CHAPTER 15

Remote Access with SSH and Telnet

IN THIS CHAPTER

The ability to control your system remotely is one of the high points of Ubuntu—you can connect from any Linux box to another Linux box in a variety of ways. If you just want to check something quickly or if you have limited bandwidth, you have the option of using only the command line, but you can also connect directly to the X server and get full graphical control.

Understanding the selection of tools available is largely a history lesson. For example, Telnet was an earlier way of connecting to another computer through the command line, but that has since been superseded by SSH. That is not to say that you should ignore Telnet; you need to know how to use it so you have it as a fallback. However, SSH is preferred because it is more secure. We cover both in this chapter.

Setting Up a Telnet Server

You will find the Telnet server installation packages in Synaptic under the `telnetd` package. Once installed, select Administration, Services and enable Telnet. Note your IP address while you are here (run `ifconfig with sudo`).

With that done, you can now fire up your other Linux box and type **`telnet <your IP>`**. You are prompted to enter your username and password. The whole conversation should look like this:

```
[paul@susannah ~]$ telnet 10.0.0.1
Trying 10.0.0.1…
Connected to 10.0.0.1 (10.0.0.1)
Escape character is '^]'.
```

```
Welcome to Caitlin
Running Ubuntu

* All access is logged *

login: paul
Password:
Last login: Sat Jul 9 12:05:41 from 10.0.0.5
[paul@caitlin ~]$
```

TIP

Note that the server responds with `Welcome to Caitlin, running Ubuntu`, which is a customized message. Your machine will probably respond with `Ubuntu` and some version information. This is insecure: giving away version numbers is never a smart move. In fact, even saying `Ubuntu` is questionable. Edit the `issue` and `issue.net` files in your `/etc` directory to change these messages.

Running the `w` command now shows you as connecting from the external IP address.

Telnet Versus SSH

Although Telnet is worth keeping around as a fail-safe, last resort option, SSH is superior in virtually every way. Telnet is fast but also insecure. It sends all your text, including your password, in plain text that can be read by anyone with the right tools. SSH, on the other hand, encrypts all your communication and so is more resource-intensive but secure—even a government security agency sniffing your packets for some reason would still have a hard time cracking the encryption.

Andy Green, posting to the `fedora-list` mailing list, summed up the Telnet situation perfectly when he said, "As Telnet is universally acknowledged to encourage evil, the service telnetd is not enabled by default." It is worthwhile taking the hint: Use Telnet as a last resort only.

Setting Up an SSH Server

If not installed already, the OpenSSH server can be installed through Synaptic by adding the openssh-server package. If you have disabled it, you can re-enable it by selecting System, Administration, Services and selecting the Remote Shell Server box. As you might have gathered, `sshd` is the name for the SSH server daemon.

Two different versions of SSH exist, called SSH1 and SSH2. The latter is newer, is more secure, comes with more features, and is the default in Ubuntu. Support for SSH1 clients is best left disabled so older clients can connect. This is set up in the `/etc/ssh/sshd_config` file on this line:

```
#Protocol 2,1
```

For maximum security, that line should read:

```
Protocol 2
```

This removes the comment sign (#) and tells `sshd` that you want it to only allow SSH2 connections. Save the file and exit your editor. The next step is to tell `sshd` to reread its configuration file, by executing this command:

```
kill -HUP `cat /var/run/sshd.pid`
```

If this returns `cat: /var/run/sshd.pid: No such file or directory`, it means you didn't have `sshd` running. Next time you start it, it reads the configuration file and uses SSH2 only.

You can test this change by trying to connect to your SSH server in SSH1 mode. From the same machine, type this:

```
ssh -1 localhost
```

The `-1` switch forces SSH1 mode. If you successfully forced the SSH2 protocol, you should get the message `Protocol major versions differ: 1 vs. 2`.

The SSH Tools

To the surprise of many, OpenSSH actually comprises a suite of tools. We have already seen `ssh`, the secure shell command that connects to other machines, and `sshd`, the SSH server daemon that accepts incoming SSH connections. However, there is also `sftp`, a replacement for `ftp`, and `scp`, a replacement for `rcp`.

You should already be familiar with the `ftp` command because it is the lowest-common-denominator system for handling FTP file transfers. Like Telnet, though, `ftp` is insecure: It sends your data in plain text across the network and anyone can sniff your packets to pick out a username and password. The SSH replacement, `sftp`, puts FTP traffic over an SSH link, thus securing it.

The `rcp` command might be new to you, largely because it is not used much anymore. Back in its day, `rcp` was the primary way of copying a single file to another server. As with `ftp`, `scp` replaces `rcp` by simply channeling the data over a secure SSH connection. The difference between `sftp` and `scp` is that the former allows you to copy many files, whereas the latter just sends one.

Using `scp` to Copy Individual Files Between Machines

The most basic use of the `scp` command is to copy a file from your current machine to a remote machine. You can do that with the following command:

```
scp test.txt 10.0.0.1:
```

The first parameter is the name of the file you want to send, and the second is the server to which you want to send it. Note that there is a colon at the end of the IP address. This is where you can specify an exact location for the file to be copied. If you have nothing after the colon, as in the previous example, `scp` copies the file to your home directory. As with SSH, `scp` prompts you for your password before copying takes place.

You can rewrite the previous command so you copy `test.txt` from the local machine and save it as `newtest.txt` on the server:

```
scp test.txt 10.0.0.1:newtest.txt
```

Alternatively, if there is a directory where you want the file to be saved, you can specify it like this:

```
scp test.txt 10.0.0.1:subdir/stuff/newtest.txt
```

The three commands so far have all assumed that your username on your local machine is the same as your username on the remote machine. If this is not the case, you need to specify your username before the remote address, like this:

```
scp test.txt japh@10.0.0.1:newtest.txt
```

You can use `scp` to copy remote files locally, simply by specifying the remote file as the source and the current directory (`.`) as the destination:

```
scp 10.0.0.1:remote.txt .
```

The `scp` command is nominally also capable of copying files from one remote machine to another remote machine, but this functionality has yet to be properly implemented in Ubuntu. If a patch is released—and we hope one is eventually—the correct command to use would be this:

```
scp 10.0.0.1:test.txt 10.0.0.2:remotetest.txt
```

That copies `test.txt` from `10.0.0.1` to `remotetest.txt` on `10.0.0.2`. If this works, you are asked for passwords for both servers.

Using `sftp` to Copy Many Files Between Machines

`sftp` is a mix between `ftp` and `scp`. Connecting to the server uses the same syntax as `scp`—you can just specify an IP address to connect using your current username, or you can specify a username using *`username@ipaddress`*. You can optionally add a colon and a directory, as with `scp`. After you are connected, the commands are the same as `ftp`: `cd`, `put`, `mput`, `get`, `quit`, and so on.

In one of the `scp` examples, we copied a remote file locally. You can do the same thing with `sftp` with the following conversation:

```
[paul@susannah ~]$ sftp 10.0.0.1
Connecting to 10.0.0.1...
```

```
paul@10.0.0.1's password:
sftp> get remote.txt
Fetching /home/paul/remote.txt to remote.txt
/home/paul/remote.txt      100%  23   0.0KB/s   00:00
sftp> quit
paul@susannah ~]$
```

Although FTP remains prominent because of the number of systems that do not have support for SSH (Windows, specifically), SFTP is gaining in popularity. Apart from the fact that it secures all communications between client and server, SFTP is popular because the initial connection between the client and server is made over port 22 through the `sshd` daemon. Someone using SFTP connects to the standard `sshd` daemon, verifies himself, and then is handed over to the SFTP server. The advantage to this is that it reduces the attack vectors because the SFTP server cannot be contacted directly and so cannot be attacked as long as the `sshd` daemon is secure.

Using `ssh-keygen` to Enable Key-based Logins

There is a weak link in the SSH system, and, inevitably, it lies with users. No matter what lengths system administrators go to in training users to be careful with their passwords, monitors around the world have Post-it notes attached to them with “pAssw0rd” written on. Sure, it has a mix of letters and numbers, but it can be cracked in less than a second by any brute-force method. *Brute-forcing* is the method of trying every password possibility, starting with likely words (such as *password* and variants, or *god*) and then just trying random letters (for example, *a*, *aa*, *ab*, *ac*, and so on).

Even very strong passwords are no more than about 16 characters; such passwords take a long time to brute-force but can still be cracked. The solution is to use key-based logins, which generate a unique, 1024-bit private and public key pair for your machine. These keys take even the fastest computers a lifetime to crack, and you can back them up with a password to stop others from using them.

Creating an SSH key is done through the `ssh-keygen` command, like this:

```
ssh-keygen –t dsa
```

Press Enter when it prompts you where to save your key, and enter a passphrase when it asks you to. This passphrase is just a password used to protect the key—you can leave it blank if you want to, but doing so would allow other people to use your account to connect to remote machines if they manage to log in as you.

After the key is generated (it might take up to 30 seconds depending on the speed of your machine), change the directory to `.ssh` (`cd ~/.ssh`), which is a hidden directory where your key is stored and also where it keeps a list of safe SSH hosts. There you will see the files `id_dsa` and `id_dsa.pub`. The first is your private key and should never be given out. The second is your public key, which is safe for distribution. You need to copy the public key to each server you want to connect to via key-based SSH.

Using `scp`, you can copy the public key over to your server, like this:

```
scp id_dsa.pub 10.0.0.1:
```

This places `id_dsa.pub` in your home directory on `10.0.0.1`. The next step is to SSH into `10.0.0.1` normally and set up that key as an authorized key. So, you can SSH in as yourself and then type

```
touch .ssh/authorized_keys
cat id_dsa.pub >> .ssh/authorized_keys
chmod 400 .ssh/authorized_keys
```

The `touch` command creates the `authorized_keys` file (if it does not exist already); then you use `cat` to append the contents of `id_dsa.pub` to the list of already authorized keys. Finally, `chmod` is used to make `authorized_keys` read only.

With that done, you can type `exit` to disconnect from the remote machine and return to your local machine. Then you can try running `ssh` again. If you are prompted for your passphrase, you have successfully configured key-based authentication.

That is the current machine secured, but what about every other machine? It is still possible to log in from another machine using only a password, which means your remote machine is still vulnerable.

The solution to this is to switch to root and edit the `/etc/ssh/sshd_config` file. Look for the `PasswordAuthentication` line and make sure it reads `no` (and that it is not commented out with a `#`). Save the file, and run ``kill -HUP `cat /var/run/sshd.pid` `` to have `sshd` reread its configuration files. With that done, `sshd` accepts only connections from clients with authorized keys, which stops crackers from brute-forcing their way in.

TIP

For extra security, consider setting `PermitRootLogin` to `no` in `/etc/ssh/sshd_config`. When this is set, it becomes impossible to SSH into your machine using the root account—you must connect with a normal user account and then use `su` or `sudo` to switch to root. This is advantageous because most brute-force attempts take place on the root account because it is the only account that is guaranteed to exist on a server.

Also, even if a cracker knows your user account, she has to guess both your user password and your root password to take control of your system.

Remote X

Everything we have looked at so far has been about command-line remoting, with no mention so far of how to bring up a graphical user interface. There are two ways of doing this in Linux: the X Display Manager Control Protocol (XDMCP) and Virtual Network Computing (VNC). The former is specific to X Windows and is very tightly integrated with the rest of the graphical system but is also very insecure. VNC is more modern and very widespread but insecure in some implementations. Both are being used with Ubuntu, so we will cover both here.

XDMCP

Unless you have Ubuntu configured to log in a specific user automatically, you will be familiar with the user login screen that appears at bootup. What you are seeing is the Gnome Display Manager (GDM), which runs your X sessions, checks passwords, and so forth. What you are doing is logging in to the local machine because that is the default configuration.

However, GDM is also equipped to allow other network users to connect to your machine through the XDMCP protocol. There are various reasons for using XDMCP, of which the most popular is that many modern machines are large and noisy. They have big hard drives, CPUs with huge fans, and powerful graphics cards, and so do not fit into a peaceful living room. On the flip side, a thin client (a machine with very little CPU power and no hard disk of its own) is silent but not powerful enough to run Gnome or OpenOffice.org.

The solution is to have your powerful machine locked away in a cupboard somewhere with a Wi-Fi connection attached and your quiet thin client sitting in the lounge also on the Wi-Fi link. The thin client connects to the powerful machine and runs all its programs from there, with all the graphics being relayed over the network.

With Ubuntu, this is easy to do. Starting with the server side first, select System > Administration > Login Window; then select the Remote tab and change the Style option to "Plain with face browser". On the client side, go to the same dialog and make sure the Show Actions Menu box is checked from the Local tab.

Now, from the client side, log out from your desktop so you return to the Ubuntu login screen. When it prompts you for your username, look for the Options button and select Remote Login with XDMCP. A new dialog box appears with a list of local XDMCP servers that are willing to accept your connection—you should see your server in there. Select it and click Connect; you will see a login screen from that server, inviting you to log in. You will, of course, need a valid account on the server to be able to log in; however, that is the only thing you need.

As you can see, because XDMCP is so core to the X Windows system, it is easy to set up. However, as you will find as you use it, XDMCP is very slow—even on a Gigabit Ethernet network, it will chew up a substantial percentage of bandwidth. It is also insecure. Anyone can monitor what you are doing with very little work. Because of these two flaws, XDMCP should never be used outside a trusted network.

VNC

The next step up from XDMCP is VNC, which was developed at AT&T's Cambridge Research Laboratory in England. VNC is widespread in the Linux world and, to a lesser extent, in the Windows world. Its main advantage is its widespread nature: Nearly all Linux distros bundle VNC, and clients are available for a wide selection of platforms.

To set up VNC, start Synaptic and install the vnc-common and xvncviewer packages. With that done, all that remains is to tell Ubuntu who should be allowed to connect to your session. This is done from the Remote Desktop option on the Preferences menu. By default, your desktop is not shared, so check Allow Other Users to View Your Desktop to share it. You should also check Allow Other Users to Control Your Desktop; otherwise, people will be able to see what you are doing but not interact with the desktop—which is not very helpful.

The second set of options on that screen is important. If you are using this as a remote way to connect to your own desktop, deselect Ask You for Confirmation. If this is not done, when you try to connect from your remote location, Ubuntu will pop up a message box on the local machine asking `Should this person be allowed to connect?` Because you are not there to click Yes, the connection will fail. If you want to let someone else remotely connect to your system, keep this box enabled so you know when people are connecting. You should always enter a password, no matter who it is that will connect. VNC, like XDMCP, should not be considered secure over the Internet, or even on untrusted networks.

Reference

- http://www.openssh.com—The home page of the OpenSSH implementation of SSH that Ubuntu uses. It is run by the same team as OpenBSD, a secure BSD-based operating system.
- http://www.realvnc.com—The home page of the team that made VNC at AT&T's Cambridge Research Laboratory. It has since started RealVNC Ltd., a company dedicated to developing and supporting VNC.
- http://www.tightvnc.com—Here you can find an alternative to VNC called TightVNC that has several key advances over the stock VNC release. The most important feature is that TightVNC can use SSH for encryption, guaranteeing security.
- http://www.nomachine.com/—Another alternative to VNC is in the pipeline, called NX. The free implementation, FreeNX, is under heavy development at the time of writing but promises to work much faster than VNC.

One book on SSH that stands out from the crowd is known as "The Snail Book" because of the picture on the cover. It covers all the SSH suite and is called *SSH: The Secure Shell* (O'Reilly), ISBN: 0-596-00011-1.

PART IV

Ubuntu As a Server

IN THIS PART

This page intentionally left blank

CHAPTER 16

File and Print

IN THIS CHAPTER

In the early days of computing, file and printer sharing was pretty much impossible because of the lack of good networking standards and interoperability. If you wanted to use a printer connected to another computer, you had to save the file to a floppy disk and walk over.

Nowadays, both file and printer sharing have become second nature in a world where it is not unusual for someone to own more than one computer. Whether it be for sharing photographs among various computers or having a central repository available for collaboration, file sharing is an important part of our information age. Alongside this is the need to be able to share printers; after all, no one wants to have to plug and unplug a computer to a printer just so he can print out a quick letter.

Whatever your reasons for needing to share files and printers across a network, you find out how to do both in this chapter. We look at how you can share files using the popular UNIX NFS protocol and the more Windows-friendly Samba system. You also find out how to configure network-attached printers with interfaces such as JetDirect. We look at both graphical and command-line tools, so you should find something to suit the way you work.

CAUTION

By default, Ubuntu ships with all its network ports blocked. That is, it does not listen to any requests on any ports when it is first installed. To configure the firewall, you must retrieve and install Firestarter using either `synaptic` or `apt-get`. After you have installed this, you can use it to configure the firewall to open specific ports relating to the topics covered in this chapter.

You can find detailed documentation about Firestarter at http://www.fs-security.com/docs/.

Using the Network File System

Network File System (NFS) is the protocol developed by Sun Microsystems that allows computers to use a remote file system as if it were a real part of the local machine. A common use of NFS is to allow users home directories to appear on every local machine they use, thus eliminating the need to have physical home directories. This opens up hot desking and other flexible working arrangements, especially because no matter where the user is, his home directory follows him around.

Another popular use for NFS is to share binary files between similar computers. If you have a new version of a package that you want all machines to have, you have to the upgrade only on the NFS server, and all hosts running the same version of Ubuntu will have the same upgraded package.

Installing and Starting or Stopping NFS

NFS is not installed by default on Ubuntu, so you need to install the `nfs-common`, `nfs-kernel-server`, and `portmap` packages. NFS itself consists of several programs that work together to provide the NFS server service. One is `rpc.portmapper`, which maps NFS requests to the correct daemon. Two others are `rpc.nfsd`, which is the NFS daemon, and `rpc.mountd`, which controls the mounting and unmounting of file systems.

Ubuntu automatically adds NFS to the system startup scripts, so it will always be available after you have configured it. To check this, use the command `sudo /etc/init.d/nfs-kernel-server status` and it will show you that the service is running. If you need to manually start the NFS server, use the following command:

```
$ sudo /etc/init.d/nfs-kernel-server start
Starting NFS services:                    [ OK ]
Starting NFS quotas:                      [ OK ]
Starting NFS daemon:                      [ OK ]
Starting NFS mountd:                      [ OK ]
```

In this example, NFS has been started. Use the `stop` keyword instead to stop the service, or `restart` to restart the server. This approach to controlling NFS proves handy, especially after configuration changes have been made. See the next section on how to configure NFS support on your Ubuntu system.

