compress the file without losing any of the information it holds. Similarly a single archive of several files packed into a larger file is easier to manipulate, upload, download, and email than multiple files. You may frequently download compressed, archived files from the Internet. The utilities described in this section compress and decompress files and pack and unpack archives.

2.1 Compress A File Using: bzip2

The bzip2 utility compresses a file by analyzing it and recoding it more efficiently.
The new version of the file looks completely different. In fact, because the new file contains many nonprinting characters, you cannot view it directly. The bzip2 utility works particularly well on files that contain a lot of repeated information, such as ext and image data, although most image data is already in a compressed format. The following example shows a boring file. Each of the 8,000 lines of the letter_e file contains 72 e's and a NEWLINE character that marks the end of the line. The file occupies more than half a megabyte of disk storage.

```
$ ls -l
-rw-rw-r--  1 sam sam 584000 Mar  1 22:31 letter_e
```

The –l (long) option causes ls to display more information about a file. Here it shows that letter_e is 584,000 bytes long. The —verbose (or –v) option causes bzip2 to report how much it was able to reduce the size of the file. In this case, it shrank the file by 99.99 percent:

```
$ bzip2 -v letter_e
letter_e: 11680.00:1, 0.001 bits/byte, 99.99% saved, 584000
in, 50 out.
$ ls -l
-rw-rw-r--  1 sam sam 50 Mar  1 22:31 letter_e.bz2
```

Now the file is only 50 bytes long. The bzip2 utility also renamed the file, appending .bz2 to its name. This naming convention reminds you that the file is compressed; you would not want to display or print it, for example, without first decompressing it. The bzip2 utility does not change the modification date associated with the file, even though it completely changes the file's contents.

In the following, more realistic example, the file zach.jpg contains a computer graphics image:

```
$ ls -l
-rw-r--r--  1 sam sam 33287 Mar  1 22:40 zach.jpg
```

The bzip2 utility can reduce the size of the file by only 28 percent because the image is already in a compressed format:

```
$ bzip2 -v zach.jpg
zach.jpg:  1.391:1,  5.749 bits/byte, 28.13% saved, 33287
in, 23922 out.
```

2.2 Decompress A File Using: bunzip2

You can use the bunzip2 utility to restore a file that has been compressed with bzip2:

```
$ bunzip2 letter_e.bz2
$ ls -l
-rw-rw-r--  1 sam sam 584000 Mar  1 22:31 letter_e
$ bunzip2 zach.jpg.bz2
$ ls -l
-rw-r--r--  1 sam sam  33287 Mar  1 22:40 zach.jpg
```

This command is similar to bzip2 with –d option.

2.3 Compress A File Using: gzip

The gzip (GNU zip) utility is older and less efficient than bzip2. Its flags and operations are very similar to those of bzip2. A file compressed by gzip is marked by a .gz filename extension. Linux stores manual pages in gzip format to save disk space; likewise, files you download from the Internet are frequently in gzip format. Use gzip and gunzip just as you would use bzip2 and bunzip2 respectively.

2.4 Archiving Files: tar

The tar utility performs many functions. Its name is short for tape archive, as its original function was to create and read archive and backup tapes. Today it is used to create a single file (called a tar file, archive, or tarball) from multiple files or directory hierarchies and to extract files from a tar file.

.

In the following example, the first ls shows the existence and sizes of the files g, b, and d. Next tar uses the –c (create), –v (verbose), and –f (write to or read from a file) options to create an archive named all.tar from these files. Each line output displays the name of the file tar is appending to the archive it is creating. The tar utility adds overhead when it creates an archive. The next command shows that the archive file all.tar occupies about 9,700 bytes, whereas the sum of the sizes of the three files is about 6,000 bytes. This overhead is more appreciable on smaller files, such as the ones in this example.

```
$ ls -l g b d
-rw-r--r--   1 jenny jenny 1302 Aug 20 14:16 g
-rw-r--r--   1 jenny other 1178 Aug 20 14:16 b
-rw-r--r--   1 jenny jenny 3783 Aug 20 14:17 d
$ tar -cvf all.tar g b d
g
b
d
$ ls -l all.tar
-rw-r--r--   1 jenny jenny 9728 Aug 20 14:17 all.tar
$ tar -tvf all.tar
-rw-r--r-- jenny/jenny    1302 2007-08-20 14:16 g
-rw-r--r-- jenny/other    1178 2007-08-20 14:16 b
-rw-r--r-- jenny/jenny    3783 2007-08-20 14:17 d
```

The final command in the preceding example uses the –t option to display a table of contents for the archive. Use –x instead of –t to extract files from a tar archive. Omit the –v option if you want tar to do its work silently.

You can use bzip2, or gzip to compress tar files, making them easier to store and handle. Many files you download from the Internet will already be in one of these formats. Files that have been processed by tar and compressed by bzip2 frequently have a filename extension of .tar.bz2 or .tbz. Those processed by tar and gzip have an extension of .tar.gz or .tz extension.

You can unpack a tarred and gzipped file in two steps. (Follow the same procedure if the file was compressed by bzip2, but use bunzip2 instead of gunzip.) The next example shows how to unpack the GNU make utility.

```
$ ls -l mak*
-rw-rw-r--  1 sam sam 1211924 Jan 20 11:49
make-3.80.tar.gz
$ gunzip mak*
$ ls -l mak*
-rw-rw-r--  1 sam sam 4823040 Jan 20 11:49
make-3.80.tar
$ tar -xvf mak*
make-3.80/
make-3.80/po/
make-3.80/po/Makefile.in.in
...
make-3.80/tests/run_make_tests.pl
make-3.80/tests/test_driver.pl
```

The first command lists the downloaded tarred and gzipped file: make-3.80.tar.gz (about 1.2 megabytes). The asterisk (*) in the filename matches any characters in any filenames, so you end up with a list of files whose names begin with mak; in this case there is only one. Using an asterisk saves typing and can improve accuracy with long filenames. The gunzip command decompresses the file and yields make-3.80.tar (no .gz extension), which is about 4.8 megabytes. The tar command creates the make-3.80 directory in the working directory and unpacks the files into it.

```
$ ls -ld mak*
drwxrwxr-x  8 sam sam    4096 Oct  3  2002 make-3.80
-rw-rw-r--  1 sam sam 4823040 Jan 20 11:49 make-3.80.tar
$ ls -l make-3.80
total 1816
-rw-r--r--  1 sam sam  24687 Oct  3  2002 ABOUT-NLS
-rw-r--r--  1 sam sam   1554 Jul  8  2002 AUTHORS
-rw-r--r--  1 sam sam  18043 Dec 10  1996 COPYING
-rw-r--r--  1 sam sam  32922 Oct  3  2002 ChangeLog
...
-rw-r--r--  1 sam sam  16520 Jan 21  2000 vmsify.c
-rw-r--r--  1 sam sam  16409 Aug  9  2002 vpath.c
drwxrwxr-x  5 sam sam   4096 Oct  3  2002 w32
```

After tar extracts the files from the archive, the working directory contains two files whose names start with mak: make-3.80.tar and make-3.80. The –d (directory) option causes ls to display only file and directory names, not the contents of directories as it normally does. The final ls command shows the files and directories in the make-3.80 directory.

Chapter 3: Mange File Ownership

3.1 Users and Groups

Anyone using a Linux computer is a user. The system keeps track of different users, by username. Security features allow different users to have different privileges. Users can belong to groups, allowing security to be managed for collections of people with different requirements. We use the su command to switch to a different user. It is quicker than logging off and back on again. The command su prompts us for the user's password:

```
$ su - bob
Password:
```

The – option makes su behave as if we've logged in as that user.

3.2 The Superuser: Root

Every Linux system has a user called 'root'. The root user is all-powerful. It can access any files. The root user account should only be used for system administration, such as installing software. When logged in as root, the shell prompt usually ends in #. It is usually best to use su for working as root:

```
$ whoami
fred
$ su -
Password:
# whoami
root
```

3.3 Changing File Ownership: chown

The chown command changes the ownership of files or directories. This is a security feature. Only the superuser can change the ownership of a file. Simple usage follows:

```
# chown bob logfile.txt
```

The above command makes logfile.txt to be owned by the user bob.
We can specify any number of files or directories as arguments in the command.

3.4 Changing File Ownership: chgrp

The chgrp command changes the group ownership of files or directories. Simple usage follows:

```
# chgrp staff report.txt
```

The above command makes staff be the group owner of the file logfile.txt
As for chown, we can specify any number of files or directories. The superuser can change the group ownership of any file to any group. The owner of a file can also change its group ownership. But only to a group of which the owner is a member

3.5 Changing the Ownership of a Directory and Its Contents

A common requirement is to change the ownership of a directory and its contents. Both chown and chgrp accept a -R (Mnemonic: 'recursive') option:

```
# chgrp -R staff shared-directory
```

The above command changes the group ownership of shared-directory and its contents and its subdirectories, recursively to staff. Changing user ownership (superuser only):

```
# chown -R root /usr/local/share/misc/
```

3.6 Manage File Permission to Control Access to Files

A permission represents an action that can be done on the file. There are three types of permissions to a file; each denoted by a letter:

Permission	Letter	Description
Read	r	Permission to read the data stored in the file
Write	w	Permission to write new data to the file, to truncate the file, or to overwrite existing data
Execute	x	Permission to attempt to execute the contents of the file as a program

The r,w,x permissions also have a meaning for directories:

Permission	Letter	Description
Read	r	Permission to get a listing of the directory
Write	w	Permission to create, delete, or rename files (or subdirectories) within the directory
Execute	x	Permission to change to the directory, or to use the directory as an intermediate part of a path to a file

As well as having different types of permission, we can apply different sets of permissions to different sets of people. A file (or directory) has an owner and a group owner. The r,w,x permissions are specified separately for the owner, for the group owner, and for everyone else (the 'world').