NFS Server Configuration

You can configure the NFS server by editing the `/etc/exports` file. This file is similar to the `/etc/fstab` file in that it is used to set the permissions for the file systems being exported. The entries look like this:

```
/file/system yourhost(options) *.yourdomain.com(options) 192.168.0.0/24(options)
```

This shows three common clients to which to share `/file/system`. The first, `yourhost`, shares `/file/system` to just one host. The second, `.yourdomain.com`, uses the asterisk (*)

as a wildcard to enable all hosts in yourdomain.com to access /file/system. The third share enables all hosts of the Class C network, 192.168.0.0, to access /file/share. For security, it is best not to use shares like the last two across the Internet because all data will be readable by any network the data passes by.

Table 16.1 shows some common options.

TABLE 16.1 /etc/fstab Options

Option	Purpose
rw	Gives read and write access
ro	Gives read-only access
async	Writes data when the server, not the client, feels the need
sync	Writes data as it is received

The following is an example of an /etc/exports file:

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
/home/andrew 192.168.0.0/24(rw,no_root_squash)
```

This file exports (makes available) /home/ahudson to any host in 192.168.0.* and allows users to read from and write to /home/andrew.

After you have finished with the /etc/exports file, the following command

```
$ sudo exportfs -a
```

exports all the file systems in the /etc/exports file to a list named xtab under the /var/lib/nfs directory, which is used as a guide for mounting when a remote computer asks for a directory to be exported. The -r option to the command reads the entire /etc/exports file and mounts all the entries. You can also use the exportfs command to export specific files temporarily. Here's an example using exportfs to export a file system:

```
/usr/sbin/exportfs -o async yourhost:/usr/tmp
```

This command exports /usr/tmp to yourhost with the async option.

Be sure to restart the NFS server after making any changes to /etc/exports. If you prefer, you can use Ubuntu's shares-admin graphical client to set up NFS while using the X Window System. Start the client by clicking the System menu and then selecting the Shared Folders menu item from the Administration menu.

After you press Enter, you are prompted for your password. Type in the password and click OK; the main window will then display. Click the Add button to open the Add Share dialog box, as shown in Figure 16.1.

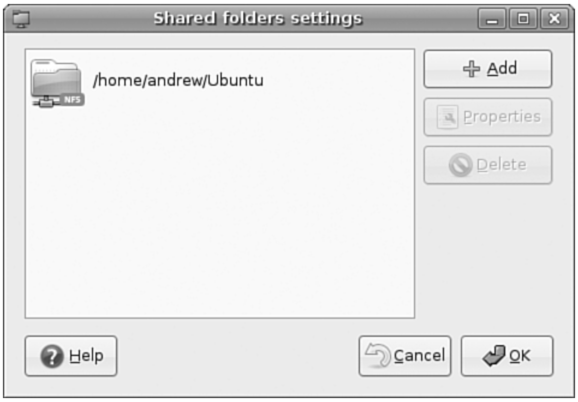


FIGURE 16.1 You can use Ubuntu's `share-admin` client to quickly set up local directories for export using NFS.

In the Path drop-down box, choose the directory that you want to share; in the Share drop-down box, choose NFS. Click the Add Host button that appears to specify which hosts, IP addresses, or networks to be allowed access to the directory. By default, a directory is exported as read/write, but you can choose read-only by ticking the Read Only option. When finished, click the OK button, click the Apply button, and then use the File menu to quit.

NFS Client Configuration

To configure your host as an NFS client (to acquire remote files or directories), edit the `/etc/fstab` file as you would to mount any local file system. However, instead of using a device name to be mounted (such as `/dev/hda1`), enter the remote hostname and the desired file system to be imported. For example, one entry might look like this:

```
# Device             Mount Point  Type  Options       Freq Pass
yourhost:/usr/local  /usr/local   nfs   nfsvers=3,ro  0    0
```

NOTE

If you use `autofs` on your system, you need to use proper `autofs` entries for your remote NFS mounts. See the `section 5` man page for `autofs`.

The Options column uses the same options as standard `fstab` file entries with some additional entries, such as `nfsvers=3`, which specifies the third version of NFS. You can also use the `mount` command, as root, to quickly attach a remote directory to a local file system by using a remote host's name and exported directory. For example:

```
$ sudo mount -t nfs 192.168.2.67:/music /music
```

After you press Enter, the entire remote directory appears on your file system. You can verify the imported file system using the `df` command, as follows:

```
$ df
Filesystem           1k-blocks     Used Available Use% Mounted on
/dev/hda2             18714368  9642600   8121124  55% /
/dev/hda1                46636    13247     30981  30% /boot
none                    120016        0    120016   0% /dev/shm
192.168.2.67:/music   36875376 20895920  14106280  60% /music
```

Make sure that the desired mount point exists before using the mount command. When finished using the directory (perhaps for copying backups), you can use the umount command to remove the remote file system. Note that if you specify the root directory (/) as a mount point, you cannot unmount the NFS directory until you reboot (because Linux complains that the file system is in use).

Putting Samba to Work

Samba uses the *Session Message Block (SMB)* protocol to enable the Windows operating system (or any operating system) to access Linux files. Using Samba, you can make your Ubuntu machine look just like a Windows computer to other Windows computers on your network. You do not need to install Windows on your PC.

Samba is a complex program—the Samba man page (when converted to text) for just the configuration file is 330KB and 7,013 lines long. Although Samba is complex, setting it up and using it does not have to be difficult. There are many options, which accounts for some of Samba's complexity. Depending on what you want, Samba's use can be as easy or as difficult as you would like it to be.

Fortunately, Ubuntu includes the *Samba Web Administration Tool (SWAT)*, which you can use to configure Samba by using the Mozilla web browser. SWAT provides an easy way to start and stop the Samba server; set up printing services; define remote access permissions; and create Samba usernames, passwords, and shared directories. This section delves into the basics of configuring Samba, and you should first read how to manually configure Samba to get an understanding of how the software works. At the end of this section, you will see how to enable, start, and use SWAT to set up simple file sharing.

Like most of the software that comes with Ubuntu, Samba is licensed under the GPL and is free. Installation is straightforward and the software can be installed using either synaptic or apt-get. Regardless of which route you take, you should snag the samba and swat packages.

Installing from source code can be more time-consuming. If you do not want to install using Ubuntu's default locations, however, installing from the source code is a more configurable method. Just download the source from http://www.samba.org/ and unpack the files. Change into the source directory and, as root, run the command ./configure along with any changes from the defaults. Then run make, make test (if you want), followed by make install to install Samba in the specified locations.

NOTE

If you haven't done so, you might need to install a meta-package (a single package that provides several related packages) called `build-essential`. This package automatically installs all the `make` tools, plus other development utilities that you will need to compile software from source code.

When you install Samba, it is a good idea to also install the `samba-doc` and `samba-doc-pdf` packages because they contain extensive documentation in text, PDF, and HTML format. After you install it, you can find this documentation in `/usr/share/doc/samba*/doc`. If you install Samba using your Ubuntu disc, you can find a large amount of documentation in the directory tree starting at `/usr/share/doc/samba-doc` or `/usr/share/doc/samba-doc-pdf` in several formats, including PDF, HTML, and text, among others. Altogether, almost 3MB of documentation is included with the source code.

After installing Samba, you can either create the file `/etc/smb.conf` or use the `smb.conf` file supplied with Samba, which is located by default under the `/etc/samba` directory with Ubuntu. You can find nearly a dozen sample configuration files under the `/usr/share/doc/samba*/examples` directory.

NOTE

Depending on your needs, `smb.conf` can be a simple file of fewer than 20 lines or a huge file spanning many pages of text. If your needs are complex, I suggest you browse through The Official Samba Howto and Reference Guide, or TOSHARG. This helpful guide can be found at http://samba.org/samba/docs/man/Samba3-HOWTO/.

Manually Configuring Samba with `/etc/samba/smb.conf`

The `/etc/samba/smb.conf` file is broken into sections. Each section is a description of the resource shared (share) and should be titled appropriately. The three special sections are as follows:

- `[global]`—Establishes the global configuration settings (defined in detail in the `smb.conf` man page and Samba documentation, found under the `/usr/share/doc/samba/docs` directory)
- `[homes]`—Shares users' home directories and specifies directory paths and permissions
- `[printers]`—Handles printing by defining shared printers and printer access

Each section in your `/etc/samba/smb.conf` configuration file should be named for the resource being shared. For example, if the resource `/usr/local/programs` is being shared, you could call the section `[programs]`. When Windows sees the share, it is called by whatever you name the section (`programs` in this example). The easiest and fastest way to set up this share is with the following example from `smb.conf`:

```
[programs]
path = /usr/local/programs
writeable = true
```

This bit shares the `/usr/local/programs` directory with any valid user who asks for it and makes that directory writable. It is the most basic share because it sets no limits on the directory.

Here are some parameters you can set in the sections:

- Requiring a user to enter a password before accessing a shared directory
- Limiting the hosts allowed to access the shared directory
- Altering permissions users are allowed to have on the directory
- Limiting the time of day during which the directory is accessible

The possibilities are almost endless. Any parameters set in the individual sections override the parameters set in the `[global]` section. The following section adds a few restrictions to the `[programs]` section:

```
[programs]
path = /usr/local/programs
writeable = true
valid users = ahudson
browseable = yes
create mode = 0700
```

The `valid users` entry limits `userid` to just `ahudson`. All other users can browse the directory because of the `browseable = yes` entry, but only `ahudson` can write to the directory. Any files created by `ahudson` in the directory give `ahudson` full permissions, but no one else will have access to the file. This is exactly the same as setting permissions with the `chmod` command. Again, there are numerous options, so you can be as creative as you want to when developing sections.

Setting Global Samba Behavior with the `[global]` Section

The `[global]` section set parameters establishes configuration settings for all of Samba. If a given parameter is not specifically set in another section, Samba uses the default setting in the `[global]` section. The `[global]` section also sets the general security configuration for Samba. The `[global]` section is the only section that does not require the name in brackets.

Samba assumes that anything before the first bracketed section not labeled `[global]` is part of the global configuration. (Using bracketed headings in `/etc/samba/smb.conf` makes your configuration file more readable.) The following sections discuss common Samba settings to share directories and printers. You will then see how to test your Samba configuration.

Sharing Home Directories Using the `[homes]` Section

The `[homes]` section shares out Ubuntu home directories for the users. The home directory is shared automatically when a user's Windows computer connects to the Linux server holding the home directory. The one problem with using the default configuration is that the user sees all the configuration files (such as `.profile` and others with a leading period in the filename) that he normally wouldn't see when logging on through Linux. One quick way to avoid this is to include a path option in the `[homes]` section. To use this solution, each user who requires a Samba share of his home directory needs a separate "home directory" to act as his Windows home directory.

For example, this pseudo home directory could be a directory named `share` in each user's home directory on your Ubuntu system. You can specify the path option when using SWAT by using the `%u` option when specifying a path for the default `homes` shares (see the section "Configuring Samba Using SWAT" later in this chapter). The complete path setting would be this:

```
/home/%u/share
```

This setting specifies that the directory named `share` under each user's directory is the shared Samba directory. The corresponding manual `smb.conf` setting to provide a separate "home directory" looks like this:

```
[homes]
        comment = Home Directories
        path = /home/%u/share
        valid users = %S
        read only = No
        create mask = 0664
        directory mask = 0775
        browseable = No
```

If you have a default `[homes]` section, the share shows up in the user's Network Neighborhood as the user's name. When the user connects, Samba scans the existing sections in `smb.conf` for a specific instance of the user's home directory. If there is not one, Samba looks up the username in `/etc/passwd`. If the correct username and password have been given, the home directory listed in `/etc/passwd` is shared out at the user's home directory. Typically, the `[homes]` section looks like this (the `browseable = no` entry prevents other users from being able to browse your home directory and is a good security practice):

```
[homes]
browseable = no
writable = yes
```

This example shares out the home directory and makes it writable to the user. Here's how you specify a separate Windows home directory for each user:

```
[homes]
browseable = no
```

```
writable = yes
path = /path/to/windows/directories
```

Sharing Printers by Editing the `[printers]` Section

The `[printers]` section works much like the `[homes]` section but defines shared printers for use on your network. If the section exists, users have access to any printer listed in your Ubuntu `/etc/printcap` file.

Like the `[homes]` section, when a print request is received, all the sections are scanned for the printer. If no share is found (with careful naming, there should not be unless you create a section for a specific printer), the `/etc/printcap` file is scanned for the printer name that is then used to send the print request.

For printing to work properly, you must correctly set up printing services on your Ubuntu computer. A typical `[printers]` section looks like the following:

```
[printers]
comment = Ubuntu Printers
browseable = no
printable = yes
path = /var/spool/samba
```

The `/var/spool/samba` is a spool path set just for Samba printing.

Testing Samba with the `testparm` Command

After you have created your `/etc/smb.conf` file, you can check it for correctness by using the `testparm` command. This command parses through your `/etc/smb.conf` file and checks for any syntax errors. If none are found, your configuration file will probably work correctly. It does not, however, guarantee that the services specified in the file will work. It is merely making sure that the file is correctly written.

As with all configuration files, if you are modifying an existing, working file, it is always prudent to copy the working file to a different location and modify that file. Then, you can check the file with the `testparm` utility. The command syntax is as follows:

```
$ sudo testparm /path/to/smb.conf.back-up
Load smb config files from smb.conf.back-up
Processing section "[homes]"
Processing section "[printers]"
Loaded services file OK.
```

This output shows that the Samba configuration file is correct, and, as long as all the services are running correctly on your Ubuntu machine, Samba should be working correctly. Now copy your old `smb.conf` file to a new location, put the new one in its place, and restart Samba with the command `/etc/init.d/smb restart`. Your new or modified Samba configuration should now be in place.

Starting the smbd Daemon

Now that your smb.conf file is correctly configured, you can start your Samba server daemon. This can be done with the /usr/sbin/smbd command, which (with no options) starts the Samba server with all the defaults. The most common option you will change in this command is the location of the smb.conf file; you change this option if you don't want to use the default location /etc/smb/smb.conf. The -s option allows you to change the smb.conf file Samba uses; this option is also useful for testing whether a new smb.conf file actually works. Another useful option is the -l option, which specifies the log file Samba uses to store information.

To start, stop, or restart Samba from the command line, use the /etc/init.d/samba script with a proper keyword, such as start, like so:

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

Using the smbstatus Command

The smbstatus command reports on the current status of your Samba connections. The syntax is as follows:

```
/usr/bin/smbstatus [options]
```

Table 16.2 shows some of the available options

TABLE 16.2 smbstatus Options

Option	Result
-b	Brief output
-d	Verbose output
-s */path/to/config*	Used if the configuration file used at startup is not the standard one
-u *username*	Shows the status of a specific user's connection
-p	Lists current smb processes, which can prove useful in scripts

Connecting with the smbclient Command

The smbclient command allows users on other Linux hosts to access your smb shares. You cannot mount the share on your host, but you can use it in a way that is similar to an FTP client. Several options can be used with the smbclient command. The most frequently used is -I followed by the IP address of the computer to which you are connecting. The smbclient command does not require root access to run:

```
smbclient -I 10.10.10.20 -Uusername%password
```

This gives you the following prompt:

```
smb: <current directory on share>
```

From here, the commands are almost identical to the standard UNIX/Linux FTP commands. Note that you can omit a password on the smbclient command line. You are then prompted to enter the Samba share password.

Mounting Samba Shares

There are two ways to mount Samba shares to your Linux host. Mounting a share is the same as mounting an available media partition or remote NFS directory except that the Samba share is accessed using SMB. The first method uses the standard Linux `mount` command:

```
$ sudo mount -t smbfs //10.10.10.20/homes /mount/point -o username=ahudson,dmask=777,\
 fmask=777
```

NOTE

You can substitute the IP address for hostname if your name service is running or the host is in your `/etc/hosts` file.

This command mounts `ahudson`'s home directory on your host and gives all users full permissions to the mount. The permissions are equal to the permissions on the `chmod` command.

The second method produces the same results using the `smbmount` command, as follows:

```
$ sudo smbmount //10.10.10.20/homes /mount/point -o username=ahudson,dmask-777,\
 fmask=777
```

To unmount the share, use the following standard command:

```
$ sudo umount /mount/point
```

You can also use these `mount` commands to mount true Windows client shares to your Ubuntu host. Using Samba, you can configure your server to provide any service Windows can serve, and no one but you will ever know.

Configuring Samba Using SWAT

The Samba team of developers has made administering Samba much easier with the *Samba Web Administration Tool (SWAT)*. SWAT is a web-based configuration and maintenance interface that gets as close to a point-and-click Samba environment as possible. This section provides a simple example of how to use SWAT to set up SMB access to a user's home directory and how to share a directory.

NOTE

Using SWAT requires you to install the `openbsd_inetd` and `swat` packages, so make sure you have these before proceeding.

Also you need to enable the root account by giving it a password by using the command `sudo passwd root`. Not enabling the root account prevents you from using SWAT effectively.

You need to perform a few steps before you can start using SWAT. First, make sure you have the Samba and the `swat` packages installed. You then enable SWAT access to your system by editing the `/etc/inetd.conf` file by changing the following lines to remove the `#<off>#` comments:

```
#<off># swat  stream  tcp  nowait.400  root\

  /usr/sbin/tcpd  /isr/sbin/swat
```

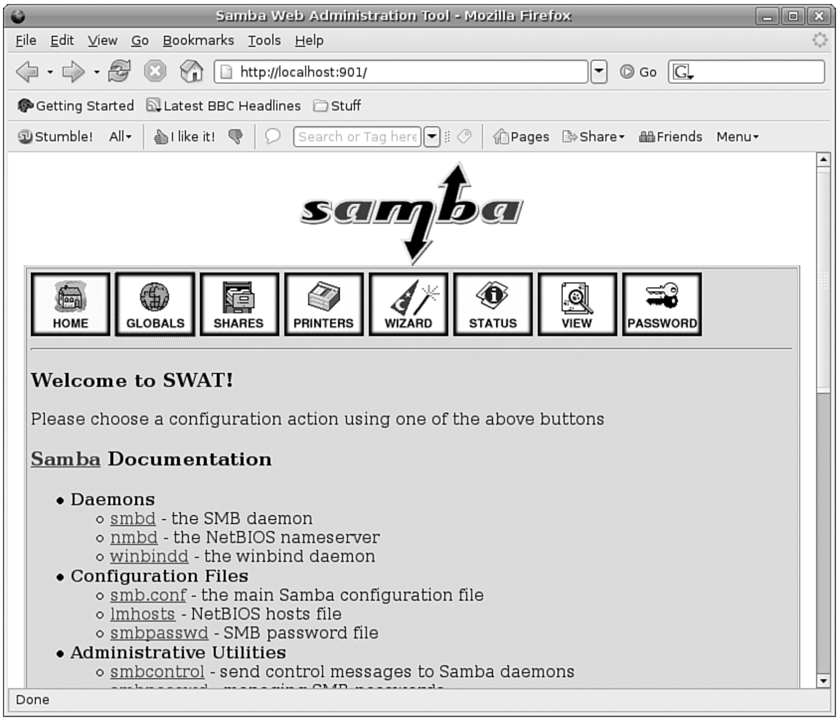
Save the file, and then restart the `openbsd_inetd` daemon using the following command:

```
$ sudo /etc/init.d/openbsd-inetd restart
```

Next, start an X session, launch Firefox, and browse to the http://localhost:901 *uniform resource locator (URL)*. You are presented a login prompt. Enter the root username and password, and then click the OK button. The screen clears, and you see the main SWAT page, as shown in Figure 16.2.

TIP

You can also configure Samba using Ubuntu's `shares-admin` client. Launch the client from the command line of an X terminal window or select the System, Administration, Shared Folders menu item (as shown later in Figure 16.9).



Welcome to SWAT!

Please choose a configuration action using one of the above buttons

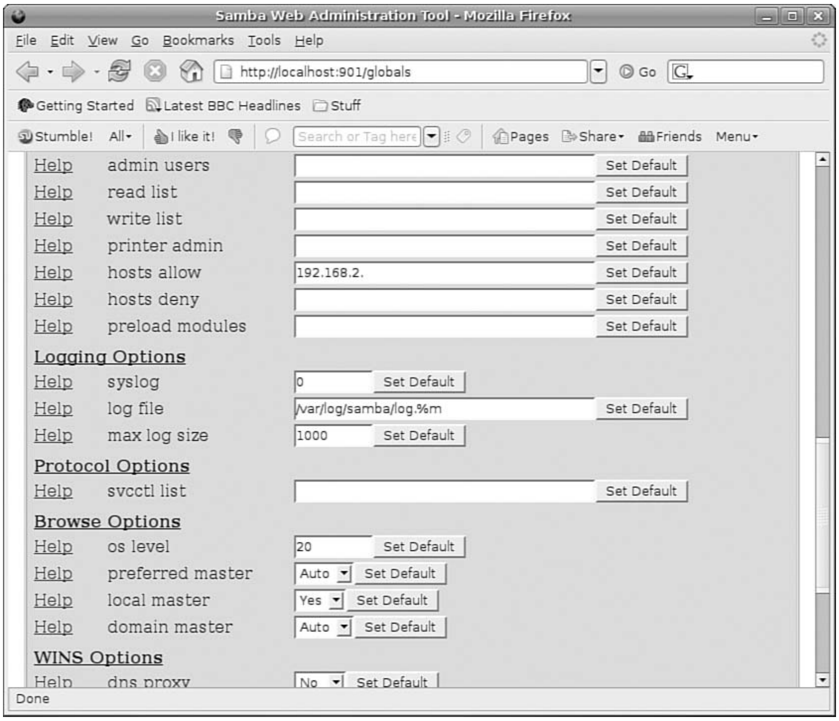
Samba Documentation

- **Daemons**
  - smbd - the SMB daemon
  - nmbd - the NetBIOS nameserver
  - winbindd - the winbind daemon
- **Configuration Files**
  - smb.conf - the main Samba configuration file
  - lmhosts - NetBIOS hosts file
  - smbpasswd - SMB password file
- **Administrative Utilities**
  - smbcontrol - send control messages to Samba daemons

FIGURE 16.2 You can use SWAT to easily configure and administer Samba on your system.

First, click the Globals icon in SWAT's main page. You see a page similar to the one shown in Figure 16.3. Many options are in the window, but you can quickly set up access for hosts from your LAN by simply entering one or more IP addresses or a subnet address (such as 192.168.0.—note the trailing period, which allows access for all hosts; in this example, on the 192.168.0 subnet) in the Hosts Allow field under the Security Options section. If you need help on how to format the entry, click the Help link to the left of the field. A new web page appears with the pertinent information.

When finished, click the Commit Changes button to save the global access settings. The next step is to create a Samba user and set the user's password. Click the Password icon on the main SWAT page (refer to Figure 16.2). The Server Password Management page opens, as shown in Figure 16.4. Type a new username in the User Name field; then type a password in the New Password and Re-type New Password fields.


Samba Web Administration Tool - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:901/globals
Go
Getting Started
Latest BBC Headlines
Stuff
Stumble! All
I like it!
Search or Tag here
Pages
Share
Friends
Menu
Done

Help	admin users	Set Default
Help	read list	Set Default
Help	write list	Set Default
Help	printer admin	Set Default
Help	hosts allow 192.168.2.	Set Default
Help	hosts deny	Set Default
Help	preload modules	Set Default
Logging Options		
Help	syslog 0	Set Default
Help	log file /var/log/samba/log.%m	Set Default
Help	max log size 1000	Set Default
Protocol Options		
Help	svcctl list	Set Default
Browse Options		
Help	os level 20	Set Default
Help	preferred master Auto	Set Default
Help	local master Yes	Set Default
Help	domain master Auto	Set Default
WINS Options		
Help	dns proxy No	Set Default

FIGURE 16.3 Configure Samba to allow access from specific hosts or subnets on your LAN.

NOTE

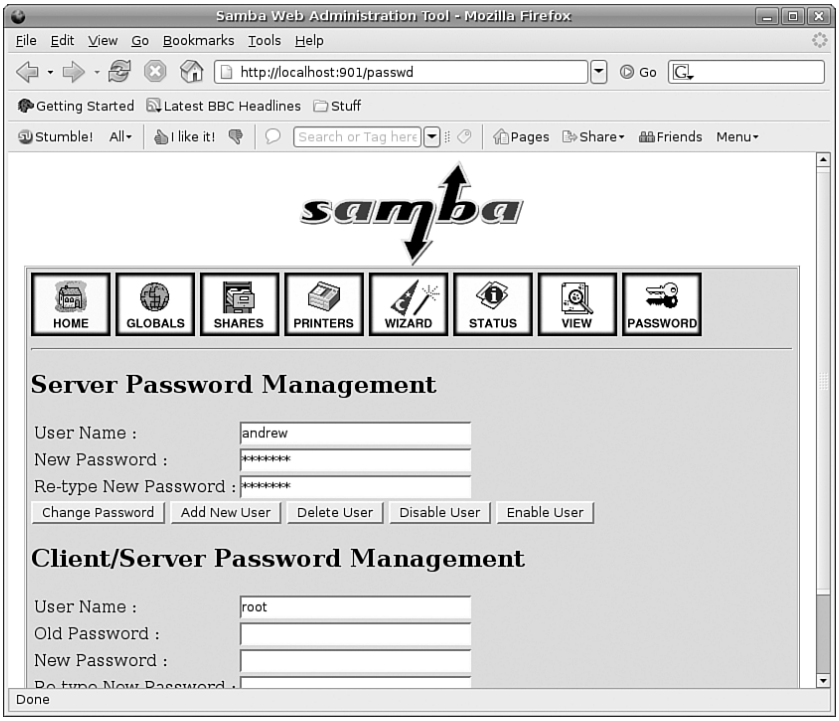
You must supply a username of an existing system user, but the password used for Samba access does not have to match the existing user's password.

When finished, click the Add New User button. SWAT then creates the username and password and displays `Added user` *`username`* (where *`username`* is the name you entered).

The new Samba user should now be able to gain access to the home directory from any allowed host if the Samba (smb) server is running.

For example, if you have set up Samba on a host named mini that has a user named andrew, the user can access the home directory on mini from any remote host (if allowed by the Globals settings), perhaps by using the smbclient command like so:

```
$ smbclient //mini/andrew -U andrew
added interface ip=192.168.0.68 bcast=192.168.0.255 nmask=255.255.255.0
Password:
Domain=[MYGROUP] OS=[Unix] Server=[Samba 2.2.5]
smb: \> pwd
Current directory is \\mini\andrew\
smb: \> quit
```



Server Password Management

User Name : andrew
New Password : ********
Re-type New Password : ********

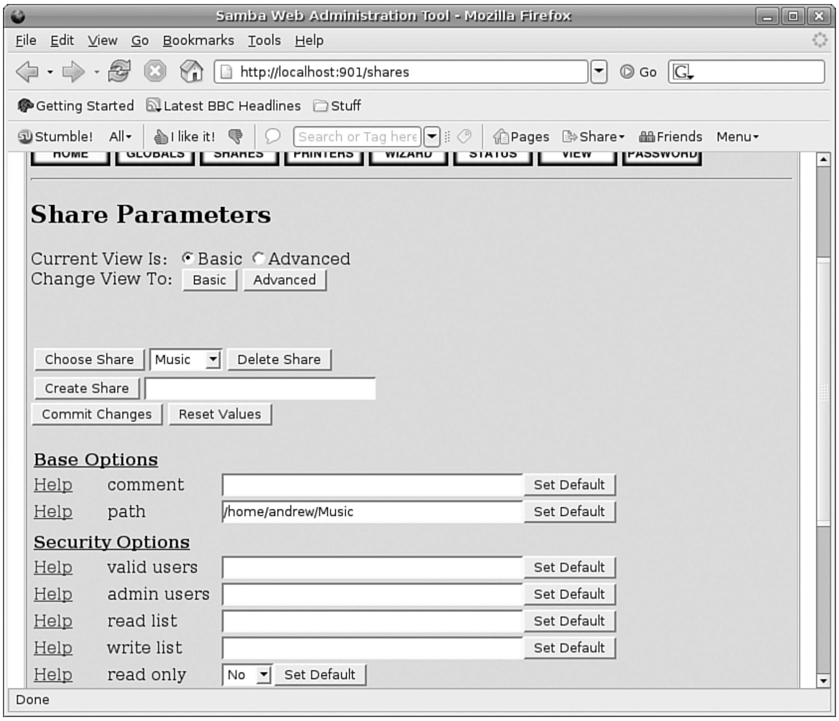
Client/Server Password Management

User Name : root
Old Password :
New Password :
Re-type New Password :

FIGURE 16.4 Enter a Samba username and password in the SWAT Password page.

Click the Status icon (as shown in Figure 16.2 or 16.4) to view Samba's status or to start, stop, or restart the server. You can use various buttons on the resulting web page to control the server and view periodic or continuous status updates.

You can also use SWAT to share a Linux directory. First, click the Shares icon in the toolbar at the top of the main Samba page (refer to Figure 16.2). Then, type a share name in the Create Shares field, and then click the Create Shares button. The SWAT Shares page displays the detailed configuration information in a dialog box, as shown in Figure 16.5, providing access to detailed configuration for the new Samba share.


Samba Web Administration Tool - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:901/shares
Getting Started Latest BBC Headlines Stuff
HOME GLOBALS SHARES PRINTERS WIZARD STATUS VIEW PASSWORD
Choose Share
Delete Share
Create Share
Commit Changes
Reset Values
Help
Set Default
Done

Share Parameters

Current View Is: [✓ Basic] Advanced
Change View To: Basic Advanced

Music

Base Options

comment

path /home/andrew/Music

Security Options

valid users

admin users

read list

write list

read only No

FIGURE 16.5 Use the SWAT Shares page to set up sharing of a portion of your Linux file system.

Type the directory name (such as `/music`) you want to share in the Path field under the Base options. Select No or Yes in the Read Only field under Security options to allow or deny read and write access. Select Yes in the Guest OK option to allow access from other users and specify a hostname, IP address, or subnet in the Hosts Allow field to allow access. Click the Commit Changes button when finished. Remote users can then access the shared volume. This is how a Linux server running Samba can easily mimic shared volumes in a mixed computing environment!

Alternatively, use the `shares-admin` client (from the command line or the Server Settings Samba Server menu item on the System Settings menu). Figure 16.6 shows the properties of a shared directory named `/music`. Use the Add button to create new shares and the Properties button to edit the share's access options.

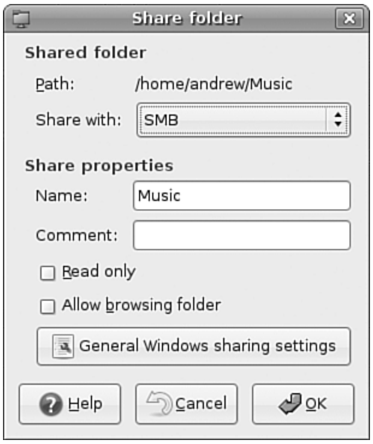


FIGURE 16.6 Configure a Samba share by editing the share defaults.

Network and Remote Printing with Ubuntu

Chapter 8, "Printing with Ubuntu," discussed how to set up and configure local printers and the associated print services. This section covers configuring printers for sharing and access across a network.

Offices all over the world benefit from using print servers and shared printers. In my office, I have two printers connected to the network via a Mac mini with Ubuntu PPC so that my wife can print from downstairs using a wireless link, and I can print from my three computers in my office. It is a simple thing to do and can bring real productivity benefits, even in small settings.

Creating Network Printers

Setting up remote printing service involves configuring a print server and then creating a remote printer entry on one or more computers on your network. This section introduces a quick method of enabling printing from one Linux workstation to another Linux computer on a LAN. You also learn about SMB printing using Samba and its utilities. Finally, this section discusses how to configure network-attached printers and use them to print single or multiple documents.

Enabling Network Printing on a LAN

If the computer with an attached printer is using Ubuntu and you want to set up the system for print serving, again use the `system-config-printer` client to create a new printer.

To enable sharing, you must follow a few steps, because by default CUPS is not set up to share printers across a network.

First, edit your `/etc/cups/cupsd.conf` file using the following command:

```
$ sudo gedit /etc/cups/cupsd.conf
```

In this example I have used `gedit`, but feel free to substitute in your favorite text editor.

Then, look for the section that begins with `<Location />` and modify it so that it reads as follows:

```
<Location />
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From 192.168.0.*
</Location>
```

This tells CUPS to share your printers across the network 192.168.0.*, for example. Make sure and change this to match your own network settings.

Next you need to look in the same file for the section that starts

```
Listen localhost:631
```

and modify it to show this:

```
Listen 631
```

This tells CUPS to listen on port 631 for any print requests. All you need to do now is open up `system-config-printer` on your clients and select Detect LAN Printers under the Global Settings menu. After a few short moments, the printer should pop up in your window, as shown in Figure 16.7

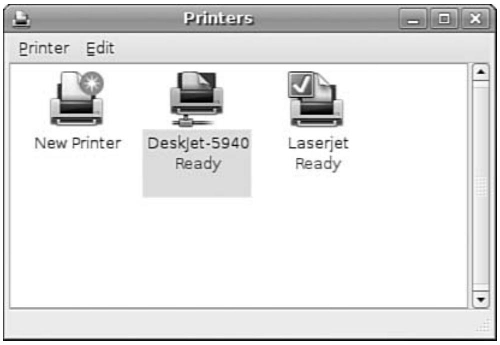


FIGURE 16.7 The highlighted printer has been shared across the network. Ubuntu makes this process as painless as possible.

Session Message Block Printing

Printing to an SMB printer requires Samba, along with its utilities such as the `smbclient` and associated `smbprint` printing filter. You can use the Samba software included with Ubuntu to print to a shared printer on a Windows network or set up a printer attached to your system as an SMB printer. This section describes how to create a local printer entry to print to a remote shared printer using SMB.

Setting up an SMB or shared printer is usually accomplished under Windows operating systems through configuration settings using the Control Panel's Network device. After enabling print sharing, reboot the computer. In the My Computer, Printers folder, right-click the name or icon of the printer you want to share and select Sharing from the pop-up menu. Set the Shared As item, and then enter a descriptive shared name, such as **`HP2100`**, and a password.

You must enter a shared name and password to configure the printer when running Linux. You also need to know the printer's workgroup name, IP address, and printer name and have the username and password on hand. To find this information, select Start, Settings, Printers; then right-click the shared printer's listing in the Printers window and select Properties from the pop-up window.

You can use CUPS to configure Samba to use your printers by editing the `smb.conf` file.

In the global section enter the following lines, if they are not already there:

```
...
load printers = yes
printing = cups
printcap name = cups
```

This tells Samba to use CUPS to provide printing services. Next you need to create a new section in the `smb.conf` file at the end of the file, as follows:

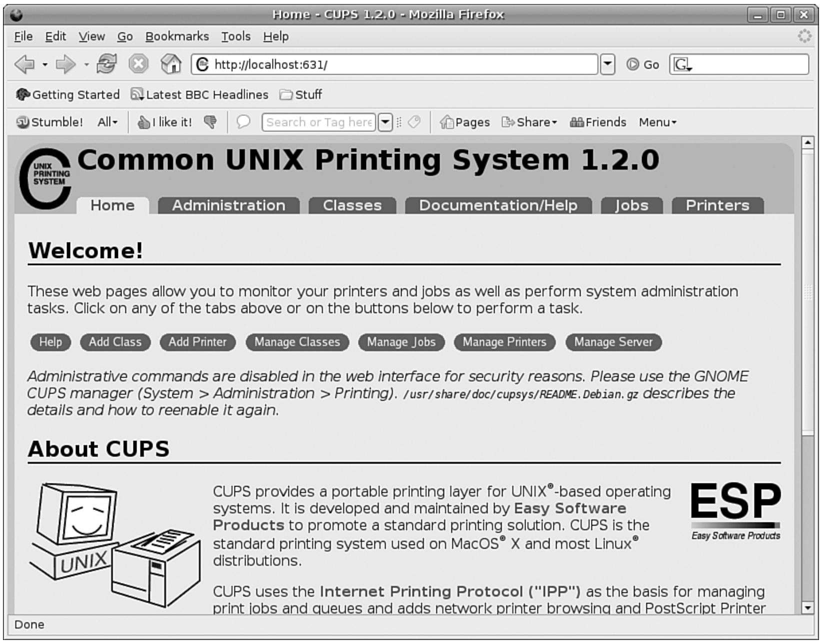
```
[printers]
comment = Use this for All Printers
path = /var/spool/samba
browseable = no
public = yes
guest ok = yes
writable = no
printable = yes
printer admin = root, andrew
```

This publishes your printers to the network and allows others to connect to them via Windows clients.

Make sure you restart the Samba service using the command shown earlier to make Samba pick up the changes to the configuration file.

Using the Common UNIX Printing System GUI

You can use CUPS to create printer queues, get print server information, and manage queues by launching a browser (such as Firefox) and browsing to http://localhost:631. CUPS provides a web-based administration interface, as shown in Figure 16.8.



Common UNIX Printing System 1.2.0

Welcome!

These web pages allow you to monitor your printers and jobs as well as perform system administration tasks. Click on any of the tabs above or on the buttons below to perform a task.

Administrative commands are disabled in the web interface for security reasons. Please use the GNOME CUPS manager (System > Administration > Printing). `/usr/share/doc/cupsys/README.Debian.gz` *describes the details and how to reenable it again.*

About CUPS

CUPS provides a portable printing layer for UNIX®-based operating systems. It is developed and maintained by **Easy Software Products** to promote a standard printing solution. CUPS is the standard printing system used on MacOS® X and most Linux® distributions.

CUPS uses the **Internet Printing Protocol ("IPP")** as the basis for managing print jobs and queues and adds network printer browsing and PostScript Printer

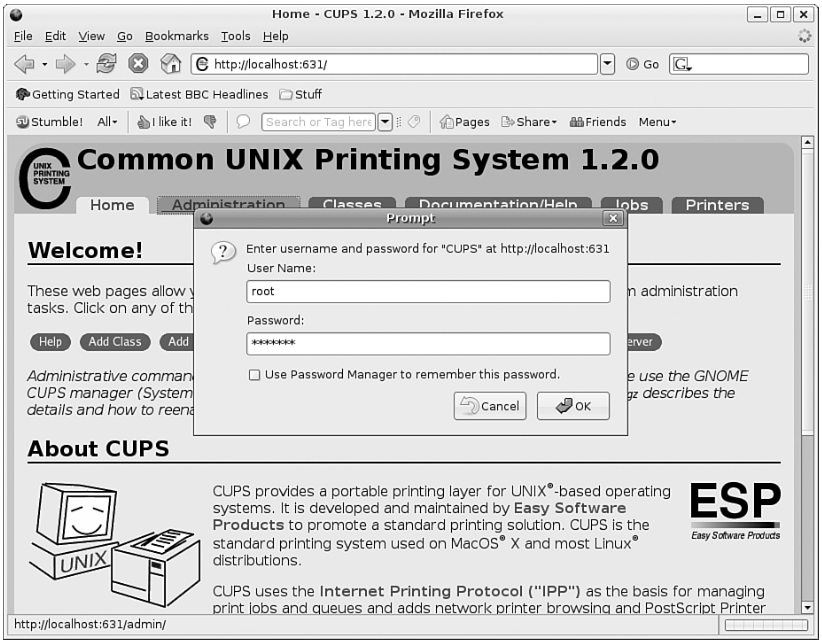
FIGURE 16.8 Use the web-based CUPS administrative interface to configure and manage printing.

If you click the Administration tab in the browser page, you are asked to enter the root password, as shown in Figure 16.9.

Creating a CUPS Printer Entry

This section provides a short example of creating a Linux printer entry using CUPS's web-based interface. Use the CUPS interface to create a printer and device queue type (such as local, remote, serial port, or Internet); then you enter a device *uniform resource identifier*

(URI), such as `lpd://192.168.2.35/lp`, which represents the IP address of a remote UNIX print server, and the name of the remote print queue on the server. You also need to specify the model or make of printer and its driver. A Printers page link allows you to print a test page, stop the printing service, manage the local print queue, modify the printer entry, or add another printer.


Home - CUPS 1.2.0 - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:631/
Go
Getting Started
Latest BBC Headlines
Stuff
Stumble! All I like it! Search or Tag here Pages Share Friends Menu
Common UNIX Printing System 1.2.0
Home Administration Classes Documentation/Help Jobs Printers
Prompt
Help Add Class Add
Cancel
OK
http://localhost:631/admin/

Enter username and password for "CUPS" at http://localhost:631
User Name: root
Password: *******
Use Password Manager to remember this password.

Welcome!

These web pages allow y... n administration tasks. Click on any of th...

Administrative comman... e use the GNOME CUPS manager (System... gz describes the details and how to reena...

About CUPS

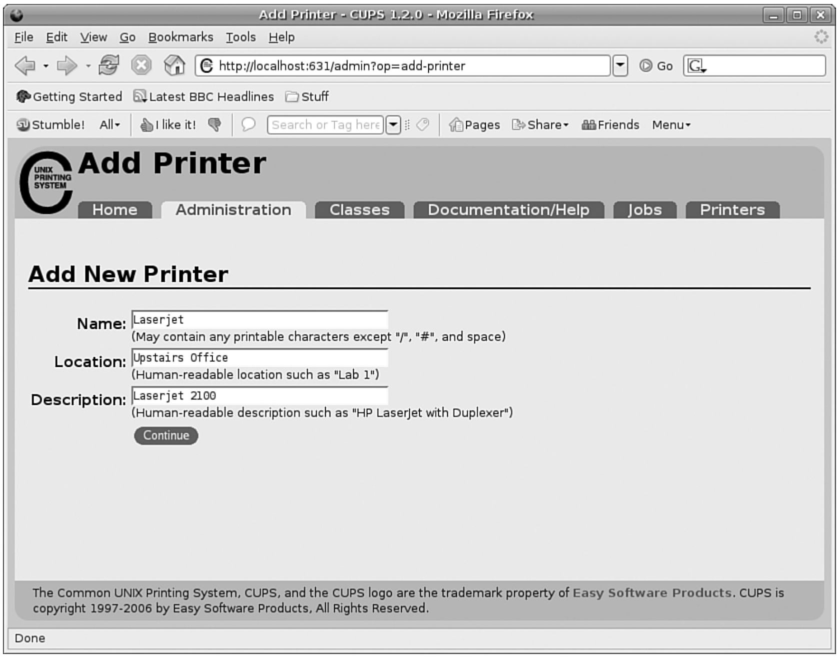
CUPS provides a portable printing layer for UNIX®-based operating systems. It is developed and maintained by **Easy Software Products** to promote a standard printing solution. CUPS is the standard printing system used on MacOS® X and most Linux® distributions.

CUPS uses the **Internet Printing Protocol ("IPP")** as the basis for managing print jobs and queues and adds network printer browsing and PostScript Printer

FIGURE 16.9 Enter the root password to perform printer administration with CUPS.

In the Admin page, click the Add Printer button and then enter a printer name in the Name field (such as `lp`), a physical location of the printer in the Location field, and a short note about the printer (such as its type) in the Description field. Figure 16.10 shows a sample entry for an HP 2100 LaserJet.

Click the Continue button. You can then select the type of printer access (local, remote, serial port, or Internet) in the Device page, as shown in Figure 16.11. For example, to configure printing to a local printer, select LPT1 or, for a remote printer, select the LPD/LPR Host or Printer entry.


Add Printer - CUPS 1.2.0 - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:631/admin?op=add-printer
Add Printer
Home Administration Classes Documentation/Help Jobs Printers

Add New Printer

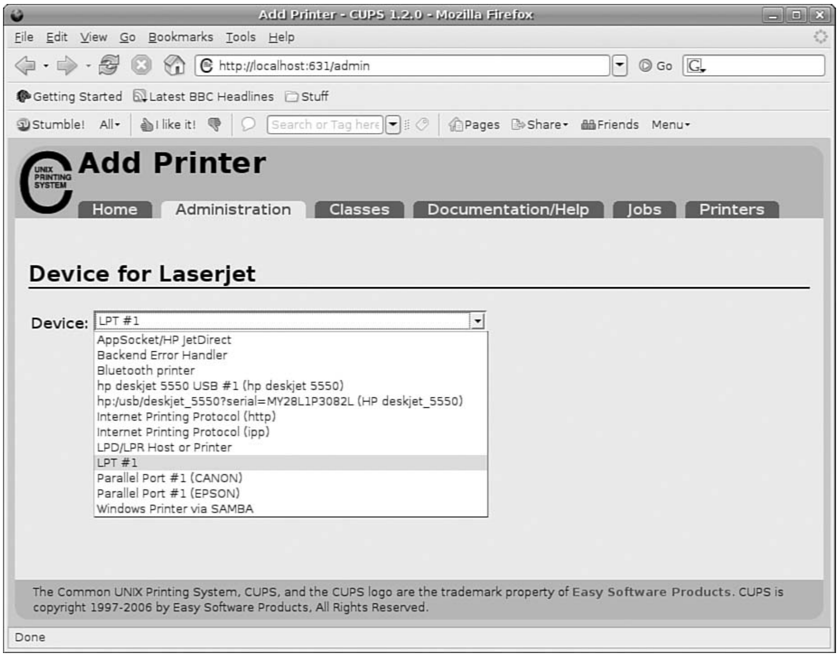
Name: Laserjet
(May contain any printable characters except "/", "#", and space)

Location: Upstairs Office
(Human-readable location such as "Lab 1")

Description: Laserjet 2100
(Human-readable description such as "HP LaserJet with Duplexer")

The Common UNIX Printing System, CUPS, and the CUPS logo are the trademark property of Easy Software Products. CUPS is copyright 1997-2006 by Easy Software Products, All Rights Reserved.

FIGURE 16.10 Use CUPS to create a new printer queue.


Add Printer - CUPS 1.2.0 - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:631/admin
Add Printer
Home Administration Classes Documentation/Help Jobs Printers

Device for Laserjet

Device: LPT #1

AppSocket/HP JetDirect
Backend Error Handler
Bluetooth printer
hp deskjet 5550 USB #1 (hp deskjet 5550)
hp:/usb/deskjet_5550?serial=MY28L1P3082L (HP deskjet_5550)
Internet Printing Protocol (http)
Internet Printing Protocol (ipp)
LPD/LPR Host or Printer
LPT #1
Parallel Port #1 (CANON)
Parallel Port #1 (EPSON)
Windows Printer via SAMBA

The Common UNIX Printing System, CUPS, and the CUPS logo are the trademark property of Easy Software Products. CUPS is copyright 1997-2006 by Easy Software Products, All Rights Reserved.

FIGURE 16.11 Select a printer device in the CUPS administrative page.

Again click Continue and select a printer make as requested in the dialog box shown in Figure 16.12.

After you click Continue, you then select the driver. After creating the printer, you can then use the Printer page, as shown in Figure 16.13, to print a test page, stop printing service, manage the local print queue, modify the printer entry, or add another printer.

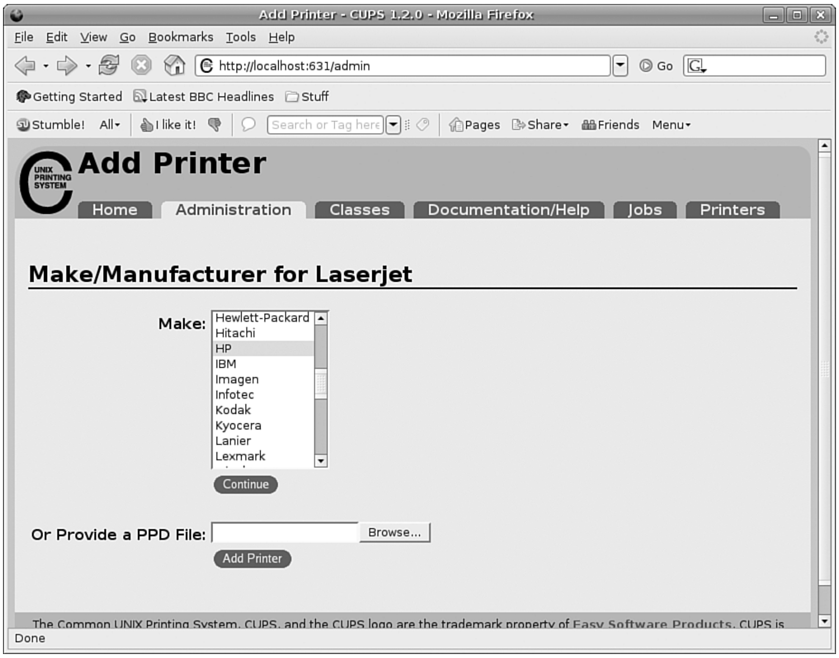
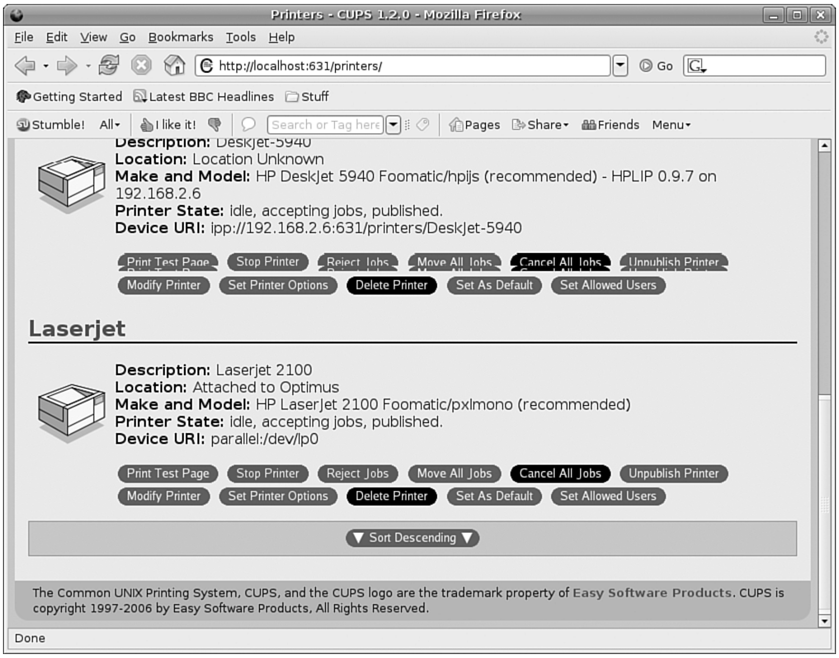

Add Printer - CUPS 1.2.0 - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:631/admin
Go
Getting Started
Latest BBC Headlines
Stuff
Stumble! All
I like it!
Search or Tag here
Pages Share Friends Menu
UNIX PRINTING SYSTEM
Add Printer
Home Administration Classes Documentation/Help Jobs Printers
Make/Manufacturer for Laserjet
Make:
Hewlett-Packard
Hitachi
HP
IBM
Imagen
Infotec
Kodak
Kyocera
Lanier
Lexmark
Continue
Or Provide a PPD File:
Browse...
Add Printer
The Common UNIX Printing System, CUPS, and the CUPS logo are the trademark property of Easy Software Products. CUPS is
Done

FIGURE 16.12 Select a printer make when creating a new queue.

CUPS offers many additional features and after it is installed, configured, and running, provides transparent traditional UNIX printing support for Ubuntu.

NOTE

To learn more about CUPS and to get a basic overview of the system, browse to http://www.cups.org/.


Description: DeskJet-5940
Location: Location Unknown
Make and Model: HP DeskJet 5940 Foomatic/hpijs (recommended) - HPLIP 0.9.7 on 192.168.2.6
Printer State: idle, accepting jobs, published.
Device URI: ipp://192.168.2.6:631/printers/DeskJet-5940

Laserjet

Description: Laserjet 2100
Location: Attached to Optimus
Make and Model: HP LaserJet 2100 Foomatic/pxlmono (recommended)
Printer State: idle, accepting jobs, published.
Device URI: parallel:/dev/lp0

The Common UNIX Printing System, CUPS, and the CUPS logo are the trademark property of Easy Software Products. CUPS is copyright 1997-2006 by Easy Software Products, All Rights Reserved.

FIGURE 16.13 Manage printers easily using the CUPS Printer page.

Avoiding Printer Support Problems

Troubleshooting printer problems can prove frustrating, especially if you find that your new printer is not working properly with Linux. Keep in mind, however, that nearly all printers on the market today work with Linux. That said, some vendors have higher batting averages in the game of supporting Linux. If you care to see a scorecard, browse to http://www.linuxprinting.org/vendors.html.

All-in-One (Print/Fax/Scan) Devices

Problematic printers, or printing devices that might or might not work with Ubuntu, include multifunction (or *all-in-one*) printers that combine scanning, faxing, and printing services. You should research any planned purchase and avoid any vendor unwilling to support Linux with drivers or development information.

One shining star in the field of Linux support for multifunction printers is the HP support of the HP OfficeJet Linux driver project at http://hpoj.sourceforge.net/. Printing and scanning are supported on many models, with fax support in development.

Using USB and Legacy Printers

Other problems can arise because of a lack of a printer's USB vendor and device ID information—a problem shared by some USB scanners under Linux. For information regarding USB printer support, check with the Linux printing folks (at the URL in the start of this section) or with the Linux USB project at http://www.linux-usb.org/.

Although many newer printers require a *universal serial bus (USB)* port, excellent support still exists for legacy parallel-port (IEEE-1284) printers with Linux, enabling sites to continue to use older hardware. You can take advantage of Linux workarounds to set up printing even if the host computer does not have a traditional parallel printer port or if you want to use a newer USB printer on an older computer.

For example, to host a parallel-port-based printer on a USB-only computer, attach the printer to the computer using an inexpensive USB-to-parallel converter. USB-to-parallel converters typically provide a Centronics connector; one end of that connector is plugged in to the older printer, and the other end is plugged in to a USB connector. The USB connector is then plugged in to your hub, desktop, or notebook USB port. On the other hand, you can use an add-on PCI card to add USB support for printing (and other devices) if the legacy computer does not have a built-in USB port. Most PCI USB interface cards add at least two ports, and you can chain devices via a hub.

Related Ubuntu and Linux Commands

The following commands help you manage printing services:

`accept`—Controls print job access to the CUPS server via the command line

`cancel`—Cancels a print job from the command line

`disable`—Control printing from the command line

`enable`—Controls CUPS printers

`lp`—Sends a specified file to the printer and allows control of the prince service

`lpc`—Displays the status of printers and print service at the console

`lpq`—Views print queues (pending print jobs) at the console

`lprm`—Removes print jobs from the print queue via the command line

`lpstat`—Displays printer and server status

Reference

- http://www.linuxprinting.org/—Browse here for specific drivers and information about USB and other types of printers.
- http://www.hp.com/wwsolutions/linux/products/printing_imaging/index.html—Short but definitive information from HP regarding printing product support under Linux.

- http://www.linuxdoc.org/HOWTO/Printing-HOWTO/—Grant Taylor's Printing HOWTO, with information on using various print services under Linux.
- http://www.cups.org/—A comprehensive repository of CUPS software, including versions for Red Hat Linux.
- http://www.pwg.org/ipp/—Home page for the Internet Printing Protocol standards.
- http://www.linuxprinting.org/cups-doc.html—Information about CUPS.
- http://www.cs.wisc.edu/~ghost/—Home page for the Ghostscript interpreter.
- http://www.samba.org/—Base entry point for getting more information about Samba and using the SMB protocol with Linux, UNIX, Mac OS, and other operating systems.
- In addition, an excellent book on Samba to help you learn more is *Samba Unleashed* (Sams Publishing, 2000, ISBN: 0-672-31862-8).
- If you are after more recent coverage of Samba, we would also recommend *Using Samba, 3rd Edition* (O'Reilly & Associates, 2007, ISBN: 0-596-00769-8).

This page intentionally left blank

CHAPTER 17
Apache Web Server Management

IN THIS CHAPTER

This chapter covers the configuration and management of the Apache web server. The chapter includes an overview of some of the major components of the server and discussions of text-based and graphical server configuration. You will see how to start, stop, and restart Apache using the command line. The chapter begins with some introductory information about this popular server and then shows you how to install, configure, and start using Apache.

About the Apache Web Server

Apache is the most widely used web server on the Internet today, according to a Netcraft survey of active websites in June 2007, which is shown in Table 17.1.

Note that these statistics do not reflect Apache's use on internal networks, known as *intranets*.

The name *Apache* appeared during the early development of the software because it was "a patchy" server, made up of patches for the freely available source code of the NCSA HTTPd web server. For a while after the NCSA HTTPd project was discontinued, a number of people wrote a variety of patches for the code, to either fix bugs or add features they wanted. A lot of this code was floating around and people were freely sharing it, but it was completely unmanaged.

After a while, Brian Behlendorf and Cliff Skolnick set up a centralized repository of these patches, and the Apache project was born. The project is still composed of a small core group of programmers, but anyone is welcome to submit patches to the group for possible inclusion in the code.

TABLE 17.1 Netcraft Survey Results (June 2007)

Web Server	Number	Percentage
Apache	65,588,298	53.76%
Microsoft*	38,836,030	31.83%
Google	4,872,765	3.99%
SunONE	2,273,173	1.86%
lighttpd	1,470,930	1.21%

**All web server products*

There has been a surge of interest in the Apache project over the past several years, partially buoyed by a new interest in open source on the part of enterprise-level information services. It's also due in part to crippling security flaws found in Microsoft's Internet Information Services (IIS); the existence of malicious web task exploits; and operating system and networking vulnerabilities to the now-infamous Code Red, Blaster, and Nimda worms. IBM made an early commitment to support and use Apache as the basis for its web offerings and has dedicated substantial resources to the project because it makes more sense to use an established, proven web server.

In mid-1999, The Apache Software Foundation was incorporated as a nonprofit company. A board of directors, who are elected on an annual basis by the ASF members, oversees the company. This company provides a foundation for several open-source software development projects, including the Apache Web Server project.

The best places to find out about Apache are the Apache Group's website, http://www.apache.org/, and the Apache Week website, http://www.apacheweek.com/, where you can subscribe to receive Apache Week by email to keep up on the latest developments in the project, keep abreast of security advisories, and research bug fixes.

TIP

You'll find an overview of Apache in its frequently asked questions (FAQs) at http://httpd.apache.org/docs-2.0/faq/. In addition to extensive online documentation, you'll also find the complete documentation for Apache in the HTML directory of your Apache server. You can access this documentation by looking at http://localhost/manual/index.html on your new Ubuntu system with one of the web browsers included on your system. You'll need to have Apache running on your system!

Ubuntu ships with Apache 2.0, and the server (named `apache2`) is included on this book's CD-ROMs and DVD. You can obtain the latest version of Apache as a package file from an Ubuntu FTP server, through Synaptic, or by getting the source code from the Apache website and, in true Linux tradition, build it for yourself.

To determine the version of Apache included with your system, use the web server's `-V` command-line option like this:

```
$ /usr/sbin/apache2 -V
Server version: Apache/2.0.50
Server built:   Jun 29 2004 11:11:55
Server's Module Magic Number: 20020903:8
Architecture:   32-bit
Server compiled with....
```

The output displays the version number, build date and time, platform, and various options used during the build. You can use the `-v` option to see terser version information.

Installing the Apache Server

You can install Apache through APT or build it yourself from source code. The Apache source builds on just about any UNIX-like operating system and on Win32.

If you are about to install a new version of Apache, you should shut down the old server. Even if it's unlikely that the old server will interfere with the installation procedure, shutting it down ensures that there will be no problems. If you do not know how to stop Apache, see the "Starting and Stopping Apache" section later in this chapter.

Installing with APT

You can find the Apache package on the Ubuntu installation media, on the Ubuntu FTP server, or at one of its many mirror sites. Updated packages usually contain important bug and security fixes. When an updated version is released, install it as quickly as possible to keep your system secure.

NOTE

Check the Apache site for security reports. Browse to http://httpd.apache.org/security_report.html for links to security vulnerabilities for Apache 1.3, 2.0, and 2.2. Subscribe to a support list or browse through up-to-date archives of all Apache mailing lists at http://httpd.apache.org/mail/ (for various articles) or http://httpd.apache.org/lists.html (for comprehensive and organized archives).

CAUTION

You should be wary of installing experimental packages, and never install them on production servers (that is, servers used in "real life"). Very carefully test the packages beforehand on a host that is not connected to a network!

The easiest way to use APT is through Synaptic, which is under the System > Administration menu. Search for apache2 and select it, and Synaptic will add the required dependencies.

Alternatively you can install Apache through a .deb package with the command-line dpkg tool by typing the following:

```
dpkg -i latest_apache.deb
```

where *`latest_apache.deb`* is the name of the latest Apache package. For more information on installing packages with APT and dpkg, refer to Chapter 31, “Managing Software.”

The Apache package installs files in the following directories:

- `/etc/apache2`—This directory contains the Apache configuration file, `apache2.conf`.
- `/etc/init.d/`—The tree under this directory contains the system startup scripts. The Apache package installs a startup script named `apache2` for the web server under the `/etc/init.d` directory. This script, which you can use to start and stop the server from the command line, also automatically starts and stops the server when the computer is halted, started, or rebooted.
- `/var/www`—The package installs the default server icons, Common Gateway Interface (CGI) programs, and HTML files in this location. If you want to keep web content elsewhere, you can do so by making the appropriate changes in the server configuration files.
- `/var/www/manual/`—If you’ve installed the `apache-manual` package, you’ll find a copy of the Apache documentation in HTML format here. You can access it with a web browser by going to http://localhost/manual/.
- `/usr/share/man`—Ubuntu’s Apache package also contains manual pages, which are placed underneath this directory. For example, the `apache2` man page is in section 8 of the `man` directory.
- `/usr/sbin`—The executable programs are placed in this directory. This includes the server executable itself, as well as various utilities.
- `/usr/bin`—Some of the utilities from the Apache package are placed here—for example, the `htpasswd` program, which is used for generating authentication password files.
- `/var/log/apache2`—The server log files are placed in this directory. By default, there are two important log files (among several others): `access_log` and `error_log`. However, you can define any number of custom logs containing a variety of information. See the section “Logging,” later in this chapter, for more detail.

When Apache is being run, it also creates the file `apache2.pid`, containing the process ID of Apache’s parent process in the `/var/run/` directory.

NOTE

If you are upgrading to a newer version of Apache, APT does not write over your current configuration files.

Building the Source Yourself

You can download the source directly from http://www.apache.org/. The latest version at the time of this writing (2.2.0) is a 6MB compressed tape archive, and the latest pre-2.0 version of Apache is 1.3.34. Although many sites continue to use the older version (for script and other compatibility reasons), many new sites are migrating to or starting out using the latest stable version.

After you have the `tar` file, you must unroll it in a temporary directory, such as `/tmp`. Unrolling this `tar` file creates a directory called `apache_`*`version_number`*, where *`version_number`* is the version you have downloaded (for example, `apache_1.3.34`).

There are two ways to compile the source—the old, familiar way (at least, to those of us who have been using Apache for many years) by editing Makefile templates, and the new, easy way using a `configure` script. You will first see how to build Apache from source the easy way. The `configure` script offers a way to have the source software automatically configured according to your system. However, manually editing the configuration files before building and installing Apache provides more control over where the software is installed and which capabilities or features are built in to Apache.

TIP

As with many software packages distributed in source code form for Linux and other UNIX-like operating systems, extracting the source code results in a directory that contains a `README` and an `INSTALL` file. Be sure to peruse the `INSTALL` file before attempting to build and install the software.

Using `./configure` to Build Apache

To build Apache the easy way, run the `./configure` script in the directory just created. You can provide it with a `--prefix` argument to install it in a directory other than the default, which is `/usr/local/apache/`. Use this command:

```
# ./configure --prefix=/preferred/directory/
```

This generates the Makefile that is used to compile the server code.

Next, type **`make`** to compile the server code. After the compilation is complete, type **`make install`** as root to install the server. You can now configure the server via the configuration files. See the "Runtime Server Configuration Settings" section for more information.

TIP

A safer way to install a new version of Apache from source is to use the `ln` command to create symbolic links of the existing file locations (listed in the "Installing with APT" section earlier in this chapter) to the new locations of the files. This method is safer because the default install locations are different from those used when the package installs the files. Failure to use this installation method could result in your web server process not being started automatically at system startup.

Another safe way to install a new version of Apache is to first back up any important configuration directories and files (such as `/etc/apache2`) and then use the `apt-get` command to remove the server. You can then install and test your new version and, if needed, easily restore your original server and settings.

It is strongly recommended that you use Ubuntu's version of Apache until you really know what happens at system startup. No "uninstall" option is available when installing Apache from source!

Apache File Locations After a Build and Install

Files are placed in various subdirectories of `/usr/local/apache` (or whatever directory you specified with the `--prefix` parameter) if you build the server from source. Before version 1.3.4, files were placed in `/usr/local/etc/httpd`.

The following is a list of the directories used by Apache, as well as brief comments on their usage:

- `/usr/local/apache/conf`—This contains several subdirectories and the Apache configuration file, `httpd.conf`. Debian-based systems such as Ubuntu often rename this to `apache2.conf`, so your mileage may vary. See the section "Editing `apache2.conf`" later in this chapter to learn more about configuration files.
- `/usr/local/apache`—The `cgi-bin`, `icons`, and `htdocs` subdirectories contain the CGI programs, standard icons, and default HTML documents, respectively.
- `/usr/local/apache/bin`—The executable programs are placed in this directory.
- `/usr/local/apache/logs`—The server log files are placed in this directory. By default, there are two log files—`access_log` and `error_log`—but you can define any number of custom logs containing a variety of information (see the "Logging" section later in this chapter). The default location for Apache's logs as installed by Ubuntu is `/var/log/apache2`.

A Quick Guide to Getting Started with Apache

Setting up, testing a web page, and starting Apache using Ubuntu can be accomplished in just a few steps. First, make sure that Apache is installed on your system. Either select it during installation or install the server and related package files (refer to Chapter 31 if you need to install the server software).

Next, set up a home page for your system by editing (as `root`) the file named `index.html` under the `/var/ www` directory on your system. Make a backup copy of the original page or www directory before you begin so you can restore your web server to its default state if necessary.

```
Start Apache through the Services window (under System, Administration from
the menu bar), making sure to enable "Web Server".
```

You can also use the `apache2` script under the `/etc/init.d/` directory, like this:

```
sudo /etc/init.d/apache2 start
```

You can then check your home page by running a favorite browser and using `localhost`, your system's hostname, or its Internet Protocol (IP) address in the URL. For example, with the `links` text browser, use a command line like this:

```
# links http://localhost/
```

For security reasons, you should not start and run Apache as `root` if your host is connected to the Internet or a company intranet. Fortunately, Apache is set to run as the user and group `www-data` no matter how it is started (by the `User` and `Group` settings in `/etc/apache2/apache2.conf`). Despite this safe default, Apache should be started and managed by the user named `apache`, defined in `/etc/passwd` as

```
www-data:x:33:33:www-data:/var/www:/sbin/nologin
```

After you are satisfied with your website, use the Services configuration dialog to ensure that Apache is started.

Starting and Stopping Apache

At this point, you have installed your Apache server with its default configuration. Ubuntu provides a default home page named `index.html` as a test under the `/var/www/` directory. The proper way to run Apache is to set system initialization to have the server run after booting, network configuration, and any firewall configuration. See Chapter 11, "Automating Tasks," for more information about how Ubuntu boots.

It is time to start it up for the first time. The following sections show how to either start and stop Apache or configure Ubuntu to start or not start Apache when booting.

Starting the Apache Server Manually

You can start Apache from the command line of a text-based console or X terminal window, and you must have root permission to do so. The server daemon, `apache2`, recognizes several command-line options you can use to set some defaults, such as specifying where `apache2` reads its configuration directives. The Apache `apache2` executable also understands other options that enable you to selectively use parts of its configuration file, specify a different location of the actual server and supporting files, use a different configuration file (perhaps for testing), and save startup errors to a specific log. The `-v` option causes Apache to print its development version and quit. The `-V` option shows all the settings that were in effect when the server was compiled.

The `-h` option prints the following usage information for the server (assuming that you are running the command through sudo):

```
sudo apache2 -h
Usage: apache2 [-D name] [-d directory] [-f file]
          [-C "directive"] [-c "directive"]
          [-k start¦restart¦graceful¦stop]
          [-v] [-V] [-h] [-l] [-L] [-t]
```

```
Options:
  -D name           : define a name for use in <IfDefine name> directives
  -d directory      : specify an alternate initial ServerRoot
  -f file           : specify an alternate ServerConfigFile
  -C "directive"    : process directive before reading config files
  -c "directive"    : process directive after reading config files
  -e level          : show startup errors of level (see LogLevel)
  -E file           : log startup errors to file
  -v                : show version number
  -V                : show compile settings
  -h                : list available command line options (this page)
  -l                : list compiled in modules
  -L                : list available configuration directives
  -t -D DUMP_VHOSTS : show parsed settings (currently only vhost settings)
-t                : run syntax check for config files
```

Other options include listing Apache's *static modules*, or special, built-in independent parts of the server, along with options that can be used with the modules. These options are called *configuration directives* and are commands that control how a static module works. Note that Apache also includes nearly 50 *dynamic modules*, or software portions of the server that can be optionally loaded and used while the server is running.

The `-t` option is used to check your configuration files. It's a good idea to run this check before restarting your server, especially if you've made changes to your configuration files. Such tests are important because a configuration file error can result in your server shutting down when you try to restart it.

NOTE

When you build and install Apache from source, start the server manually from the command line through sudo (such as when testing). You do this for two reasons:

- The standalone server uses the default HTTP port (port 80), and only the superuser can bind to Internet ports that are lower than 1024.
- Only processes owned by `root` can change their UID and GID as specified by Apache's `User` and `Group` directives. If you start the server under another UID, it runs with the permissions of the user starting the process.

Note that although some of the following examples show how to start the server through sudo, you should do so only for testing after building and installing Apache.

Using `/etc/init.d/apache2`

Ubuntu uses the scripts in the `/etc/init.d` directory to control the startup and shutdown of various services, including the Apache web server. The main script installed for the Apache web server is `/etc/init.d/apache2`, although the actual work is done by the `apache2ctl` shell script included with Apache.

NOTE

/etc/init.d/apache2 is a shell script and is not the same as the Apache server located in /usr/sbin. That is, /usr/sbin/apache2 is the program executable file (the server); /etc/init.d/apache2 is a shell script that uses another shell script, apache2ctl, to control the server. See Chapter 11 for a description of some service scripts under /etc/init.d and how the scripts are used to manage services such as apache2.

You can use the /etc/init.d/apache2 script and the following options to control the web server:

- start—The system uses this option to start the web server during bootup. You, through sudo, can also use this script to start the server.
- stop—The system uses this option to stop the server gracefully. You should use this script, rather than the kill command, to stop the server.
- reload—You can use this option to send the HUP signal to the apache2 server to have it reread the configuration files after modification.
- restart—This option is a convenient way to stop and then immediately start the web server. If the apache2 server isn't running, it is started.
- condrestart—The same as the restart parameter, except that it restarts the apache2 server only if it is actually running.
- status—This option indicates whether the server is running; if it is, it provides the various PIDs for each instance of the server.

For example, to check on the status of your server, use the command

```
sudo /etc/init.d/apache2 status
```

This prints the following for me:

```
apache2 (pid 15997 1791 1790 1789 1788 1787 1786 1785 1784 1781) is running...
```

This indicates that the web server is running; in fact, 10 instances of the server are currently running in this configuration.

In addition to the previous options, the apache2 script also offers these features:

- help—Prints a list of valid options to the apache2 script (which are passed onto the server as if called from the command line).
- configtest—A simple test of the server's configuration, which reports Status OK if the setup is correct. You can also use apache2's -t option to perform the same test, like this:

  ```
  audo apache2 -t
  ```

- `fullstatus`—Displays a verbose status report.
- `graceful`—The same as the `restart` parameter, except that the `configtest` option is used first and open connections are not aborted.

TIP

Use the `reload` option if you are making many changes to the various server configuration files. This saves time when you are stopping and starting the server by having the system simply reread the configuration files.

Runtime Server Configuration Settings

At this point, the Apache server will run, but perhaps you want to change a behavior, such as the default location of your website's files. This section talks about the basics of configuring the server to work the way you want it to work.

Runtime configurations are stored in just one file—`apache2.conf`, which is found under the `/etc/apache2` directory. This configuration file can be used to control the default behavior of Apache, such as the web server's base configuration directory (`/etc/apache2`), the name of the server's process identification (PID) file (`/var/run/apache2.pid`), or its response timeout (300 seconds). Apache reads the data from the configuration file when started (or restarted). You can also cause Apache to reload configuration information with the command `/etc/init.d/apache2 reload`, which is necessary after making changes to its configuration file. (You learned how to accomplish this in the earlier section, "Starting and Stopping Apache.")

Runtime Configuration Directives

You perform runtime configuration of your server with configuration directives, which are commands that set options for the `apache2` daemon. The directives are used to tell the server about various options you want to enable, such as the location of files important to the server configuration and operation. Apache supports nearly 300 configuration directives using the following syntax:

```
directive option option...
```

Each directive is specified on a single line. See the following sections for some sample directives and how to use them. Some directives only set a value such as a filename, whereas others enable you to specify various options. Some special directives, called sections, look like HTML tags. Section directives are surrounded by angle brackets, such as `<directive>`. Sections usually enclose a group of directives that apply only to the directory specified in the section:

```
<Directory somedir/in/your/tree>
  directive option option
  directive option option
</Directory>
```

All sections are closed with a matching section tag that looks like this: `</directive>`. Note that section tags, like any other directives, are specified one per line.

TIP

After installing and starting Apache, you'll find an index of directives at http://localhost/manual/mod/directives.html.

Editing `apache2.conf`

Most of the default settings in the config file are okay to keep, particularly if you've installed the server in a default location and aren't doing anything unusual on your server. In general, if you do not understand what a particular directive is for, you should leave it set to the default value.

The following sections describe some of the configuration file settings you *might* want to change concerning operation of your server.

`ServerRoot`

The `ServerRoot` directive sets the absolute path to your server directory. This directive tells the server where to find all the resources and configuration files. Many of these resources are specified in the configuration files relative to the `ServerRoot` directory.

Your `ServerRoot` directive should be set to `/etc/apache2` if you installed the Ubuntu package or `/usr/local/apache` (or whatever directory you chose when you compiled Apache) if you installed from the source.

`Listen`

The `Listen` directive indicates on which port you want your server to run. By default, this is set to 80, which is the standard HTTP port number. You might want to run your server on another port—for example, when running a test server that you don't want people to find by accident. Do not confuse this with real security! See the "File System Authentication and Access Control" section for more information about how to secure parts of your web server.

`User` **and** `Group`

The `User` and `Group` directives should be set to the UID and group ID (GID) the server will use to process requests.

In Ubuntu, set these configurations to a user with few or no privileges. In this case, they're set to user `apache` and group `apache`—a user defined specifically to run Apache. If you want to use a different UID or GID, be aware that the server will run with the permissions of the user and group set here. That means in the event of a security breach, whether on the server or (more likely) in your own CGI programs, those programs will run with the assigned UID. If the server runs as `root` or some other privileged user, someone can exploit the security holes and do nasty things to your site. Always think in terms of the specified user running a command such as `rm -rf /` because that would wipe all files from your system. That should convince you that leaving `apache` as a user with no privileges is probably a good thing.

Instead of specifying the `User` and `Group` directives using names, you can specify them using the UID and GID numbers. If you use numbers, be sure that the numbers you specify correspond to the user and group you want and that they're preceded by the pound (`#`) symbol.

Here's how these directives look if specified by name:

```
User apache
Group apache
```

Here's the same specification by UID and GID:

```
User #48
Group #48
```

TIP

If you find a user on your system (other than root) with a UID and GID of `0`, your system has been compromised by a malicious user.

`ServerAdmin`

The `ServerAdmin` directive should be set to the address of the webmaster managing the server. This address should be a valid email address or alias, such as `webmaster@gnulix.org`, because this address is returned to a visitor when a problem occurs on the server.

`ServerName`

The `ServerName` directive sets the hostname the server will return. Set it to a fully qualified domain name (FQDN). For example, set it to `www.`*`your.domain`* rather than simply `www`. This is particularly important if this machine will be accessible from the Internet rather than just on your local network.

You don't need to set this unless you want a name other than the machine's canonical name returned. If this value isn't set, the server will figure out the name by itself and set it to its canonical name. However, you might want the server to return a friendlier address, such as `www.`*`your.domain`*. Whatever you do, `ServerName` should be a real domain name service (DNS) name for your network. If you're administering your own DNS, remember to add an alias for your host. If someone else manages the DNS for you, ask that person to set this name for you.

`DocumentRoot`

Set this directive to the absolute path of your document tree, which is the top directory from which Apache will serve files. By default, it's set to `/var/www/`. If you built the source code yourself, `DocumentRoot` is set to `/usr/local/apache/htdocs` (if you did not choose another directory when you compiled Apache).

`UserDir`

The `UserDir` directive disables or enables and defines the directory (relative to a local user's home directory) where that user can put public HTML documents. It is relative because each user has her own HTML directory. This setting is disabled by default but can be enabled to store user web content under any directory.

The default setting for this directive, if enabled, is `public_html`. Each user can create a directory called `public_html` under her home directory, and HTML documents placed in that directory are available as `http://servername/~`*`username`*, where *`username`* is the username of the particular user.

`DirectoryIndex`

The `DirectoryIndex` directive indicates which file should be served as the index for a directory, such as which file should be served if the URL `http://`*`servername`*`/`*`_SomeDirectory`*`/` is requested.

It is often useful to put a list of files here so that if `index.html` (the default value) isn't found, another file can be served instead. The most useful application of this is to have a CGI program run as the default action in a directory. If you have users who make their web pages on Windows, you might want to add `index.htm` as well. In that case, the directive would look like `DirectoryIndex index.html index.cgi index.htm`.

Apache Multiprocessing Modules

Apache version 2.0 and greater now uses a new internal architecture supporting multiprocessing modules (MPMs). These modules are used by the server for a variety of tasks, such as network and process management, and are compiled into Apache. MPMs enable Apache to work much better on a wider variety of computer platforms, and they can help improve server stability, compatibility, and scalability.

Apache can use only one MPM at any time. These modules are different from the base set included with Apache (see the "Apache Modules" section later in this chapter) but are used to implement settings, limits, or other server actions. Each module in turn supports numerous additional settings, called *directives*, which further refine server operation.

The internal MPM modules relevant for Linux include

- `mpm_common`—A set of 20 directives common to all MPM modules
- `prefork`—A nonthreaded, preforking web server that works similar to earlier (1.3) versions of Apache
- `worker`—Provides a hybrid multiprocess multithreaded server

MPM enables Apache to be used on equipment with fewer resources yet still handle massive numbers of hits and provide stable service. The `worker` module provides directives to control how many simultaneous connections your server can handle.

NOTE

Other MPMs are available for Apache related to other platforms, such as `mpm_netware` for NetWare hosts and `mpm_winnt` for NT platforms. An MPM named `perchild`, which provides user ID assignment to selected daemon processes, is under development. For more information, browse to the Apache Software Foundation's home page at http://www.apache.org.

Using `.htaccess` Configuration Files

Apache also supports special configuration files, known as `.htaccess` files. Almost any directive that appears in `apache2.conf` can appear in a `.htaccess` file. This file, specified in the `AccessFileName` directive in `apache2.conf` sets configurations on a per-directory (usually in a user directory) basis. As the system administrator, you can specify both the name of this file and which of the server configurations can be overridden by the contents of this file. This is especially useful for sites in which there are multiple content providers and you want to control what these people can do with their space.

To limit which server configurations the `.htaccess` files can override, use the `AllowOverride` directive. `AllowOverride` can be set globally or per directory. For example, in your `apache2.conf` file, you could use the following:

```
# Each directory to which Apache has access can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# permissions.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
```

`Options` **Directives**

To configure which configuration options are available to Apache by default, you must use the `Options` directive. `Options` can be `None`; `All`; or any combination of `Indexes`, `Includes`, `FollowSymLinks`, `ExecCGI`, and `MultiViews`. `MultiViews` isn't included in `All` and must be specified explicitly. These options are explained in Table 17.2.

TABLE 17.2 Switches Used by the `Options` Directive

Switch	**Description**
`None`	None of the available options are enabled for this directory.
`All`	All the available options, except for `MultiViews`, are enabled for this directory.
`Indexes`	In the absence of an `index.html` file or another `DirectoryIndex` file, a listing of the files in the directory is generated as an HTML page for display to the user.
`Includes`	Server-side includes (SSIs) are permitted in this directory. This can also be written as `IncludesNoExec` if you want to allow includes but don't want to allow the `exec` option in them. For security reasons, this is usually a good idea in directories over which you don't have complete control, such as `UserDir` directories.
`FollowSymLinks`	Allows access to directories that are symbolically linked to a document directory. You should never set this globally for the whole server and only rarely for individual directories. This option is a potential security risk because it allows web users to escape from the document directory and could potentially allow them access to portions of your file system where you really don't want people poking around.
`ExecCGI`	CGI programs are permitted in this directory, even if it is not a directory defined in the `ScriptAlias` directive.
`MultiViews`	This is part of the `mod_negotiation` module. When a client requests a document that can't be found, the server tries to figure out which document best suits the client's requirements. See http://localhost/manuals/mod/_mod_negotiation.html for your local copy of the Apache documentation.

NOTE

These directives also affect all subdirectories of the specified directory.

`AllowOverrides` **Directives**

The `AllowOverrides` directives specify which configuration options `.htaccess` files can override. You can set this directive individually for each directory. For example, you can have different standards about what can be overridden in the main document `root` and in `UserDir` directories.

This capability is particularly useful for user directories, where the user does not have access to the main server configuration files.

`AllowOverrides` can be set to `All` or any combination of `Options`, `FileInfo`, `AuthConfig`, and `Limit`. These options are explained in Table 17.3.

TABLE 17.3 Switches Used by the `AllowOverrides` Directive

Switch	Description
`Options`	The `.htaccess` file can add options not listed in the `Options` directive for this directory.
`FileInfo`	The `.htaccess` file can include directives for modifying document type information.
`AuthConfig`	The `.htaccess` file might contain authorization directives.
`Limit`	The `.htaccess` file might contain `allow`, `deny`, and `order` directives.

File System Authentication and Access Control

You're likely to include material on your website that isn't supposed to be available to the public. You must be able to lock out this material from public access and provide designated users with the means to unlock the material. Apache provides two methods for accomplishing this type of access: authentication and authorization. You can use different criteria to control access to sections of your website, including checking the client's IP address or hostname, or requiring a username and password. This section briefly covers some of these methods.

CAUTION

Allowing individual users to put web content on your server poses several important security risks. If you're operating a web server on the Internet rather than on a private network, you should read the WWW Security FAQ at http://www.w3.org/Security/Faq/www-security-faq.html.

Restricting Access with `allow` and `deny`

One of the simplest ways to limit access to website material is to restrict access to a specific group of users, based on IP addresses or hostnames. Apache uses the `allow` and `deny` directives to accomplish this.

Both directives take an address expression as a parameter. The following list provides the possible values and use of the address expression:

- `all` can be used to affect all hosts.
- A hostname or domain name, which can either be a partially or a fully qualified domain name; for example, `test.gnulix.org` or `gnulix.org`.
- An IP address, which can be either full or partial; for example, `212.85.67` or `212.85.67.66`.

- A network/netmask pair, such as `212.85.67.0/255.255.255.0`.
- A network address specified in classless inter-domain routing (CIDR) format; for example, `212.85.67.0/24`. This is the CIDR notation for the same network and netmask that were used in the previous example.

If you have the choice, it is preferable to base your access control on IP addresses rather than hostnames. Doing so results in faster performance because no name lookup is necessary—the IP address of the client is included with each request.

You also can use `allow` and `deny` to provide or deny access to website material based on the presence or absence of a specific environment variable. For example, the following statement denies access to a request with a context that contains an environment variable named `NOACCESS`:

```
deny from env=NOACCESS
```

The default behavior of Apache is to apply all the `deny` directives first and then check the `allow` directives. If you want to change this order, you can use the `order` statement. Apache might interpret this statement in three different ways:

- `Order deny,allow`—The `deny` directives are evaluated before the `allow` directives. If a host is not specifically denied access, it is allowed to access the resource. This is the default ordering if nothing else is specified.
- `Order allow,deny`—All `allow` directives are evaluated before `deny` directives. If a host is not specifically allowed access, it is denied access to the resource.
- `Order mutual-failure`—Only hosts that are specified in an `allow` directive and at the same time do not appear in a `deny` directive are allowed access. If a host does not appear in either directive, it is not granted access.

Consider this example. Suppose you want to allow only persons from within your own domain to access the `server-status` resource on your web. If your domain were named `gnulix.org`, you could add these lines to your configuration file:

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from gnulix.org
</Location>
```

Authentication

Authentication is the process of ensuring that visitors really are who they claim to be. You can configure Apache to allow access to specific areas of web content only to clients who can authenticate their identity. There are several methods of authentication in Apache; Basic Authentication is the most common (and the method discussed in this chapter).

Under Basic Authentication, Apache requires a user to supply a username and a password to access the protected resources. Apache then verifies that the user is allowed to access the resource in question. If the username is acceptable, Apache verifies the password. If the password also checks out, the user is authorized and Apache serves the request.

HTTP is a stateless protocol; each request sent to the server and each response is handled individually, and not in an intelligent fashion. Therefore, the authentication information must be included with each request. That means each request to a password-protected area is larger and therefore somewhat slower. To avoid unnecessary system use and delays, protect only those areas of your website that absolutely need protection.

To use Basic Authentication, you need a file that lists which users are allowed to access the resources. This file is composed of a plain text list containing name and password pairs. It looks very much like the `/etc/passwd` user file of your Linux system.

CAUTION

Do not use `/etc/passwd` as a user list for authentication. When you're using Basic Authentication, passwords and usernames are sent as base64-encoded text from the client to the server—which is just as readable as plain text. The username and password are included in each request that is sent to the server. So, anyone who might be snooping on Net traffic would be able to get this information!

To create a user file for Apache, use the `htpasswd` command. This is included with the Apache package. If you installed using the packages, it is in `/usr/bin`. Running `htpasswd` without any options produces the following output:

```
Usage:
        htpasswd [-cmdps] passwordfile username
        htpasswd -b[cmdps] passwordfile username password

        htpasswd -n[mdps] username
        htpasswd -nb[mdps] username password
 -c  Create a new file.
 -n  Don't update file; display results on stdout.
 -m  Force MD5 encryption of the password.
 -d  Force CRYPT encryption of the password (default).
 -p  Do not encrypt the password (plaintext).
 -s  Force SHA encryption of the password.
 -b  Use the password from the command line rather than prompting for it.
 -D  Delete the specified user.
On Windows, TPF and NetWare systems the '-m' flag is used by default.
On all other systems, the '-p' flag will probably not work.
```

As you can see, it isn't a very difficult command to use. For example, to create a new user file named `gnulixusers` with a user named `wsb`, you need to do something like this:

```
sudo htpasswd -c gnulixusers wsb
```

You would then be prompted for a password for the user. To add more users, you would repeat the same procedure, only omitting the `-c` flag.

You can also create user group files. The format of these files is similar to that of `/etc/groups`. On each line, enter the group name, followed by a colon, and then list all users, with each user separated by spaces. For example, an entry in a user group file might look like this:

```
gnulixusers: wsb pgj jp ajje nadia rkr hak
```

Now that you know how to create a user file, it's time to look at how Apache might use this to protect web resources.

To point Apache to the user file, use the `AuthUserFile` directive. `AuthUserFile` takes the file path to the user file as its parameter. If the file path is not absolute—that is, beginning with a `/`—it is assumed that the path is relative to the `ServerRoot`. Using the `AuthGroupFile` directive, you can specify a group file in the same manner.

Next, use the `AuthType` directive to set the type of authentication to be used for this resource. Here, the type is set to `Basic`.

Now you need to decide to which realm the resource will belong. Realms are used to group different resources that will share the same users for authorization. A realm can consist of just about any string. The realm is shown in the Authentication dialog box on the user's web browser. Therefore, you should set the realm string to something informative. The realm is defined with the `AuthName` directive.

Finally, state which type of user is authorized to use the resource. You do this with the `require` directive. The three ways to use this directive are as follows:

- If you specify `valid-user` as an option, any user in the user file is allowed to access the resource (that is, provided she also enters the correct password).
- You can specify a list of users who are allowed access with the `users` option.
- You can specify a list of groups with the `group` option. Entries in the group list, as well as the user list, are separated by a space.

Returning to the `server-status` example you saw earlier, instead of letting users access the `server-status` resource based on hostname, you can require the users to be authenticated to access the resource. You can do so with the following entry in the configuration file:

```
<Location /server-status>
    SetHandler server-status
    AuthType Basic
    AuthName "Server status"
    AuthUserFile "gnulixusers"
    Require valid-user
</Location>
```

Final Words on Access Control

If you have host-based as well as user-based access protection on a resource, the default behavior of Apache is to require the requester to satisfy both controls. But assume that you want to mix host-based and user-based protection and allow access to a resource if either method succeeds. You can do so using the `satisfy` directive. You can set the `satisfy` directive to `All` (this is the default) or `Any`. When set to `All`, all access control methods must be satisfied before the resource is served. If `satisfy` is set to `Any`, the resource is served if any access condition is met.