3.7 Examining Permission of a file: ls –l

The ls -l command allows us to look at the permissions on a file:

```
$ ls -l
drwxr-x--- 9 aaronc staff 4096 Oct 12 12:57 accounts
-rw-rw-r-- 1 aaronc staff 11170 Dec 9 14:11 report.txt
```

The third and fourth columns are the owner and group-owner. The first column specify the permissions:

- one character for the file type: d for directories, – for plain files.
- three characters of rwx permissions for the owner (or a dash if the permission isn't available)
- three characters of rwx permissions for the group owner
- three characters of rwx permissions for everyone else

If someone owns a file, then per-owner permissions apply to him. Otherwise, if he is in the group that group-owns the file, then per-group permissions apply to him. If neither of those is the case, then for-everyone-else permissions apply to him.

3.8 Changing Permissions of Files and Directories: chmod

The chmod command changes the permissions of a file or directory. A file's permissions may be changed only by its owner or by the superuser. The command chmod takes an argument describing the new permissions. The permissions can be specified in many flexible (but correspondingly complex) ways.

Permissions can be set using letters in the following format:

[ugoa][+=-][rwx]

- The first letters indicate who to set permissions for: u for the file's owner, g for the group owner, o for other users, or a for all users
- = sets permissions for files, + adds permissions to those already set, and – removes permissions
- The final letters indicate which of the r,w,x permissions to set

For example if we want to add executable permission for a program named bubblesort to all users, we type the following command:

```
$ chmod a+x bubblesort
```

We may use numerical permissions with chmod. Three decimal numbers identify permissions for owner, group and others. The number in binary format should be interpreted as follows:

Decimal: 664
Bianry: 110 110 100
Meaning: rwx rwx rwx
Explanation: A '1' in each position specifies 'permission', a '0' specifies 'no permission'.

For example:

```
$ chmod 664 bubblesort
```

The above command is equivalent to:

```
$ chmod ug=rw,o=r bubblesort
```

A common requirement is to change the permissions of a directory and its contents. The command chmod accepts a –R (Mnemonic: 'recursive') option:

```
$ chmod -R g+rwX,o+rX public-directory
```

The above command

- Adds rwx permissions on public-directory for the group owner, and adds rx permissions on it for everyone else
- And any subdirectories, recursively
- Any any contained executable files
- Contained non-executable files have rw permissions added for the group owner, and r permission for everyone else

3.9 Special Directory Permissions: Sticky

The /tmp directory must be world-writable, so that anyone may create temporary files within it. But that would normally mean that anyone may delete any files within it — obviously a security hole. A directory may have 'sticky' permissions: Only a file's owner may delete it from a sticky directory. Expressed with a t (mnemonic: temporary directory) in a listing:

```
$ ls -l -d /tmp
drwxrwxrwt 30 root root 11264 Dec 21 09:35 /tmp
```

We enable 'sticky' permission with the following command:

```
# chmod +t /data/tmp
```

3.10 Special Directory Permissions: Setgid

If a directory is setgid ('set group-id'), files created within it acquire the group ownership of the directory and directories created within it acquire both the group ownership and setgid permission. It is useful for a shared directory where all users working on its files are in a given group. It is expressed with an s in 'group' position in a listing:

```
$ ls -l -d /data/projects
drwxrwsr-x 16 root staff 4096 Oct 19 13:14 data/projects
```

We enable setgid with:

```
# chmod g+s /data/projects
```

Chapter 4: FileSystem: Mouning and Unmouning

A filesystem in this context is a hierarchy of directories that is located on a single partition (logically independent section of a hard disk drive) or other *device*, such as a CDROM, DVD, floppy disk or USB key drive, and has a single *filesystem type* (i.e., method for organizing data).
As far as many parts of a Linux system are concerned, a partition contains entirely arbitrary data. When installing, we set things up so that a partition contains a filesystem — a way of organising data into files and directories. One filesystem is made the root filesystem: the root directory on that filesystem becomes the directory named /. Other filesystems can be mounted: the root directory of that filesystem is grafted onto a
directory of the root filesystem. This arranges for every file in every mounted filesystem to be accessible from a single unified name space. The directory grafted onto is called the mount point.

4.1 Mounting filesystem: mount

Mounting refers to logically attaching a filesystem to a specified location on the currently accessible (and thus already mounted) filesystem(s) on a computer system so that its contents can be accessed by users.

Important filesystems are mounted at boot-up; other filesystems can be mounted or unmounted at any time. The mount command mounts a filesystem. We usually need to have root permission to mount a filesystem. The mount command makes it easy to mount filesystems configured by the system administrator. For example, many systems are configured so that the following command:

```
# mount /mnt/cdrom
```

will mount the contents of the machine's CD-ROM drive under the directory /mnt/cdrom

```
# mount /dev/sdb3 /mnt/extra
```

The above command mounts the filesystem stored in the /dev/sdb3 device on the mount point /mnt/extra. We may occasionally need to specify the filesystem type explicitly:

```
# mount -t vfat /dev/hdd1 /mnt/windows
```

Allowable filesystem types are listed in the mount(8) manpage. To see a list of the filesystems currently mounted, run mount without any options.

The /etc/fstab file contains information about filesystems that are known to the system administrator. Specifying a filesystem in /etc/fstab makes it possible to use its mount point as the only argument to mount. /etc/fstab also configures which filesystems should be mounted at boot-up Each line in /etc/fstab describes one filesystem. There are six columns on each line.

Sample /etc/fstab is shown below:

Device	Mount-point	Type	Options	Dump	Pass-no
/dev/hda3	/	Ext2	Defaults	1	1
/dev/hda1	/bot	Ext2	Defauls	1	2
/dev/hda5	/usr	Ext2	Defaults	1	2`
/dev/hdb1	/usr/local	Ext2	Defaults	1	2
/dev/hdb2	/home	Ext2	Defaults	1	2
/dev/scd0	/mnt/cdrom	Iso9660	Noauto, users,ro	0	0
/dev/fd0	/mnt/floppy	Auto	Noauto, users	0	0

The most common filesystem types are:

- ext2 – The standard Linux filesystem
- iso9660 – The filesystem used on CD-ROMs
- proc – Not a real filesystem, so uses none as the device. Used as a way for the kernel to report system information to user processes
- vfat – The filesystem used by Windows 95
- auto – Not a real filesystem type. Used as a way of asking the mount command to probe for various filesystem types, particularly for removable media
- Networked filesystems include nfs (Unix-specific) and smbfs (Windows or Samba)
- Other, less common types exist; see mount(8)

There are comma-separated options in /etc/fstab. Alternatively, use comma-separated options with -o on the mount command line. Common mount options:

- Noauto – In /etc/fstab, prevents the filesystem being mounted at bootup. Useful for removable media

- ro – Mount the filesystem read-only
- users – Let non-root users mount and unmount this filesystem
- user – Like users, but non-root users can only unmount filesystems that they themselves mounted

Other less common mount options exist, as well as many options for individual filesystem types – see mount(8).

The fifth column is called dump. It is used by the dump and restore backup utilities. Few people use those tools. We just use 1 for normal filesystems, and 0 for removable filesystems.

The sixth column is called pass-no. This controls the order in which automatically-mounted filesystems are checked by fsck.We use 1 for the root filesystem and 0 for filesystems that aren't mounted at boot-up. We use 2 for other filesystems.

4.2 Unmounting Filesystem: umount

Unmounting refers to logically detaching a filesystem from the currently accessible filesystem(s). All mounted filesystems are unmounted automatically when a computer is shut down in an orderly manner. However, there are times when it is necessary to unmount an individual filesystem while a computer is still running. A common example is when it is desired to remove an external device such as a USB key drive; should such device be removed before the filesystem on it is properly unmounted, it is possible that any data recently added to it might not be saved. The basic syntax of umount is:

```
# umount [options] filesystem
```

umount is most commonly used without any of its several options. The filesystem is identified by the full pathname of the directory in which it has been mounted, not by its type. Thus, for example, to unmount a filesystem that is mounted in a directory called /dir1, all that would be necessary is to type in the following at the keyboard and press the Enter key:

```
# umount /dir1
```

Likewise, a USB key device, assuming that it had been mounted in the directory /mnt/usb, would be unmounted with the following:

```
# umount /mnt/usb
```

Attempts to unmount a filesystem are not always successful. The most common problem is that the filesystem is busy. That is, it is currently being used by some process (i.e., instance of a program in execution). In such case an error message such as umount: /dir1: device is busy will be displayed on the screen. This busy state could be the result of something as simple as an GUI window being open that shows an icon of the directory containing the filesystem, in which case it can be easily solved by closing the window. Or it could be the result of a file on that filesystem being open, in which case all that is necessary is to close the file. In less obvious cases, it may be

necessary to use a command such as ps or pstree to try to locate the offending process(es) and then use a command such as kill to terminate such process(es).

Another cause of failure is when a user attempts to unmount a filesystem that has already been unmounted. In such case an error message such as umount: /dir1: not mounted will be returned.

In the event that the unmounting is successful, umount usually works silently; that is, there is no message on the screen to confirm its success. However, umount can be made to provide such a message by using the -v (i.e., verbose) option. (This should not be confused with the -V option, which merely returns information about the currently installed version of umount.)

umount allows the name of the physical device on which the filesystem is mounted to be included in the command if desired. This is convenient because it can minimize typing by allowing the user to utilize the upward pointing arrow on the keyboard to display the command that was previously used to mount that filesystem (i.e., to use the history command) and then merely insert the letter u before the word mount and press the Enter key in order to unmount the filesystem. Thus, for example, if a filesystem that is physically located on the second partition of the first HDD (which is designated by dev/hda2) is mounted in a directory called /dir2, it can be unmounted with either of the following:

```
# umount /dir2
or
# umount /dev/hda2 /dir2
```

Interestingly, when the physical device is included, a confirmation message is automatically supplied.

There are several options that can be tried in the event that umount refuses to unmount a filesystem for no immediately apparent reason. Perhaps the most useful is the -l (i.e., lazy) option, which immediately detaches the filesystem from the main filesystem and then cleans up all references to the unmounted filesystem as soon as it is no longer busy. This capability requires Linux kernel 2.4.11 or later.

Another way to deal with an unmounting failure is to use the -r option, which remounts the filesystem as read-only. This presumably allows devices or media to be removed without affecting data which has just been written to them. In addition, the -f option forces unmounting in the case of an unreachable NFS (network filesystem) filesystem.
The -a option causes all of the filesystems described in /etc/mtab to be unmounted. (However, with umount version 2.7 and later the proc filesystem is not unmounted.) /etc/mtab is a file that is similar to /etc/fstab and which is updated by mount and umount whenever filesystems are mounted or unmounted. The -n option causes unmounting to occur without writing to /etc/mtab.

The -t option followed by the filesystem type indicates that the actions should only be taken on filesystems of that type. Multiple types can be specified in a comma-separated list. This list can be prefixed with the word no to specify filesystem types on which no action should be taken.

The -O options indicate that the actions should only be taken on filesystems with the specified options in /etc/fstab. Multiple option types can be specified in a comma-separated list. Those options for which no action should be taken can be prefixed with no.
umount will free any loop device associated with a mounted filesystem if it finds the option loop=... in /etc/mtab or if the -d option is used. A loop device is a pseudo-device that is able to redirect and transform data that goes through its loop and which is used mainly used for encrypting filesystems.

Note the symmetry between the umount and mount commands, including the fact that many of the options are identical or very similar (including -a, -h, -r, -t, -O, -v and -V). This is consistent with the Unix philosophy, a fundamental component of which is simplicity (and hence consistency to the extent practical among commands), in that it eliminates unnecessary complexity.

umount could have instead been called unmount. This might have simplified things for people who are new to the command line (i.e., text-only operation). However, eliminating unnecessary typing is also a part of the Unix philosophy, and thus the n was not used.

Chapter 5: Managing User Accounts

5.1 What is an Account?

When a computer is used by many people it is usually necessary to differentiate between the users, for example, so that their private files can be kept private. This is important even if the computer can only be use by a single person at a time, as with most microcomputers. Thus, each user is given a unique username, and that name is used to log in. There's more to a user than just a name, however. An account is all the files, resources, and information belonging to one user. The term hints at banks, and in a commercial system each account usually has some money attached to it, and that money vanishes at different speeds depending on how much the user stresses the system. For example, disk space might have a price per megabyte and day, and processing time might have a price per second

5.2 Creating User Account: adduser

To create a user account, you use the adduser command, which has the form:

```
# adduser userid
```

where userid specifies the name of the user account that you want to create. The command prompts you for the information needed to create the account.

Here's a typical example of using the command, which creates a user account named newbie:

```
# adduser newbie
Adding user newbie...
Adding new group newbie (1001).
Adding new user newbie (1001) with group newbie.
Creating home directory /home/newbie.
Copying files from /etc/skel
Changing password for newbie
Enter the new password (minimum of 5, maximum of 8
characters)
Please use a combination of upper and lower case letters
and numbers.
Re-enter new password:
Password changed.
Changing the user information for newbie
Enter the new value, or press return for the default
Full Name []: Newbie Dewbie
Room Number []:
Work Phone []:
```

```
Home Phone []:
Other []:
Is the information correct? [y/n]
y
#
```

Notice that the lines where the password was typed were overwritten by the subsequent lines. Moreover, for security, passwords are not echoed to the console as they are typed. Notice also that several of the information fields were omitted - for example, Room Number. You can specify such information if you think it may be useful, but the system makes no use of the information and doesn't require you to provide it. The similarly named useradd command also creates a user account, but does not prompt you for the password or other information.

5.3 Changing a User's name: chfn

You can change the name associated with a user account, by using the chfn command:

```
# chfn -f   name userid
```

where name specifies the new name and userid specifies the account to be modified. If the name contains spaces or other special characters, it should be enclosed in double quotes ("). For example, to change the name associated with the account newbie to Dewbie Newbie, you would enter the following command:

```
# chfn -f "Dewbie Newbie" newbie
```

5.4 Changing a User Account's Password: passwd

From time to time, you should change your password, making it more difficult for others to break into your system. As system administrator, you may sometimes need to change the password associated with a user's account. For instance, some users have a bad habit of forgetting their password. They'll come to you, the system administrator, seeking help in accessing their account.

To change a password, you use the passwd command. To change your own password, enter a command like this one:

```
$ passwd
```

This command changes the password associated with the current user account. You don't have to be logged in as root to change a password. Because of this, users can change their own passwords without the help of the system administrator. The root user, however, can change

the password associated with any user account, as you'll see shortly. Of course, only root can do so - other users can change only their own password.

The passwd command initiates a simple dialog that resembles the following:

```
$ passwd
Changing password for newbie
Old password:
Enter the new password (minimum of 5, maximum of 8
characters)
Please use a combination of upper and lower case letters
and numbers.
New password:
Re-enter new password:
Password changed.
```

Notice the restrictions governing the choice of password, which are designed to prohibit passwords that might be easily guessed. If you choose a password that violates these restrictions, the command will refuse the password, prompting you for another.

As the root user, you can change the password associated with any user account. The system doesn't ask you for the current password, it immediately prompts for the new password:

```
# passwd newbie
Changing password for newbie
Enter the new password (minimum of 5, maximum of 8
characters)
Please use a combination of upper and lower case letters
and numbers.
New password:
Re-enter new password:
Password changed.
```

Information on users is stored in the file /etc/passwd, which you can view using a text editor. Any user can read this file, though only the root user can modify it. If you selected shadow passwords, passwords are encrypted and stored in the file /etc/shadow, which can be read only by the root user.

5.5 Configuring Group Definitions

Linux uses groups to define a set of related user accounts that can share access to a file or directory. You probably won't often find it necessary to configure group definitions, particularly if you use your system as a desktop system rather than a server. However, when you wish, you create and delete groups and modify their membership lists.

5.6 Creating a Group: groupadd

To create a new group, use the groupadd command:

```
# groupadd group
```

where group specifies the name of the group to be added. Groups are stored in the file /etc/group, which can be read by any user but modified only by root.

For example, to add a group named newbies, you would enter the following command:

```
# groupadd newbies
```

5.7 Deleting a Group

To delete a group, user the groupdel command:

```
# groupdel group
```

where group specifies the name of the group to be deleted. For example, to delete the group named newbies, you would enter the following command:

```
# groupdel newbies
```

5.8 Adding a member to a group

To add a member to a group, you use a special form of the adduser command:

```
# adduser user group
```

where user specifies the member and group specifies the group to which the member is added. For example, to add the user newbie01 to the group newbies, you would enter the following command:

```
# adduser newbie01 newbies
```

5.9 Removing a member from a group

Unfortunately, no command removes a user from a specified group. The easiest way to remove a member from a group is by editing the /etc/group file. Here's an excerpt from a typical /etc/group file:

```
users:x:100:
nogroup:x:65534:
bmccarty:x:1000:
newbies:x:1002:newbie01,newbie02,newbie03
```

Each line in the file describes a single group and has the same form as other lines, consisting of a series of fields separated by colons (:). The fields are:

- Group name ： The name of the group.
- Password : The encrypted password associated with the group. This field is not generally used, containing an x instead.
- Group ID ： The unique numeric ID associated with the group.
- Member list ： A list of user accounts, with a comma (,) separating each user account from the next.

To remove a member from a group, first create a backup copy of the /etc/group file:

```
# cp /etc/group /etc/group.SAVE
```

The backup can prove helpful if you modify the file incorrectly. Next, open the /etc/group file in a text editor. Locate the line that describes the group and delete the user name and the following comma, if any. Save the file, exit the editor, and check your work.

5.10 Deleting a User Account

To delete a user account, use the userdel command:

```
# userdel user
```

where user specifies the account to be deleted. If you want to delete the user's home directory, its files and subdirectories, use this form of the command:

```
# userdel -r user
```

Chapter 6: Samba File Server

One of the most common ways to network Ubuntu and Windows computers is to configure Samba as a File Server. This section covers setting up a **Samba** server to share files with Windows clients.

The server will be configured to share files with any client on the network without prompting for a password.

6.1 Installation

The first step is to install the **samba** package. From a terminal prompt enter:

```
$ sudo apt-get install samba
```

That's all there is to it; you are now ready to configure Samba to share files.

6.2 Configuration

The main Samba configuration file is located in `/etc/samba/smb.conf`. The default configuration file has a significant amount of comments in order to document various configuration directives (Not all the available options are included in the default configuration file. See the smb.conf **man** page).

First, edit the following key/value pairs in the *[global]* section of `etc/samba/smb.conf`:

```
workgroup = EXAMPLE
...
security = user
```

The *security* parameter is farther down in the [global] section, and is commented by default. Also, change *EXAMPLE* to better match your environment.

1. Create a new section at the bottom of the file, or uncomment one of the examples, for the directory to be shared:

```
[share]
    comment = Ubuntu File Server Share
    path = /srv/samba/share
    browsable = yes
    guest ok = yes
    read only = no
    create mask = 0755
```

- *comment:* a short description of the share. Adjust to fit your needs.

- *path:* the path to the directory to share.

  This example uses /srv/samba/sharename because, according to the *Filesystem Hierarchy Standard (FHS)*, /srv is where site-specific data should be served. Technically Samba shares can be placed anywhere on the filesystem as long as the permissions are correct, but adhering to standards is recommended.

- *browsable:* enables Windows clients to browse the shared directory using **Windows Explorer**.
- *guest ok:* allows clients to connect to the share without supplying a password.
- *read only:* gives write access to the shared directory.
- *create mask:* determines the permissions new files will have when created.

2. Now that **Samba** is configured, the directory needs to be created and the permissions changed. From a terminal enter:

```
$ sudo mkdir -p /srv/samba/share
$ sudo chown nobody.nogroup /srv/samba/share/
```

The *-p* switch tells mkdir to create the entire directory tree if it doesn't exist. Change the share name to fit your environment.

3. Finally, restart the **samba** services to enable the new configuration:

```
$ sudo /etc/init.d/samba restart
```

Once again, the above configuration gives all access to any client on the local network.

From a Windows client you should now be able to browse to the Ubuntu file server and see the shared directory. To check that everything is working try creating a directory from Windows.

To create additional shares simply create new *[dir]* sections in `/etc/samba/smb.conf`, and restart *Samba*. Just make sure that the directory you want to share actually exists and the permissions are correct.

6.3 Securing a Samba File and Print Server

Samba Security Modes

There are two security levels available to the Common Internet Filesystem (CIFS) network protocol *user-level* and *share-level*. Samba's *security mode* implementation allows more flexibility, providing four ways of implementing user-level security and one way to implement share-level:

- *security = user:* requires clients to supply a username and password to connect to shares. Samba user accounts are separate from system accounts, but the **libpam-smbpass** package will sync system users and passwords with the Samba user database.
- *security = domain:* this mode allows the Samba server to appear to Windows clients as a Primary Domain Controller (PDC), Backup Domain Controller (BDC), or a Domain Member Server (DMS).
- *security = ADS:* allows the Samba server to join an Active Directory domain as a native member.
- *security = server:* this mode is left over from before Samba could become a member server, and due to some security issues should not be used. See the Server Security section of the Samba guide for more details.
- *security = share:* allows clients to connect to shares without supplying a username and password.

The security mode you choose will depend on your environment and what you need the Samba server to accomplish.

Security = User

First, install the **libpam-smbpass** package which will sync the system users to the Samba user database:

```
$ sudo apt-get install libpam-smbpass
```

If you chose the *Samba Server* task during installation **libpam-smbpass** is already installed.

Edit /etc/samba/smb.conf, and in the *[share]* section change:

```
guest ok = no
```

Finally, restart Samba for the new settings to take effect:

```
$ sudo /etc/init.d/samba restart
```

Now when connecting to the shared directories or printers you should be prompted for a username and password.

If you choose to map a network drive to the share you can check the “Reconnect at Logon” check box, which will require you to only enter the username and password once, at least until the password changes.

Share Security

There are several options available to increase the security for each individual shared directory. Using the *[share]* example, this section will cover some common options.

Groups

Groups define a collection of computers or users which have a common level of access to particular network resources and offer a level of granularity in controlling access to such resources. For example, if a group qa is defined and contains the users freda, danika, and rob and a second group support is defined and consists of users danika, jeremy, and vincent then certain network resources configured to allow access by the qa group will subsequently enable access by freda, danika, and rob, but not jeremy or vincent. Since the user danika belongs to both the qa and support groups, she will be able to access resources configured for access by both groups, whereas all other users will have only access to resources explicitly allowing the group they are part of.

By default Samba looks for the local system groups defined in /etc/group to determine which users belong to which groups.

When defining groups in the Samba configuration file, /etc/samba/smb.conf, the recognized syntax is to preface the group name with an "@" symbol. For example, if you wished to define a group named sysadmin in a certain section of the /etc/samba/smb.conf, you would do so by entering the group name as **@sysadmin**.

File Permissions

File Permissions define the explicit rights a computer or user has to a particular directory, file, or set of files. Such permissions may be defined by editing the /etc/samba/smb.conf file and specifying the explicit permissions of a defined file share.

For example, if you have defined a Samba share called *share* and wish to give read-only permissions to the group of users known as qa, but wanted to allow writing to the share by the group called sysadmin and the user named vincent, then you could edit the /etc/samba/smb.conf file, and add the following entries under the *[share]* entry:

```
read list = @qa
write list = @sysadmin, vincent
```

Another possible Samba permission is to declare *administrative* permissions to a particular shared resource. Users having administrative permissions may read, write, or modify any

information contained in the resource the user has been given explicit administrative permissions to.

For example, if you wanted to give the user melissa administrative permissions to the share example, you would edit the /etc/samba/smb.conf file, and add the following line under the *[share]* entry:

```
admin users = melissa
```

After editing /etc/samba/smb.conf, restart Samba for the changes to take effect:

```
$ sudo /etc/init.d/samba restart
```

For the *read list* and *write list* to work the Samba security mode must *not* be set to security = share

Now that Samba has been configured to limit which groups have access to the shared directory, the filesystem permissions need to be updated.

Traditional Linux file permissions do not map well to Windows NT Access Control Lists (ACLs). Fortunately POSIX ACLs are available on Ubuntu servers providing more fine grained control. For example, to enable ACLs on /srv an EXT3 filesystem, edit /etc/fstab adding the *acl* option:

```
UUID=66bcdd2e-8861-4fb0-b7e4-e61c569fe17d /srv  ext3
noatime,relatime,acl 0       1
```

Then remount the partition:

```
$ sudo mount -v -o remount /srv
```

The above example assumes /srv on a separate partition. If /srv, or wherever you have configured your share path, is part of the / partition a reboot may be required.

To match the Samba configuration above the *sysadmin* group will be given read, write, and execute permissions to /srv/samba/share, the *qa* group will be given read and execute permissions, and the files will be owned by the username *melissa*. Enter the following in a terminal:

```
$ sudo chown -R melissa /srv/samba/share/
$ sudo chgrp -R sysadmin /srv/samba/share/
$ sudo setfacl -R -m g:qa:rx /srv/samba/share/
```

The **setfacl** command above gives *execute* permissions to all files in the /srv/samba/share directory, which you may or may not want.

Now from a Windows client you should notice the new file permissions are implemented. See the **acl** and **setfacl** man pages for more information on POSIX ACLs.

Samba AppArmor Profile

Ubuntu comes with the **AppArmor** security module, which provides mandatory access controls. The default AppArmor profile for Samba will need to be adapted to your configuration.

There are default AppArmor profiles for `/usr/sbin/smbd` and `/usr/sbin/nmbd`, the Samba daemon binaries, as part of the **apparmor-profiles** packages. To install the package, from a terminal prompt enter:

```
$ sudo apt-get install apparmor-profiles
```

By default the profiles for **smbd** and **nmbd** are in *complain* mode allowing Samba to work without modifying the profile, and only logging errors. To place the **smbd** profile into *enforce* mode, and have Samba work as expected, the profile will need to be modified to reflect any directories that are shared.

Edit `/etc/apparmor.d/usr.sbin.smbd` adding information for *[share]* from the file server example:

```
/srv/samba/share/ r,
/srv/samba/share/** rwkix,
```

Now place the profile into *enforce* and reload it:

```
$ sudo aa-enforce /usr/sbin/smbd
$ cat /etc/apparmor.d/usr.sbin.smbd | sudo apparmor_parser
-r
```

You should now be able to read, write, and execute files in the shared directory as normal, and the **smbd** binary will have access to only the configured files and direcotories. Be sure to add entries for each directory you configure Samba to share. Also, any errors will be logged to /var/log/syslog.

Chapter 7: Network File System (NFS)

NFS allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files.

Some of the most notable benefits that NFS can provide are:

- Local workstations use less disk space because commonly used data can be stored on a single machine and still remain accessible to others over the network.
- There is no need for users to have separate home directories on every network machine. Home directories could be set up on the NFS server and made available throughout the network.
- Storage devices such as floppy disks, CDROM drives, and USB Thumb drives can be used by other machines on the network. This may reduce the number of removable media drives throughout the network.

7.1 Installation

At a terminal prompt enter the following command to install the NFS Server:

```
$ sudo apt-get install nfs-kernel-server
```

7.2 Configuration

You can configure the directories to be exported by adding them to the /etc/exports file. For example:

```
/ubuntu  *(ro,sync,no_root_squash)
/home    *(rw,sync,no_root_squash)
```

You can replace * with one of the hostname formats. Make the hostname declaration as specific as possible so unwanted systems cannot access the NFS mount.

To start the NFS server, you can run the following command at a terminal prompt:

```
$ sudo /etc/init.d/nfs-kernel-server start
```

7.3 NFS Client Configuration

Use the **mount** command to mount a shared NFS directory from another machine, by typing a command line similar to the following at a terminal prompt:

```
$ sudo mount example.hostname.com:/ubuntu /local/Ubuntu
```

The mount point directory /local/ubuntu must exist. There should be no files or subdirectories in the /local/ubuntu directory.

An alternate way to mount an NFS share from another machine is to add a line to the /etc/fstab file. The line must state the hostname of the NFS server, the directory on the server being exported, and the directory on the local machine where the NFS share is to be mounted.

The general syntax for the line in /etc/fstab file is as follows:

```
example.hostname.com:/ubuntu /local/ubuntu nfs
rsize=8192,wsize=8192,timeo=14,intr
```

If you have trouble mounting an NFS share, make sure the **nfs-common** package is installed on your client. To install **nfs-common** enter the following command at the terminal prompt:

```
$ sudo apt-get install nfs-common
```

Chapter 8: FTP Server

File Transfer Protocol (FTP) is a TCP protocol for uploading and downloading files between computers. FTP works on a client/server model. The server component is called an *FTP daemon*. It continuously listens for FTP requests from remote clients. When a request is received, it manages the login and sets up the connection. For the duration of the session it executes any of commands sent by the FTP client.

Access to an FTP server can be managed in two ways:

- Anonymous
- Authenticated

In the Anonymous mode, remote clients can access the FTP server by using the default user account called "anonymous" or "ftp" and sending an email address as the password. In the Authenticated mode a user must have an account and a password. User access to the FTP server directories and files is dependent on the permissions defined for the account used at login. As a general rule, the FTP daemon will hide the root directory of the FTP server and change it to the FTP Home directory. This hides the rest of the file system from remote sessions.

8.1 vsftpd - FTP Server Installation

vsftpd is an FTP daemon available in Ubuntu. It is easy to install, set up, and maintain. To install **vsftpd** you can run the following command:

```
$ sudo apt-get install vsftpd
```

8.2 Anonymous FTP Configuration

By default **vsftpd** is configured to only allow anonymous download. During installation a *ftp* user is created with a home directory of /home/ftp. This is the default FTP directory.

If you wish to change this location, to /srv/ftp for example, simply create a directory in another location and change the *ftp* user's home directory:

```
$ sudo mkdir /srv/ftp
$ sudo usermod -d /srv/ftp ftp
```

After making the change restart **vsftpd**:

```
$ sudo /etc/init.d/vsftpd restart
```

Finally, copy any files and directories you would like to make available through anonymous FTP to /srv/ftp.

8.3 User Authenticated FTP Configuration

To configure **vsftpd** to authenticate system users and allow them to upload files edit /etc/vsftpd.conf:

```
local_enable=YES
write_enable=YES
```

Now restart **vsftpd**:

```
$ sudo /etc/init.d/vsftpd restart
```

Now when system users login to FTP they will start in their *home* directories where they can download, upload, create directories, etc.

Similarly, by default, the anonymous users are not allowed to upload files to FTP server. To change this setting, you should uncomment the following line, and restart **vsftpd**:

```
anon_upload_enable=YES
```

Enabling anonymous FTP upload can be an extreme security risk. It is best to not enable anonymous upload on servers accessed directly from the Internet.

The configuration file consists of many configuration parameters. The information about each parameter is available in the configuration file. Alternatively, you can refer to the man page, **man 5 vsftpd.conf** for details of each parameter.

8.4 Securing FTP

There are options in /etc/vsftpd.conf to help make **vsftpd** more secure. For example users can be limited to their home directories by uncommenting:

```
chroot_local_user=YES
```

You can also limit a specific list of users to just their home directories:

```
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd.chroot_list
```

After uncommenting the above options, create a /etc/vsftpd.chroot_list containing a list of users one per line. Then restart **vsftpd**:

```
$ sudo /etc/init.d/vsftpd restart
```

Also, the /etc/ftpusers file is a list of users that are *disallowed* FTP access. The default list includes root, daemon, nobody, etc. To disable FTP access for additional users simply add them to the list.

FTP can also be encrypted using *FTPS*. Different from *SFTP*, *FTPS* is FTP over Secure Socket Layer (SSL). *SFTP* is a FTP like session over an encrypted *SSH* connection. A major difference is that users of SFTP need to have a *shell* account on the system, instead of a *nologin* shell. Providing all users with a shell may not be ideal for some environments, such as a shared web host.

To configure *FTPS*, edit /etc/vsftpd.conf and at the bottom add:

```
ssl_enable=Yes
```

Also, notice the certificate and key related options:

```
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-
snakeoil.key
```

By default these options are set the the certificate and key provided by the **ssl-cert** package. In a production environment these should be replaced with a certificate and key generated for the specific host.

Now restart **vsftpd**, and non-anonymous users will be forced to use *FTPS*:

```
$ sudo /etc/init.d/vsftpd restart
```

To allow users with a shell of /usr/sbin/nologin access to FTP, but have no shell access, edit /etc/shells adding the *nologin* shell:

```
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
```

```
/usr/sbin/nologin
```

This is necessary because, by default **vsftpd** uses PAM for authentication, and the /etc/pam.d/vsftpd configuration file contains:

```
auth     required        pam_shells.so
```

The *shells* PAM module restricts access to shells listed in the /etc/shells file.

Most popular FTP clients can be configured connect using FTPS. The **lftp** command line FTP client has the ability to use FTPS as well.

Chapter 9: Dynamic Host Configuration Protocol (DHCP)

The Dynamic Host Configuration Protocol (DHCP) is a network service that enables host computers to be automatically assigned settings from a server as opposed to manually configuring each network host. Computers configured to be DHCP clients have no control over the settings they receive from the DHCP server, and the configuration is transparent to the computer's user.

The most common settings provided by a DHCP server to DHCP clients include:

- IP-Address and Netmask
- DNS
- WINS

However, a DHCP server can also supply configuration properties such as:

- Host Name
- Domain Name
- Default Gateway
- Time Server
- Print Server

The advantage of using DHCP is that changes to the network, for example a change in the address of the DNS server, need only be changed at the DHCP server, and all network hosts will be reconfigured the next time their DHCP clients poll the DHCP server. As an added advantage, it is also easier to integrate new computers into the network, as there is no need to check for the availability of an IP address. Conflicts in IP address allocation are also reduced.

A DHCP server can provide configuration settings using two methods:

MAC Address

This method entails using DHCP to identify the unique hardware address of each network card connected to the network and then continually supplying a constant configuration each time the DHCP client makes a request to the DHCP server using that network device.

Address Pool

This method entails defining a pool (sometimes also called a range or scope) of IP addresses from which DHCP clients are supplied their configuration properties dynamically and on a "first come, first served" basis. When a DHCP client is no longer on

the network for a specified period, the configuration is expired and released back to the address pool for use by other DHCP Clients.

Ubuntu is shipped with both DHCP server and client. The server is **dhcpd** (dynamic host configuration protocol daemon). The client provided with Ubuntu is **dhclient** and should be installed on all computers required to be automatically configured. Both programs are easy to install and configure and will be automatically started at system boot.

9.1 Installation

At a terminal prompt, enter the following command to install **dhcpd**:

```
$ sudo apt-get install dhcp3-server
```

You will probably need to change the default configuration by editing /etc/dhcp3/dhcpd.conf to suit your needs and particular configuration.

You also need to edit /etc/default/dhcp3-server to specify the interfaces dhcpd should listen to. By default it listens to eth0.

NOTE: dhcpd's messages are being sent to syslog. Look there for diagnostics messages.

9.2 Configuration

The error message the installation ends with might be a little confusing, but the following steps will help you configure the service:

Most commonly, what you want to do is assign an IP address randomly. This can be done with settings as follows:

```
# Sample /etc/dhcpd.conf
# (add your comments here)
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-name "mydomain.example";

subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.10 192.168.1.100;
range 192.168.1.150 192.168.1.200;
}
```

This will result in the DHCP server giving a client an IP address from the range 192.168.1.10-192.168.1.100 or 192.168.1.150-192.168.1.200. It will lease an IP address for 600 seconds if the client doesn't ask for a specific time frame. Otherwise the maximum (allowed) lease will be 7200 seconds. The server will also "advise" the client that it should use 255.255.255.0 as its subnet mask, 192.168.1.255 as its broadcast address, 192.168.1.254 as the router/gateway and 192.168.1.1 and 192.168.1.2 as its DNS servers.

If you need to specify a WINS server for your Windows clients, you will need to include the netbios-name-servers option, e.g.

```
option netbios-name-servers 192.168.1.1;
```

Chapter 10: Squid - Proxy Server

Squid is a full-featured web proxy cache server application which provides proxy and cache services for Hyper Text Transport Protocol (HTTP), File Transfer Protocol (FTP), and other popular network protocols. Squid can implement caching and proxying of Secure Sockets Layer (SSL) requests and caching of Domain Name Server (DNS) lookups, and perform transparent caching. Squid also supports a wide variety of caching protocols, such as Internet Cache Protocol, (ICP) the Hyper Text Caching Protocol, (HTCP) the Cache Array Routing Protocol (CARP), and the Web Cache Coordination Protocol. (WCCP)

The Squid proxy cache server is an excellent solution to a variety of proxy and caching server needs, and scales from the branch office to enterprise level networks while providing extensive, granular access control mechanisms and monitoring of critical parameters via the Simple Network Management Protocol (SNMP). When selecting a computer system for use as a dedicated Squid proxy, or caching servers, ensure your system is configured with a large amount of physical memory, as Squid maintains an in-memory cache for increased performance.

10.1 Installation

At a terminal prompt, enter the following command to install the Squid server:

```
$ sudo apt-get install squid
```

10.2 Configuration

Squid is configured by editing the directives contained within the /etc/squid/squid.conf configuration file. The following examples illustrate some of the directives which may be modified to affect the behavior of the Squid server. For more in-depth configuration of Squid, see the References section.

Prior to editing the configuration file, you should make a copy of the original file and protect it from writing so you will have the original settings as a reference, and to re-use as necessary.

Copy the /etc/squid/squid.conf file and protect it from writing with the following commands entered at a terminal prompt:

```
$ sudo cp /etc/squid/squid.conf
/etc/squid/squid.conf.original
$ sudo chmod a-w /etc/squid/squid.conf.original
```

- To set your Squid server to listen on TCP port 8888 instead of the default TCP port 3128, change the http_port directive as such:
- http_port 8888
- Change the visible_hostname directive in order to give the Squid server a specific hostname. This hostname does not necessarily need to be the computer's hostname. In this example it is set to *weezie*
- visible_hostname weezie
- Again, Using Squid's access control, you may configure use of Internet services proxied by Squid to be available only users with certain Internet Protocol (IP) addresses. For example, we will illustrate access by users of the 192.168.42.0/24 subnetwork only:

  Add the following to the **bottom** of the ACL section of your /etc/squid/squid.conf file:

  ```
  acl fortytwo_network src 192.168.42.0/24
  ```

  Then, add the following to the **top** of the http_access section of your /etc/squid/squid.conf file:

  ```
  http_access allow fortytwo_network
  ```

- Using the excellent access control features of Squid, you may configure use of Internet services proxied by Squid to be available only during normal business hours. For example, we'll illustrate access by employees of a business which is operating between 9:00AM and 5:00PM, Monday through Friday, and which uses the 10.1.42.0/42 subnetwork:

  Add the following to the **bottom** of the ACL section of your /etc/squid/squid.conf file:

  ```
  acl biz_network src 10.1.42.0/24
  acl biz_hours time M T W T F 9:00-17:00
  ```

  Then, add the following to the **top** of the http_access section of your /etc/squid/squid.conf file:

  ```
  http_access allow biz_network biz_hours
  ```

  After making changes to the /etc/squid/squid.conf file, save the file and restart the **squid** server application to effect the changes using the following command entered at a terminal prompt:

  ```
  $ sudo /etc/init.d/squid restart
  ```

Chapter 11: DNS

11.1 Installation

At a terminal prompt, enter the following command to install **dns**:

```
$ sudo apt-get install bind9
```

A very useful package for testing and troubleshooting DNS issues is the dnsutils package. To install **dnsutils** enter the following:

```
$ sudo apt-get install dnsutils
```

11.2 Configuration

There a many ways to configure **BIND9**. Some of the most common configurations are a caching nameserver, primary master, and a as a secondary master.

- When configured as a caching nameserver BIND9 will find the answer to name queries and remember the answer when the domain is queried again.
- As a primary master server BIND9 reads the data for a zone from a file on it's host and is authoritative for that zone.
- In a secondary master configuration BIND9 gets the zone data from another nameserver authoritative for the zone.

11.3 Overview

The DNS configuration files are stored in the /etc/bind directory. The primary configuration file is /etc/bind/named.conf.

The *include* line specifies the filename which contains the DNS options. The *directory* line in the /etc/bind/named.conf.options file tells DNS where to look for files. All files BIND uses will be relative to this directory.

The file named /etc/bind/db.root describes the root nameservers in the world. The servers change over time, so the /etc/bind/db.root file must be maintained now and then. This is usually done as updates to the **bind9** package. The *zone* section defines a master server, and it is stored in a file mentioned in the *file* option.

It is possible to configure the same server to be a caching name server, primary master, and secondary master. A server can be the Start of Authority (SOA) for one zone, while providing secondary service for another zone. All the while providing caching services for hosts on the local LAN.

Caching Nameserver

The default configuration is setup to act as a caching server. All that is required is simply adding the IP Addresses of your ISP's DNS servers. Simply uncomment and edit the following in /etc/bind/named.conf.options:

```
forwarders {
                1.2.3.4;
                5.6.7.8;
            };
```

Replace *1.2.3.4* and *5.6.7.8* with the IP Adresses of actual nameservers.

Now restart the DNS server, to enable the new configuration. From a terminal prompt:

sudo /etc/init.d/bind9 restart

Primary Master

In this section **BIND9** will be configured as the Primary Master for the domain *example.com*. Simply replace example.com with your FQDN (Fully Qualified Domain Name).

Forward Zone File

To add a DNS zone to BIND9, turning BIND9 into a Primary Master server, the first step is to edit /etc/bind/named.conf.local:

```
zone "example.com" {
       type master;
        file "/etc/bind/db.example.com";
};
```

Now use an existing zone file as a template to create the /etc/bind/db.example.com file:

```
$ sudo cp /etc/bind/db.local /etc/bind/db.example.com
```

Edit the new zone file /etc/bind/db.example.com change *localhost.* to the FQDN of your server, leaving the additional "." at the end. Change *127.0.0.1* to the nameserver's IP Address and *root.localhost* to a valid email address, but with a "." instead of the usual "@" symbol, again leaving the "." at the end.

Also, create an *A record* for ns.example.com. The name server in this example:

;

```
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     ns.example.com. root.example.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.example.com.
@       IN      A       127.0.0.1
@       IN      AAAA    ::1
ns      IN      A       192.168.1.10
```

You must increment the *Serial Number* every time you make changes to the zone file. If you make multiple changes before restarting BIND9, simply increment the Serial once.

Now, you can add DNS records to the bottom of the zone file

Many admins like to use the last date edited as the serial of a zone, such as *2007010100* which is yyyymmddss (where *ss* is the Serial Number)

Once you have made a change to the zone file **BIND9** will need to be restarted for the changes to take effect:

```
$ sudo /etc/init.d/bind9 restart
```

Reverse Zone File

Now that the zone is setup and resolving names to IP Adresses a *Reverse zone* is also required. A Reverse zone allows DNS to resolve an address to a name.

Edit /etc/bind/named.conf.local and add the following:

```
zone "1.168.192.in-addr.arpa" {
        type master;
        notify no;
        file "/etc/bind/db.192";
};
```

Replace *1.168.192* with the first three octets of whatever network you are using. Also, name the zone file /etc/bind/db.192 appropriately. It should match the first octet of your network.

Now create the /etc/bind/db.192 file:

```
$ sudo cp /etc/bind/db.127 /etc/bind/db.192
```

Next edit /etc/bind/db.192 changing the basically the same options as /etc/bind/db.example.com:

```
;
; BIND reverse data file for local loopback interface
;
$TTL    604800
@       IN      SOA     ns.example.com. root.example.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.
10      IN      PTR     ns.example.com.
```

The *Serial Number* in the Reverse zone needs to be incremented on each changes as well. For each *A record* you configure in /etc/bind/db.example.com you need to create a *PTR record* in /etc/bind/db.192.

After creating the reverse zone file restart **BIND9**:

```
$ sudo /etc/init.d/bind9 restart
```

Secondary Master

Once a *Primary Master* has been configured a *Secondary Master* is needed in order to maintain the availability of the domain should the Primary become unavailable.

First, on the Primary Master server, the zone transfer needs to be allowed. Add the *allow-transfer* option to the example Forward and Reverse zone definitions in /etc/bind/named.conf.local:

```
zone "example.com" {
        type master;
 file "/etc/bind/db.example.com";
        allow-transfer { 192.168.1.11; };
};

zone "1.168.192.in-addr.arpa" {
        type master;
        notify no;
        file "/etc/bind/db.192";
 allow-transfer { 192.168.1.11; };
 };
```

Replace *192.168.1.11* with the IP Address of your Secondary nameserver

Next, on the Secondary Master, install the **bind9** package the same way as on the Primary. Then edit the /etc/bind/named.conf.local and add the following declarations for the Forward and Reverse zones:

```
zone "example.com" {
       type slave;
        file "/var/cache/bind/db.example.com";
        masters { 192.168.1.10; };
};

zone "1.168.192.in-addr.arpa" {
       type slave;
        file "/var/cache/bind/db.192";
        masters { 192.168.1.10; };
};
```

Replace *192.168.1.10* with the IP Address of your Primary nameserver

Restart **BIND9** on the Secondary Master:

```
$ sudo /etc/init.d/bind9 restart
```

In /var/log/syslog you should see something similar to:

```
slave zone "example.com" (IN) loaded (serial 6)
slave zone "100.18.172.in-addr.arpa" (IN) loaded (serial
3)
```

Note: A zone is only transferred if the *Serial Number* on the Primary is larger than the one on the Secondary.
The default directory for non-authoritative zone files is /var/cache/bind/.

Chapter 12: HTTPD - Apache2 Web Server

Apache is the most commonly used Web Server on Linux systems. Web Servers are used to serve Web Pages requested by client computers. Clients typically request and view Web Pages using Web Browser applications such as **Firefox**, **Opera**, or **Mozilla**.

The most common protocol used to transfer Web pages is the Hyper Text Transfer Protocol (HTTP). Protocols such as Hyper Text Transfer Protocol over Secure Sockets Layer (HTTPS), and File Transfer Protocol (FTP), a protocol for uploading and downloading files, are also supported.

Apache Web Servers are often used in combination with the **MySQL** database engine, the HyperText Preprocessor (**PHP**) scripting language, and other popular scripting languages such as **Python** and **Perl**. This configuration is termed LAMP (Linux, Apache, MySQL and Perl/Python/PHP) and forms a powerful and robust platform for the development and deployment of Web-based applications.

12.1 Installation

The Apache2 web server is available in Ubuntu Linux. To install Apache2:

- At a terminal prompt enter the following command:

```
$ sudo apt-get install apache2
```

12.2 Configuration

Apache2 is configured by placing *directives* in plain text configuration files. The configuration files are separated between the following files and directories:

- *apache2.conf:* the main Apache2 configuration file. Contains settings that are *global* to Apache2.
- *conf.d:* contains configuration files which apply *globally* to Apache. Other packages that use Apache2 to serve content may add files, or symlinks, to this directory.
- *envvars:* file where Apache2 *environment* variables are set.
- *httpd.conf:* historically the main Apache2 configuration file, named after the **httpd** daemon. The file can be used for *user specific* configuration options that globally effect Apache2.
- *mods-available:* this directory contains configuration files to both load *modules* and configure them. Not all modules will have specific configuration files, however.

- *mods-enabled:* holds *symlinks* to the files in /etc/apache2/mods-available. When a module configuration file is symlinked it will be enabled the next time **apache2** is restarted.
- *ports.conf:* houses the directives that determine which TCP ports Apache2 is listening on.
- *sites-available:* this directory has configuration files for Apache *Virtual Hosts*. Virtual Hosts allow Apache2 to be configured for multiple sites that have separate configurations.
- *sites-enabled:* like mods-enabled, sites-enabled contains symlinks to the /etc/apache2/sites-available directory. Similarly when a configuration file in sites-available is symlinked it will be active once Apache is restarted.

In addition, other configuration files may be added using the *Include* directive, and wildcards can be used to include many configuration files. Any directive may be placed in any of these configuration files. Changes to the main configuration files are only recognized by Apache2 when it is started or restarted.

The server also reads a file containing mime document types; the filename is set by the *TypesConfig* directive, and is /etc/mime.types by default.

12.3 Basic Settings

This section explains Apache2 server essential configuration parameters.

- Apache2 ships with a virtual-host-friendly default configuration. That is, it is configured with a single default virtual host (using the *VirtualHost* directive) which can modified or used as-is if you have a single site, or used as a template for additional virtual hosts if you have multiple sites. If left alone, the default virtual host will serve as your default site, or the site users will see if the URL they enter does not match the *ServerName* directive of any of your custom sites. To modify the default virtual host, edit the file /etc/apache2/sites-available/default.

  The directives set for a virtual host only apply to that particular virtual host. If a directive is set server-wide and not defined within the virtual host settings, the default setting is used. For example, you can define a Webmaster email address and not define individual email addresses for each virtual host.

- If you wish to configure a new virtual host or site, copy that file into the same directory with a name you choose. For example:

  ```
  $ sudo cp /etc/apache2/sites-available/default
  /etc/apache2/sites-available/mynewsite
  ```

- Edit the new file to configure the new site using some of the directives described below.

- The *ServerAdmin* directive specifies the email address to be advertised for the server's administrator. The default value is webmaster@localhost. This should be changed to an email address that is delivered to you (if you are the server's administrator). If your website has a problem, Apache2 will display an error message containing this email address to report the problem to. Find this directive in your site's configuration file in /etc/apache2/sites-available.
- The *Listen* directive specifies the port, and optionally the IP address, Apache2 should listen on. If the IP address is not specified, Apache2 will listen on all IP addresses assigned to the machine it runs on. The default value for the Listen directive is 80. Change this to 127.0.0.1:80 to cause Apache2 to listen only on your loopback interface so that it will not be available to the Internet, to (for example) 81 to change the port that it listens on, or leave it as is for normal operation. This directive can be found and changed in its own file, /etc/apache2/ports.conf
- The *ServerName* directive is optional and specifies what FQDN your site should answer to. The default virtual host has no ServerName directive specified, so it will respond to all requests that do not match a ServerName directive in another virtual host. If you have just acquired the domain name ubunturocks.com and wish to host it on your Ubuntu server, the value of the ServerName directive in your virtual host configuration file should be ubunturocks.com. Add this directive to the new virtual host file you created earlier (/etc/apache2/sites-available/mynewsite).

  You may also want your site to respond to www.ubunturocks.com, since many users will assume the www prefix is appropriate. Use the *ServerAlias* directive for this. You may also use wildcards in the ServerAlias directive.

  For example, the following configuration will cause your site to respond to any domain request ending in *.ubunturocks.com*.

  ServerAlias *.ubunturocks.com

- The *DocumentRoot* directive specifies where Apache should look for the files that make up the site. The default value is /var/www. No site is configured there, but if you uncomment the *RedirectMatch* directive in /etc/apache2/apache2.conf requests will be redirected to /var/www/apache2-default where the default Apache2 site awaits. Change this value in your site's virtual host file, and remember to create that directory if necessary!

The /etc/apache2/sites-available directory is **not** parsed by Apache2. Symbolic links in /etc/apache2/sites-enabled point to "available" sites.

Enable the new *VirtualHost* using the **a2ensite** utility and restart Apache:

```
$ sudo a2ensite mynewsite
$ sudo /etc/init.d/apache2 restart
```

Be sure to replace *mynewsite* with a more descriptive name for the VirtualHost. One method is to name the file after the *ServerName* directive of the VirtualHost.

Similarly, use the **a2dissite** utility to disable sites. This is can be useful when troubleshooting configuration problems with multiple VirtualHosts:

```
$ sudo a2dissite mynewsite
$ sudo /etc/init.d/apache2 restart
```

12.4 Default Settings

This section explains configuration of the Apache2 server default settings. For example, if you add a virtual host, the settings you configure for the virtual host take precedence for that virtual host. For a directive not defined within the virtual host settings, the default value is used.

- The *DirectoryIndex* is the default page served by the server when a user requests an index of a directory by specifying a forward slash (/) at the end of the directory name.

  For example, when a user requests the page http://www.example.com/this_directory/, he or she will get either the DirectoryIndex page if it exists, a server-generated directory list if it does not and the Indexes option is specified, or a Permission Denied page if neither is true. The server will try to find one of the files listed in the DirectoryIndex directive and will return the first one it finds. If it does not find any of these files and if Options Indexes is set for that directory, the server will generate and return a list, in HTML format, of the subdirectories and files in the directory. The default value, found in /etc/apache2/apache2.conf is " index.html index.cgi index.pl index.php index.xhtml". Thus, if Apache2 finds a file in a requested directory matching any of these names, the first will be displayed.

- The *ErrorDocument* directive allows you to specify a file for Apache to use for specific error events. For example, if a user requests a resource that does not exist, a 404 error will occur, and per Apache2's default configuration, the file /usr/share/apache2/error/HTTP_NOT_FOUND.html.var will be displayed. That file is not in the server's DocumentRoot, but there is an Alias directive in /etc/apache2/apache2.conf that redirects requests to the /error directory to /usr/share/apache2/error/.

  To see a list of the default ErrorDocument directives, use this command:

  ```
  $ grep ErrorDocument /etc/apache2/apache2.conf
  ```

- By default, the server writes the transfer log to the file /var/log/apache2/access.log. You can change this on a per-site basis in your virtual host configuration files with the *CustomLog* directive, or omit it to accept the default, specified in /etc/apache2/apache2.conf. You may also specify the file to which errors are logged, via the *ErrorLog* directive, whose default is /var/log/apache2/error.log. These are kept separate from the transfer logs to aid in troubleshooting problems with your Apache2 server. You may also specify the *LogLevel* (the default value is "warn") and the *LogFormat* (see /etc/apache2/apache2.conf for the default value).

- Some options are specified on a per-directory basis rather than per-server. *Options* is one of these directives. A Directory stanza is enclosed in XML-like tags, like so:

```
<Directory /var/www/mynewsite>
...
</Directory>
```

The *Options* directive within a Directory stanza accepts one or more of the following values (among others), separated by spaces:

- **ExecCGI** - Allow execution of CGI scripts. CGI scripts are not executed if this option is not chosen.

  Most files should not be executed as CGI scripts. This would be very dangerous. CGI scripts should kept in a directory separate from and outside your DocumentRoot, and only this directory should have the ExecCGI option set. This is the default, and the default location for CGI scripts is /usr/lib/cgi-bin.

- **Includes** - Allow server-side includes. Server-side includes allow an HTML file to *include* other files. This is not a common option.
- **IncludesNOEXEC** - Allow server-side includes, but disable the *#exec* and *#include* commands in CGI scripts.
- **Indexes** - Display a formatted list of the directory's contents, if no *DirectoryIndex* (such as index.html) exists in the requested directory.

  For security reasons, this should usually not be set, and certainly should not be set on your DocumentRoot directory. Enable this option carefully on a per-directory basis only if you are certain you want users to see the entire contents of the directory.

- **Multiview** - Support content-negotiated multiviews; this option is disabled by default for security reasons.
- **SymLinksIfOwnerMatch** - Only follow symbolic links if the target file or directory has the same owner as the link.

12.5 httpd Settings

This section explains some basic **httpd** daemon configuration settings.

LockFile - The LockFile directive sets the path to the lockfile used when the server is compiled with either USE_FCNTL_SERIALIZED_ACCEPT or USE_FLOCK_SERIALIZED_ACCEPT. It must be stored on the local disk. It should be left to the default value unless the logs directory is located on an NFS share. If this is the case, the default value should be changed to a location on the local disk and to a directory that is readable only by root.

PidFile - The PidFile directive sets the file in which the server records its process ID (pid). This file should only be readable by root. In most cases, it should be left to the default value.

User - The User directive sets the userid used by the server to answer requests. This setting determines the server's access. Any files inaccessible to this user will also be inaccessible to your website's visitors. The default value for User is www-data.

Unless you know exactly what you are doing, do not set the User directive to root. Using root as the User will create large security holes for your Web server.

The Group directive is similar to the User directive. Group sets the group under which the server will answer requests. The default group is also www-data.

Apache Modules

Apache is a modular server. This implies that only the most basic functionality is included in the core server. Extended features are available through modules which can be loaded into Apache. By default, a base set of modules is included in the server at compile-time. If the server is compiled to use dynamically loaded modules, then modules can be compiled separately, and added at any time using the LoadModule directive. Otherwise, Apache must be recompiled to add or remove modules.

Ubuntu compiles Apache2 to allow the dynamic loading of modules. Configuration directives may be conditionally included on the presence of a particular module by enclosing them in an *<IfModule>* block.

You can install additional Apache2 modules and use them with your Web server. For example, run the following command from a terminal prompt to install the *MySQL Authentication* module:

```
$ sudo apt-get install libapache2-mod-auth-mysql
```

See the /etc/apache2/mods-available directory, for additional modules.

Use the **a2enmod** utility to enable a module:

```
$ sudo a2enmod auth_mysql
$ sudo /etc/init.d/apache2 restart
```

Similarly, **a2dismod** will disable a module:

```
$ sudo a2dismod auth_mysql
$ sudo /etc/init.d/apache2 restart
```

HTTPS Configuration

The **mod_ssl** module adds an important feature to the Apache2 server - the ability to encrypt communications. Thus, when your browser is communicating using SSL, the https:// prefix is used at the beginning of the Uniform Resource Locator (URL) in the browser navigation bar.

The **mod_ssl** module is available in **apache2-common** package. Execute the following command from a terminal prompt to enable the **mod_ssl** module:

```
$ sudo a2enmod ssl
```

There is a default HTTPS configuration file in /etc/apache2/sites-available/default-ssl. In order for **Apache** to provide HTTPS, a *certificate* and *key* file are also needed. The default HTTPS configuration will use a certificate and key generated by the **ssl-cert** package. They are good for testing, but the auto-generated certificate and key should be replaced by a certificate specific to the site or server.

To configure **Apache** for HTTPS, enter the following:

```
$ sudo a2ensite default-ssl
```

The directories /etc/ssl/certs and /etc/ssl/private are the default locations. If you install the certificate and key in another directory make sure to change *SSLCertificateFile* and *SSLCertificateKeyFile* appropriately.

With Apache now configured for HTTPS, restart the service to enable the new settings:

```
$ sudo /etc/init.d/apache2 restart
```

Depending on how you obtained your certificate you may need to enter a passphrase when **Apache** starts.

You can access the secure server pages by typing https://your_hostname/url/ in your browser address bar.

Chapter 13: MySQL

MySQL is a fast, multi-threaded, multi-user, and robust SQL database server. It is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software.

13.1 Installation

To install MySQL, run the following command from a terminal prompt:

```
$ sudo apt-get install mysql-server
```

During the installation process you will be prompted to enter a password for the **MySQL** root user.

Once the installation is complete, the MySQL server should be started automatically. You can run the following command from a terminal prompt to check whether the MySQL server is running:

```
$ sudo netstat -tap | grep mysql
```

When you run this command, you should see the following line or something similar:

```
tcp    0   0 localhost:mysql      *:*      LISTEN      2556/mysqld
```

If the server is not running correctly, you can type the following command to start it:

```
$ sudo /etc/init.d/mysql restart
```

13.2 Configuration

You can edit the /etc/mysql/my.cnf file to configure the basic settings -- log file, port number, etc. For example, to configure **MySQL** to listen for connections from network hosts, change the *bind_address* directive to the server's IP address:

```
bind-address            = 192.168.0.5
```

Replace 192.168.0.5 with the appropriate address.

After making a change to /etc/mysql/my.cnf the **mysql** daemon will need to be restarted:

```
$ sudo /etc/init.d/mysql restart
```

If you would like to change the **MySQL** *root* password, in a terminal enter:

```
$ sudo dpkg-reconfigure mysql-server-5.0
```

The **mysql** daemon will be stopped, and you will be prompted to enter a new password.

Chapter 14: Postfix (Mail server)

Postfix is the default Mail Transfer Agent (MTA) in Ubuntu. It attempts to be fast and easy to administer and secure. It is compatible with the MTA **sendmail**. This section explains how to install and configure **postfix**. It also explains how to set it up as an SMTP server using a secure connection (for sending emails securely).

14.1 Installation

To install **postfix** run the following command:

```
$ sudo apt-get install postfix
```

Simply press return when the installation process asks questions, the configuration will be done in greater detail in the next stage.

14.2 Basic Configuration

To configure **postfix**, run the following command:

```
$ sudo dpkg-reconfigure postfix
```

The user interface will be displayed. On each screen, select the following values:

- Internet Site
- mail.example.com
- steve
- mail.example.com, localhost.localdomain, localhost
- No
- 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 192.168.0/24
- 0
- +
- All

Replace mail.example.com with your mail server hostname, 192.168.0/24 with the actual network and class range of your mail server, and steve with the appropriate username.

SMTP Authentication

SMTP-AUTH allows a client to identify itself through an authentication mechanism (SASL). Transport Layer Security (TLS) should be used to encrypt the authentication process. Once authenticated the SMTP server will allow the client to relay mail.

Configuring **Postfix** for SMTP-AUTH is very simple using the **dovecot-postfix** package. This package will install **Dovecot** and configure **Postfix** to use it for both SASL authentication and as a Mail Delivery Agent (MDA). The package also configures **Dovecot** for IMAP, IMAPS, POP3, and POP3S.

To install the package, from a terminal prompt enter:

```
$ sudo apt-get install dovecot-postfix
```

You should now have a working mail server, but there are a few options that you may wish to further customize. For example, the package uses the certificate and key from the **ssl-cert** package, and in a production environment you should use a certificate and key generated for the host.

Once you have a customized certificate and key for the host, change the following options in /etc/postfix/main.cf:

```
smtpd_tls_cert_file = /etc/ssl/certs/ssl-mail.pem
smtpd_tls_key_file = /etc/ssl/private/ssl-mail.key
```

Then restart Postfix:

```
$ sudo /etc/init.d/postfix restart
```

14.3 Testing

SMTP-AUTH configuration is complete. Now it is time to test the setup.

To see if SMTP-AUTH and TLS work properly, run the following command:

```
$ telnet mail.example.com 25
```

After you have established the connection to the postfix mail server, type:

```
$ ehlo mail.example.com
```

If you see the following lines among others, then everything is working perfectly. Type **quit** to exit.

```
250-STARTTLS
```

```
250-AUTH LOGIN PLAIN
250-AUTH=LOGIN PLAIN
250 8BITMIME
```

Troubleshooting

This section introduces some common ways to determine the cause if problems arise.

Escaping chroot

The Ubuntu **postfix** package will by default install into a *chroot* environment for security reasons. This can add greater complexity when troubleshooting problems.

To turn off the chroot operation locate for the following line in the /etc/postfix/master.cf configuration file:

```
smtp      inet  n       -       -       -       -       smtpd
```

and modify it as follows:

```
smtp      inet  n       -       n       -       -       smtpd
```

You will then need to restart Postfix to use the new configuration. From a terminal prompt enter:

```
$ sudo /etc/init.d/postfix restart
```

Log Files

Postfix sends all log messages to /var/log/mail.log. However error and warning messages can sometimes get lost in the normal log output so they are also logged to /var/log/mail.err and /var/log/mail.warn respectively.

To see messages entered into the logs in real time you can use the **tail -f** command:

```
$ tail -f /var/log/mail.err
```

The amount of detail that is recorded in the logs can be increased. Below are some configuration options for increasing the log level for some of the areas covered above.

- To increase *TLS* activity logging set the *smtpd_tls_loglevel* option to a value from 1 to 4.

  ```
  $ sudo postconf -e 'smtpd_tls_loglevel = 4'
  ```

- If you are having trouble sending or receiving mail from a specific domain you can add the domain to the *debug_peer_list* parameter.

```
$ sudo postconf -e 'debug_peer_list = problem.domain'
```

- You can increase the verbosity of any **Postfix** daemon process by editing the /etc/postfix/master.cf and adding a *-v* after the entry. For example edit the *smtp* entry:

```
smtp      unix  -       -       -       -      -       smtp –v
```

It is important to note that after making one of the logging changes above the **Postfix** process will need to be reloaded in order to recognize the new configuration: **sudo /etc/init.d/postfix reload**

- To increase the amount of information logged when troubleshooting *SASL* issues you can set the following options in /etc/dovecot/dovecot.conf

```
auth_debug=yes
auth_debug_passwords=yes
```

Just like **Postfix** if you change a **Dovecot** configuration the process will need to be reloaded: **sudo /etc/init.d/dovecot reload**.
Some of the options above can drastically increase the amount of information sent to the log files. Remember to return the log level back to normal after you have corrected the problem. Then reload the appropriate daemon for the new configuration to take affect.