Here's another access control example, again using the previous `server-status` example. This time, you combine access methods so all users from the `Gnulix` domain are allowed access and those from outside the domain must identify themselves before gaining access. You can do so with the following:

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from gnulix.org
    AuthType Basic
    AuthName "Server status"
    AuthUserFile "gnulixusers"
    Require valid-user
    Satisfy Any
</Location>
```

There are more ways to protect material on your web server, but the methods discussed here should get you started and will probably be more than adequate for most circumstances. Look to Apache's online documentation for more examples of how to secure areas of your site.

Apache Modules

The Apache core does relatively little; Apache gains its functionality from modules. Each module solves a well-defined problem by adding necessary features. By adding or removing modules to supply the functionality you want Apache to have, you can tailor Apache server to suit your exact needs.

Nearly 50 core modules are included with the basic Apache server. Many more are available from other developers. The Apache Module Registry is a repository for add-on modules for Apache, and it can be found at http://modules.apache.org/. The modules are stored in the `/usr/lib/apache2/modules` directory (your list might look different):

```
mod_access.so        mod_cern_meta.so  mod_log_config.so     mod_setenvif.so
mod_actions.so       mod_cgi.so        mod_mime_magic.so     mod_speling.so
mod_alias.so         mod_dav_fs.so     mod_mime.so           mod_ssl.so
mod_asis.so          mod_dav.so        mod_negotiation.so    mod_status.so
mod_auth_anon.so     mod_dir.so        mod_perl.so           mod_suexec.so
mod_auth_dbm.so      mod_env.so        mod_proxy_connect.so  mod_unique_id.so
mod_auth_digest.so   mod_expires.so    mod_proxy_ftp.so      mod_userdir.so
mod_auth_mysql.so    mod_headers.so    mod_proxy_http.so     mod_usertrack.so
mod_auth_pgsql.so    mod_imap.so       mod_proxy.so          mod_vhost_alias.so
mod_auth.so          mod_include.so    mod_python.so         mod_autoindex.so
mod_info.so          mod_rewrite.so
```

Each module adds new directives that can be used in your configuration files. As you might guess, there are far too many extra commands, switches, and options to describe them all in this chapter. The following sections briefly describe a subset of those modules available with Ubuntu's Apache installation.

mod_access

`mod_access` controls access to areas on your web server based on IP addresses, hostnames, or environment variables. For example, you might want to allow anyone from within your own domain to access certain areas of your web. Refer to the "File System Authentication and Access Control" section for more information.

mod_alias

`mod_alias` manipulates the URLs of incoming HTTP requests, such as redirecting a client request to another URL. It also can map a part of the file system into your web hierarchy. For example,

```
Alias /images/ /home/wsb/graphics/
```

fetches contents from the `/home/wsb/graphics` directory for any URL that starts with `/images/`. This is done without the client knowing anything about it. If you use a redirection, the client is instructed to go to another URL to find the requested content. More advanced URL manipulation can be accomplished with `mod_rewrite`.

mod_asis

`mod_asis` is used to specify, in fine detail, all the information to be included in a response. This completely bypasses any headers Apache might have otherwise added to the response. All files with an `.asis` extension are sent straight to the client without any changes.

As a short example of the use of mod_asis, assume you've moved content from one location to another on your site. Now you must inform people who try to access this resource that it has moved, as well as automatically redirect them to the new location. To provide this information and redirection, you can add the following code to a file with a .asis extension:

```
Status: 301 No more old stuff!
Location: http://gnulix.org/newstuff/
Content-type: text/html

<HTML>
 <HEAD>
  <TITLE>We've moved...</TITLE>
 </HEAD>
 <BODY>
   <P>We've moved the old stuff and now you'll find it at:</P>
   <A HREF="http://gnulix.org/newstuff/">New stuff</A>!.
 </BODY>
</HTML>
```

mod_auth

mod_auth uses a simple user authentication scheme, referred to as Basic Authentication, which is based on storing usernames and encrypted passwords in a text file. This file looks very much like UNIX's /etc/passwd file and is created with the htpasswd command. Refer to the "File System Authentication and Access Control" section earlier in this chapter for more information about this subject.

mod_auth_anon

The mod_auth_anon module provides anonymous authentication similar to that of anonymous FTP. The module enables you to define user IDs of those who are to be handled as guest users. When such a user tries to log on, he is prompted to enter his email address as his password. You can have Apache check the password to ensure that it's a (more or less) proper email address. Basically, it ensures that the password contains an @ character and at least one . character.

mod_auth_dbm

mod_auth_dbm uses Berkeley DB files instead of text for user authentication files.

mod_auth_digest

An extension of the basic mod_auth module, instead of sending the user information in plain text, mod_auth_digestis sent via the MD5 Digest Authentication process. This authentication scheme is defined in RFC 2617. Compared to using Basic Authentication, this is a much more secure way of sending user data over the Internet. Unfortunately, not all web browsers support this authentication scheme.

To create password files for use with `mod_auth_dbm`, you must use the `htdigest` utility. It has more or less the same functionality as the `htpasswd` utility. See the man page of `htdigest` for further information.

mod_autoindex

The `mod_autoindex` module dynamically creates a file list for directory indexing. The list is rendered in a user-friendly manner similar to those lists provided by FTP's built-in `ls` command.

mod_cgi

`mod_cgi` allows execution of CGI programs on your server. CGI programs are executable files residing in the `/var/www/cgi-bin` directory and are used to dynamically generate data (usually HTML) for the remote browser when requested.

mod_dir and mod_env

The `mod_dir` module is used to determine which files are returned automatically when a user tries to access a directory. The default is `index.html`. If you have users who create web pages on Windows systems, you should also include `index.htm`, like this:

```
DirectoryIndex index.html index.htm
```

`mod_env` controls how environment variables are passed to CGI and SSI scripts.

mod_expires

`mod_expires` is used to add an expiration date to content on your site by adding an `Expires` header to the HTTP response. Web browsers or cache servers won't cache expired content.

mod_headers

`mod_headers` is used to manipulate the HTTP headers of your server's responses. You can replace, add, merge, or delete headers as you see fit. The module supplies a `Header` directive for this. Ordering of the `Header` directive is important. A `set` followed by an `unset` for the same HTTP header removes the header altogether. You can place `Header` directives almost anywhere within your configuration files. These directives are processed in the following order:

1. Core server
2. Virtual host
3. `<Directory>` and `.htaccess` files

4. *`<Location>`*
5. *`<Files>`*

`mod_include`

`mod_include` enables the use of server-side includes on your server, which were quite popular before PHP took over this part of the market.

`mod_info` **and** `mod_log_config`

`mod_info` provides comprehensive information about your server's configuration. For example, it displays all the installed modules, as well as all the directives used in its configuration files.

`mod_log_config` defines how your log files should look. See the "Logging" section for further information about this subject.

`mod_mime` **and** `mod_mime_magic`

The `mod_mime` module tries to determine the MIME type of files from their extensions.

The `mod_mime_magic` module tries to determine the MIME type of files by examining portions of their content.

`mod_negotiation`

Using the `mod_negotiation` module, you can select one of several document versions that best suits the client's capabilities. There are several options to select which criteria to use in the negotiation process. You can, for example, choose among different languages, graphics file formats, and compression methods.

`mod_proxy`

`mod_proxy` implements proxy and caching capabilities for an Apache server. It can proxy and cache FTP, CONNECT, HTTP/0.9, and HTTP/1.0 requests. This isn't an ideal solution for sites that have a large number of users and therefore have high proxy and cache requirements. However, it's more than adequate for a small number of users.

`mod_rewrite`

`mod_rewrite` is the Swiss army knife of URL manipulation. It enables you to perform any imaginable manipulation of URLs using powerful regular expressions. It provides rewrites, redirection, proxying, and so on. There is very little that you can't accomplish using this module.

TIP

See http://localhost/manual/misc/rewriteguide.html for a cookbook that gives you an in-depth explanation of what the `mod_rewrite` module is capable of.

mod_setenvif

`mod_setenvif` allows manipulation of environment variables. Using small snippets of text-matching code known as *regular expressions*, you can conditionally change the content of environment variables. The order in which `SetEnvIf` directives appear in the configuration files is important. Each `SetEnvIf` directive can reset an earlier `SetEnvIf` directive when used on the same environment variable. Be sure to keep that in mind when using the directives from this module.

mod_speling

`mod_speling` is used to enable correction of minor typos in URLs. If no file matches the requested URL, this module builds a list of the files in the requested directory and extracts those files that are the closest matches. It tries to correct only one spelling mistake.

mod_status

You can use `mod_status` to create a web page containing a plethora of information about a running Apache server. The page contains information about the internal status as well as statistics about the running Apache processes. This can be a great aid when you're trying to configure your server for maximum performance. It's also a good indicator of when something's amiss with your Apache server.

mod_ssl

`mod_ssl` provides Secure Sockets Layer (version 2 and 3) and transport layer security (version 1) support for Apache. At least 30 directives exist that deal with options for encryption and client authorization and that can be used with this module.

mod_unique_id

`mod_unique_id` generates a unique request identifier for every incoming request. This ID is put into the `UNIQUE_ID` environment variable.

mod_userdir

The `mod_userdir` module enables mapping of a subdirectory in each user's home directory into your web tree. The module provides several ways to accomplish this.

mod_usertrack

mod_usertrack is used to generate a cookie for each user session. This can be used to track the user's click stream within your web tree. You must enable a custom log that logs this cookie into a log file.

mod_vhost_alias

mod_vhost_alias supports dynamically configured mass virtual hosting, which is useful for Internet service providers (ISPs) with many virtual hosts. However, for the average user, Apache's ordinary virtual hosting support should be more than sufficient.

There are two ways to host virtual hosts on an Apache server. You can have one IP address with multiple CNAMEs, or you can have multiple IP addresses with one name per address. Apache has different sets of directives to handle each of these options. (You learn more about virtual hosting in Apache in the next section of this chapter.)

Again, the available options and features for Apache modules are too numerous to describe completely in this chapter. You can find complete information about the Apache modules in the online documentation for the server included with Ubuntu or at the Apache Group's website.

Virtual Hosting

One of the more popular services to provide with a web server is to host a virtual domain. Also known as a *virtual host*, a virtual domain is a complete website with its own domain name, as if it were a standalone machine, but it's hosted on the same machine as other websites. Apache implements this capability in a simple way with directives in the apache2.conf configuration file.

Apache now can dynamically host virtual servers by using the mod_vhost_alias module you read about in the preceding section of the chapter. The module is primarily intended for ISPs and similar large sites that host a large number of virtual sites. This module is for more advanced users and, as such, it is outside the scope of this introductory chapter. Instead, this section concentrates on the traditional ways of hosting virtual servers.

Address-Based Virtual Hosts

After you've configured your Linux machine with multiple IP addresses, setting up Apache to serve them as different websites is simple. You need only put a VirtualHost directive in your apache2.conf file for each of the addresses you want to make an independent website:

```
<VirtualHost 212.85.67.67>
ServerName gnulix.org
DocumentRoot /home/virtual/gnulix/public_html
TransferLog /home/virtual/gnulix/logs/access_log
ErrorLog /home/virtual/gnulix/logs/error_log
</VirtualHost>
```

Use the IP address, rather than the hostname, in the VirtualHost tag.

You can specify any configuration directives within the <VirtualHost> tags. For example, you might want to set AllowOverrides directives differently for virtual hosts than you do for your main server. Any directives that aren't specified default to the settings for the main server.

Name-Based Virtual Hosts

Name-based virtual hosts enable you to run more than one host on the same IP address. You must add the names to your DNS as CNAMEs of the machine in question. When an HTTP client (web browser) requests a document from your server, it sends with the request a variable indicating the server name from which it's requesting the document. Based on this variable, the server determines from which of the virtual hosts it should serve content.

NOTE

Some older browsers are unable to see name-based virtual hosts because this is a feature of HTTP 1.1 and the older browsers are strictly HTTP 1.0–compliant. However, many other older browsers are partially HTTP 1.1–compliant, and this is one of the parts of HTTP 1.1 that most browsers have supported for a while.

Name-based virtual hosts require just one step more than IP address-based virtual hosts. You must first indicate which IP address has the multiple DNS names on it. This is done with the NameVirtualHost directive:

```
NameVirtualHost 212.85.67.67
```

You must then have a section for each name on that address, setting the configuration for that name. As with IP-based virtual hosts, you need to set only those configurations that must be different for the host. You must set the ServerName directive because it's the only thing that distinguishes one host from another:

```
<VirtualHost 212.85.67.67>
ServerName bugserver.gnulix.org
ServerAlias bugserver
DocumentRoot /home/bugserver/htdocs
ScriptAlias /home/bugserver/cgi-bin
TransferLog /home/bugserver/logs/access_log
</VirtualHost>

<VirtualHost 212.85.67.67>
ServerName pts.gnulix.org
ServerAlias pts
DocumentRoot /home/pts/htdocs
ScriptAlias /home/pts/cgi-bin
```

```
TransferLog /home/pts/logs/access_log
ErrorLog /home/pts/logs/error_log
</VirtualHost>
```

TIP

If you are hosting websites on an intranet or internal network, users will likely use the shortened name of the machine rather than the FQDN. For example, users might type http://bugserver/index.html in their browser location field rather than http://bugserver.gnulix.org/index.html. In that case, Apache would not recognize that those two addresses should go to the same virtual host. You could get around this by setting up `VirtualHost` directives for both `bugserver` and `bugserver.gnulix.org`, but the easy way around it is to use the `ServerAlias` directive, which lists all valid aliases for the machine:

```
ServerAlias bugserver
```

For more information about `VirtualHost`, refer to the help system on http://localhost/_manual.

Logging

Apache provides logging for just about any web access information you might be interested in. Logging can help with

- System resource management, by tracking usage
- Intrusion detection, by documenting bad HTTP requests
- Diagnostics, by recording errors in processing requests

Two standard log files are generated when you run your Apache server: `access_log` and `error_log`. They are found under the `/var/log/apache2` directory. (Others include the SSL logs `ssl_access_log`, `ssl_error_log` and `ssl_request_log`.) All logs except for the `error_log` (by default, this is just the `access_log`) are generated in a format specified by the `CustomLog` and `LogFormat` directives. These directives appear in your `apache2.conf` file.

A new log format can be defined with the `LogFormat` directive:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

The `common` log format is a good starting place for creating your own custom log formats. Note that most of the available log analysis tools assume you're using the `common` log format or the `combined` log format—both of which are defined in the default configuration files.

The following variables are available for `LogFormat` statements:

`%a`	Remote IP address.
`%A`	Local IP address.
`%b`	Bytes sent, excluding HTTP headers. This is shown in Apache's Combined Log Format (CLF). For a request without any data content, a `-` is shown instead of `0`.
`%B`	Bytes sent, excluding HTTP headers.
`%{VARIABLE}e`	The contents of the environment variable `VARIABLE`.
`%f`	The filename of the output log.
`%h`	Remote host.
`%H`	Request protocol.
`%{HEADER}i`	The contents of `HEADER`; header line(s) in the request sent to the server.
`%l`	Remote log name (from `identd`, if supplied).
`%m`	Request method.
`%{NOTE}n`	The contents of note `NOTE` from another module.
`%{HEADER}o`	The contents of `HEADER`; header line(s) in the reply.
`%p`	The canonical port of the server serving the request.
`%P`	The process ID of the child that serviced the request.
`%q`	The contents of the query string, prepended with a `?` character. If there's no query string, this evaluates to an empty string.
`%r`	The first line of request.
`%s`	Status. For requests that were internally redirected, this is the status of the original request—`%>s` for the last.
`%t`	The time, in common log time format.
`%{format}t`	The time, in the form given by `format`, which should be in `strftime(3)` format. See the section "Basic SSI Directives" later in this chapter for a complete list of available formatting options.
`%T`	The seconds taken to serve the request.
`%u`	Remote user from `auth`; this might be bogus if the return status (`%s`) is `401`.
`%U`	The URL path requested.
`%V`	The server name according to the `UseCanonicalName` directive.
`%v`	The canonical `ServerName` of the server serving the request.

You can put a conditional in front of each variable to determine whether the variable is displayed. If the variable isn't displayed, `-` is displayed instead. These conditionals are in the form of a list of numerical return values. For example, `%!401u` displays the value of `REMOTE_USER` unless the return code is `401`.

You can then specify the location and format of a log file using the `CustomLog` directive:

```
CustomLog logs/access_log common
```

If it is not specified as an absolute path, the location of the log file is assumed to be relative to the `ServerRoot`.

Other Web Servers for Use with Ubuntu

To determine the best web server for your use, consider the needs of the website you manage. Does it need heavy security (for e-commerce), multimedia (music, video, and pictures), or the capability to download files easily? How much are you willing to spend for the software? Do you need software that is easy to maintain and troubleshoot or that includes tech support? The answers to these questions might steer you to something other than Apache.

The following sections list some of the more popular alternatives to using Apache as your web server.

Sun ONE Web Server

Despite the Netcraft numbers shown previously in Table 17.1, there is evidence that the Sun Java System Web Server (formerly known as the iPlanet Web Server, and subsequently Sun ONE Web Server) might be even more popular than Apache in strictly corporate arenas. The server got its start as the Netscape Enterprise Server—one of the first powerful web servers ever to hit the market. Sun ONE Web Server comes in many flavors, and all of them are big. In addition to the enterprise-level web server that can be run on Ubuntu, the software features application, messaging, calendar, and directory servers—just to name a few.

Sun ONE Web Server is great for handling big web needs, and it comes with an appropriately big price tag. It's definitely not something to run the school website—unless your school happens to be a major state university with several regional campuses. For more information on Sun Java System Web Server, you can visit its website (http://wwws.sun.com/software/products/web_srvr/home_web_srvr.html).

Zope

Zope is another open-source web server. Although it is still relatively young and might not have as much flexibility as Apache, it is making strong inroads in the web server market.

What makes Zope different from Apache is the fact that it is managed through a completely web-based graphic interface. This has broad appeal for those who are not enthused about a command-line–only interface.

Zope is a product of the Zope Corporation (formerly Digital Creations), the same firm that made the Python programming language. And, like all things open source, it is free. Information on Zope can be found at both http://www.zope.com (for the commercial version) and http://www.zope.org (for the open-source version).

Zeus Web Server

Ubuntu sites can also use the Zeus Web Server from Zeus Technology. This server offers a scalable SSL implementation, security settings across multiple websites, and an online administration server. The current price is $1,700 for a host platform with up to two CPUs, but load balancing via the Zeus Load Balancer costs $12,000 (at the time of writing) for each pair of load-balancing computers.

You can get more information about the Zeus Web Server at http://www.zeus.com/products/zws/.

Related Ubuntu and Linux Commands

You will use these commands when managing your Apache web Server in Ubuntu:

- `apache2ctl`—Server control shell script included with Apache
- `apache2`—The Apache web server
- `konqueror`—KDE's graphical web browser
- `elinks`—A text-based, graphical menu web browser
- `firefox`—The premier open-source web browser

Reference

There is a plethora of Apache documentation online. For more information about Apache and the subjects discussed in this chapter, look at some of the following resources:

- http://news.netcraft.com/archives/web_server_survey.html—A statistical graph of web server usage by millions of servers. The research points out that Apache is by far the most widely used server for Internet sites.
- http://www.apache.org/—Extensive documentation and information about Apache are available at The Apache Project website.
- http://www.apacheweek.com/—You can obtain breaking news about Apache and great technical articles at the Apache Week site.
- http://apachetoday.com/—Another good Apache site. Original content as well as links to Apache-related stories on other sites can be found at Apache Today's site.

- http://www.hwg.org/—HTML, CGI, and related subjects are available at The HTML Writers Guild site.
- http://modules.apache.org/—Available add-on modules for Apache can be found at The Apache Module Registry website.

There are several good books about Apache. For example, *Apache Server Unleashed* (Sams Publishing), ISBN 0-672-31808-3.

For more information on Zope, see *The Zope Book* (New Riders Publishing), ISBN 0-7357-11372.

CHAPTER 18
Remote File Serving with FTP

IN THIS CHAPTER

File Transfer Protocol (FTP) was once considered the primary method used to transfer files over a network from computer to computer. FTP is still heavily used today, although many graphical FTP clients now supplement the original text-based interface command. As computers have evolved, so has FTP, and Ubuntu includes many ways with which to use a graphical interface to transfer files over FTP.

This chapter contains an overview of the available FTP software included with Ubuntu, along with some details concerning initial setup, configuration, and use of FTP-specific clients. Ubuntu also includes an FTP server software package named `vsftpd`, the Very Secure FTP Daemon, and a number of associated programs you can use to serve and transfer files with the FTP protocol.

Choosing an FTP Server

FTP uses a client/server model. As a client, FTP accesses a server, and as a server, FTP provides access to files or storage. Just about every computer platform available has software written to enable a computer to act as an FTP server, but Ubuntu provides the average user with the capability do this without paying hefty licensing fees and without regard for client usage limitations.

There are two types of FTP servers and access: anonymous and standard. A *standard* FTP server requires an account name and password from anyone trying to access the server. *Anonymous* servers allow anyone to connect to the server to retrieve files. Anonymous servers provide the most flexibility, but they can also present a security risk. Fortunately, as you will read in this chapter, Ubuntu is set up to use proper file and directory permissions and common-sense default configuration, such as disallowing `root` to perform an FTP login.

NOTE

Many Linux users now use OpenSSH and its suite of clients, such as the `sftp` command, for a more secure solution when transferring files. The OpenSSH suite provides the `sshd` daemon and enables encrypted remote logins (see Chapter 15 for more information).

Choosing an Authenticated or Anonymous Server

When you are preparing to set up your FTP server, you must first make the decision to install either the authenticated or anonymous service. *Authenticated* service requires the entry of a valid username and password for access. As previously mentioned, *anonymous* service allows the use of the username `anonymous` and an email address as a password for access.

Authenticated FTP servers are used to provide some measure of secure data transfer for remote users, but will require maintenance of user accounts as usernames and passwords are used. Anonymous FTP servers are used when user authentication is not needed or necessary, and can be helpful in providing an easily accessible platform for customer support or public distribution of documents, software, or other data.

If you use an anonymous FTP server in your home or business Linux system, it is vital that you properly install and configure it to retain a relatively secure environment. Generally, sites that host anonymous FTP servers place them outside the firewall on a dedicated machine. The dedicated machine contains only the FTP server and should not contain data that cannot be restored quickly. This dedicated-machine setup prevents malicious users who compromise the server from obtaining critical or sensitive data. For an additional, but by no means more secure setup, the FTP portion of the file system can be mounted read-only from a separate hard drive partition or volume, or mounted from read-only media, such as CD-ROM, DVD, or other optical storage.

Ubuntu FTP Server Packages

The Very Secure `vsftpd` server, like `wu-ftpd` (also discussed in this chapter), is licensed under the GNU GPL. The server can be used for personal or business purposes. The `wu-ftpd` and `vsftpd` servers are covered in the remainder of this chapter.

Other FTP Servers

One alternative server is NcFTPd, available from http://www.ncftp.com. This server operates independently of `inetd` (typically used to enable and start the `wu-ftp` server) and provides its own optimized daemon. Additionally, NcFTPd has the capability to cache directory listings of the FTP server in memory, thereby increasing the speed at which users can obtain a list of available files and directories. Although NcFTPd has many advantages over `wu-ftpd`, NcFTPd is not GPL-licensed software, and its licensing fees vary according to the maximum number of simultaneous server connections ($199 for 51 or more concurrent users and $129 for up to 50 concurrent users, but free to education institutions with a compliant domain name).

NOTE

Do not confuse the `ncftp` client with `ncftpd`. The `ncftp-3.1` package included with Ubuntu is the client software, a replacement for `ftp-0.17`, and includes the `ncftpget` and `ncftpput` commands for transferring files via the command line or by using a remote file uniform resource locator (URL) address. `ncftpd` is the FTP server, which can be downloaded from www.ncftpd.com.

Another FTP server package for Linux is ProFTPD, licensed under the GNU GPL. This server works well with most Linux distributions and has been used by a number of Linux sites, including ftp.kernel.org and ftp.sourceforge.net. ProFTPD is actively maintained and updated for bug fixes and security enhancements. Its developers recommend that you use the latest release (1.2.10 at the time of this writing) to avoid exposure to exploits and vulnerabilities. Browse to http://www.proftpd.org to download a copy.

Yet another FTP server package is `Bsdftpd-ssl`, which is based on the BSD `ftpd` (and distributed under the BSD license). `Bsdftpd-ssl` offers simultaneous standard and secure access using security extensions; secure access requires a special client. For more details, browse to http://bsdftpd-ssl.sc.ru.

Finally, another alternative is to use Apache (and the HTTP protocol) for serving files. Using a web server to provide data downloads can reduce the need to monitor and maintain a separate software service (or directories) on your server. This approach to serving files also reduces system resource requirements and gives remote users a bit more flexibility when downloading (such as enabling them to download multiple files at once). See Chapter 17, “Apache Web Server Management,” for more information about using Apache.

Installing FTP Software

As part of the standard installation, the client software for FTP is already installed. You can verify that FTP-related software is installed on your system by using `dpkg`, `grep`, and `sort` commands in this query:

```
$ dpkg --get-selections ¦ grep ftp ¦ sort
```

The example results might differ, depending on what software packages are installed. In your Ubuntu file system, you will find the `/usr/bin/pftp` file symbolically linked to

`/usr/bin/netkit-ftp` as well as the `vsftpd` server under the `/usr/sbin` directory. Other installed packages include additional text-based and graphical FTP clients.

If `vsftpd` is not installed, you can install it through Synaptic.

NOTE

If you host an FTP server connected to the Internet, make it a habit to always check the Ubuntu site for up-to-date system errata and security and bug fixes for your server software.

You can find packages for a wide variety of FTP tools, including `wu-ftpd`, in Synaptic. Again, just make sure you have the Universe repository enabled.

The FTP User

After installing Ubuntu, an FTP user is created. This user is not a normal user per se, but a name for anonymous FTP users. The FTP user entry in `/etc/passwd` looks like this:

```
ftp:x:14:50:FTP User:/home/ftp:/bin/false
```

NOTE

The FTP user, as discussed here, applies to anonymous FTP configurations and server setup.

Also, note that other Linux distributions might use a different default directory, such as `/usr/local/ftp`, for FTP files and anonymous users.

This entry follows the standard `/etc/passwd` entry: username, password, User ID, Group ID, comment field, home directory, and shell. To learn more about `/etc/password`, see the section "The Password File" in Chapter 10, "Managing Users."

Each of the items in this entry is separated by colons. In the preceding example, you can see that the Ubuntu system hosting the server uses `shadowed password` because an `x` is present in the traditional password field. The shadow password system is important because it adds an additional level of security to Ubuntu; the shadow password system is normally installed during the Ubuntu installation.

The FTP server software uses this user account to assign permissions to users connecting to the server. By using a default shell of `/bin/false` for anonymous FTP users versus `/bin/bash` or some other standard, interactive shell, an anonymous FTP user, will be unable to log in as a regular user. `/bin/false` is not a shell, but a program usually assigned to an account that has been locked. As root inspection of the `/etc/shadow` file shows (see Listing 18.2), it is not possible to log in to this account, denoted by the `*` as the password.

LISTING 18.1 Shadow Password File `ftp` User Entry

```
# cat /etc/shadow
bin:*:11899:0:99999:7:::
daemon:*:11899:0:99999:7:::
adm:*:11899:0:99999:7:::
lp:*:11899:0:99999:7:::
...
ftp:*:12276:0:99999:7:::
...
```

The shadow file (only a portion of which is shown in Listing 18.1) contains additional information not found in the standard `/etc/passwd` file, such as account expiration, password expiration, whether the account is locked, and the encrypted password. The `*` in the password field indicates that the account is not a standard login account; thus, it does not have a password.

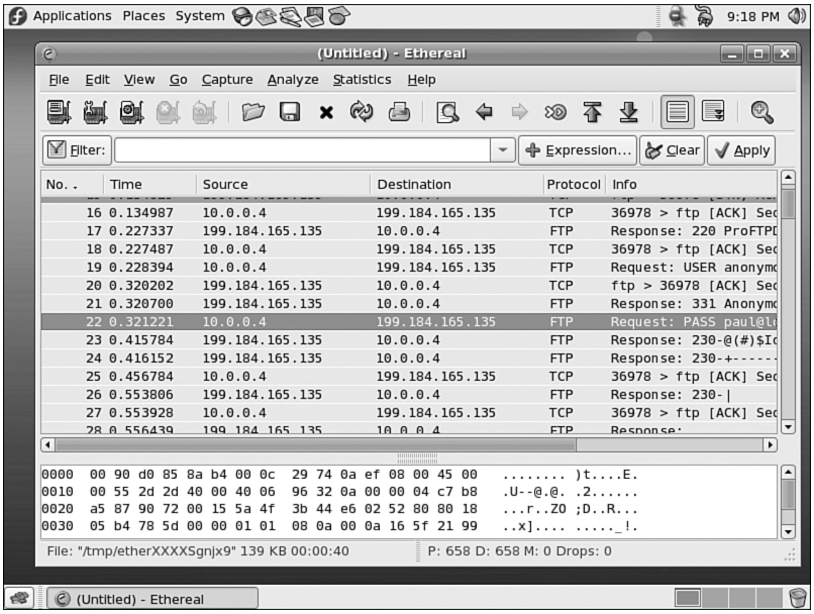
Although shadow passwords are in use on the system, passwords are not transmitted in a secure manner when using FTP. Because FTP was written before the necessity of encryption and security, it does not provide the mechanics necessary to send encrypted passwords. Account information is sent in plain text on FTP servers; anyone with enough technical knowledge and a network sniffer can find the password for the account you connect to on the server. Many sites use an anonymous-only FTP server specifically to prevent normal account passwords from being transmitted over the Internet.

Figure 18.1 shows a portion of an `ethereal` capture of an FTP session for an anonymous user and his password ("`foobarbaz`"). The `ethereal` client is a graphical browser used to display network traffic in real time, and it can be used to watch packet data, such as an FTP login on a LAN.

Quick and Dirty FTP Service

Conscientious Linux administrators will take the time to carefully install, set up, and configure a production FTP server before offering public service or opening up for business on the Internet. However, you can set up a server very quickly on a secure LAN by following a few simple steps:

1. Ensure that the FTP server package is installed, networking is enabled, and firewall rules on the server allow FTP access. See Chapter 14, "Networking," to learn about firewalling.
2. If anonymous access to server files is desired, populate the `/home/ftp/pub` directory. Do this by mounting or copying your content, such as directories and files, under this directory.
3. Edit and then save the appropriate configuration file (such as `vsftpd.conf` for `vsftpd`) to enable access.
4. If you are using `wu-ftpd`, you must then start or restart `inetd` like so: `/etc/init.d/inetutils-inetd restart`. Make sure you have the `inetutils-inetd` package installed through Synaptic/APT. If you are using `vsftpd`, you must then start or restart the server like so: `service vsftpd start`.



No.	Time	Source	Destination	Protocol	Info
16	0.134987	10.0.0.4	199.184.165.135	TCP	36978 > ftp [ACK] Sec
17	0.227337	199.184.165.135	10.0.0.4	FTP	Response: 220 ProFTPD
18	0.227487	10.0.0.4	199.184.165.135	TCP	36978 > ftp [ACK] Sec
19	0.228394	10.0.0.4	199.184.165.135	FTP	Request: USER anonymo
20	0.320202	199.184.165.135	10.0.0.4	TCP	ftp > 36978 [ACK] Sec
21	0.320700	199.184.165.135	10.0.0.4	FTP	Response: 331 Anonymo
22	0.321221	10.0.0.4	199.184.165.135	FTP	Request: PASS paul@lo
23	0.415784	199.184.165.135	10.0.0.4	FTP	Response: 230-@(#)$Id
24	0.416152	199.184.165.135	10.0.0.4	FTP	Response: 230-+------
25	0.456784	10.0.0.4	199.184.165.135	TCP	36978 > ftp [ACK] Sec
26	0.553806	199.184.165.135	10.0.0.4	FTP	Response: 230-|
27	0.553928	10.0.0.4	199.184.165.135	TCP	36978 > ftp [ACK] Sec
28	0.556439	199.184.165.135	10.0.0.4	FTP	Response:

```
0000  00 90 d0 85 8a b4 00 0c  29 74 0a ef 08 00 45 00   ........ )t....E.
0010  00 55 2d 2d 40 00 40 06  96 32 0a 00 00 04 c7 b8   .U--@.@. .2......
0020  a5 87 90 72 00 15 5a 4f  3b 44 e6 02 52 80 80 18   ...r..ZO ;D..R...
0030  05 b4 78 5d 00 00 01 01  08 0a 00 0a 16 5f 21 99   ..x].... ....._!.
```

File: "/tmp/etherXXXXSgnjx9" 139 KB 00:00:40 P: 658 D: 658 M: 0 Drops: 0

FIGURE 18.1 The `ethereal` client can filter and sniff FTP sessions to capture usernames and passwords.

`inetd` Configuration for `wu-ftpd`

`inetd` (pronounced "eye-net-d") is the extended Internet services daemon, and handles incoming connections for network services.

This daemon controls a number of services on your system, according to settings in `/etc/inetd.conf`.

Using an editor, change the `disable = yes` line to `disable = no`. Save the file and exit the editor. You then must restart `inetd` because configuration files are parsed only at startup. To start `inetd` as root, issue the command **`/etc/init.d/inetutils-inetd start`**. This makes a call to the same shell script that is called at any runlevel to start or stop the `inetd` daemon (and thus start up or shut down the system). `inetd` should report its status as

```
# /etc/init.d/inetutils-inetd restart
Stopping internet superserver inetd:                                   [  OK  ]
Starting internet superserver inetd:                                   [  OK  ]
```

After it is restarted, the FTP server is accessible to all incoming requests.

Starting the Very Secure FTP Server (vsftpd) Package

You can use the shell script named vsftp under the /etc/init.d directory to start, stop, restart, and query the vsftpd server. You must have root permission to use the vsftpd script to control the server, but any user can query the server (to see if it is running and to see its process ID number) using the status keyword, like this:

```
$ /etc/init.d/vsftpd status
```

Be sure not to run two FTP servers on your system at the same time!

Configuring the Very Secure FTP Server

The server offers simplicity, security, and speed. It has been used by a number of sites, such as ftp.debian.org, ftp.gnu.org, rpmfind.net, and ftp.gimp.org. Note that despite its name, the Very Secure FTP server does not enable use of encrypted usernames or passwords.

Its main configuration file is vsftpd.conf, which resides under the /etc directory. The server has a number of features and default policies, but these can be overridden by changing the installed configuration file.

By default, anonymous logins are enabled, but users are not allowed to upload files, create new directories, or delete or rename files. The configuration file installed by Ubuntu allows local users (that is, users with a login and shell account) to log in and then access their home directory. This configuration presents potential security risks because usernames and passwords are passed without encryption over a network. The best policy is to deny your users access to the server from their user accounts. The standard vsftpd configuration disables this feature.

Controlling Anonymous Access

Toggling anonymous access features for your FTP server is done by editing the vsftpd.conf file and changing related entries to YES or NO in the file. Settings to control how the server works for anonymous logins include:

- anonymous_enable—Enabled by default. Use a setting of NO, and then restart the server to turn off anonymous access.
- anon_mkdir_write_enable—Allows or disallows creating of new directories.
- anon_other_write_enable—Allows or disallows deleting or renaming of files and directories.

- `anon_upload_enable`—Controls whether anonymous users can upload files (also depends on the global `write_enable` setting). This is a potential security and liability hazard and should rarely be used; if enabled, consistently monitor any designated upload directory.
- `anon_world_readable_only`—Allows only anonymous users to download files with world-readable (444) permission.

After making any changes to your server configuration file, make sure to restart the server; this forces `vsftpd` to reread its settings.

Other `vsftpd` Server Configuration Files

You can edit `vsftpd.conf` to enable, disable, and configure many features and settings of the `vsftpd` server, such as user access, filtering of bogus passwords, and access logging. Some features might require the creation and configuration of other files, such as:

- `/etc/vsftpd.user_list`—Used by the `userlist_enable` and/or the `userlist_deny` options; the file contains a list of usernames to be denied access to the server.
- `/etc/vsftpd.chroot_list`—Used by the `chroot_list_enable` and/or `chroot_local_user` options, this file contains a list of users who are either allowed or denied access to a home directory. An alternate file can be specified by using the `chroot_list_file` option.
- `/etc/vsftpd.banned_emails`—A list of anonymous password entries used to deny access if the `deny_email_enable` setting is enabled. An alternative file can be specified by using the `banned_email` option.
- `/var/log/vsftpd.log`—Data transfer information is captured to this file if logging is enabled using the `xferlog_enable` setting.

TIP

Whenever editing the FTP server files, make a backup file first. Also, it is always a good idea to comment out (using a pound sign at the beginning of a line) what is changed instead of deleting or overwriting entries. Follow these comments with a brief description explaining why the change was made. This leaves a nice audit trail of what was done, by whom, when, and why. If you have any problems with the configuration, these comments and details can help you troubleshoot and return to valid entries if necessary. You can use the `dpkg` command or other Linux tools (such as `mc`) to extract a fresh copy of a configuration file from the software's package archive. Be aware, however, that the extracted version will replace the current version and overwrite your configuration changes.

Default `vsftpd` Behaviors

The contents of a file named `.message` (if it exists in the current directory) are displayed when a user enters the directory. This feature is enabled in the installed configuration file, but disabled by the daemon. FTP users are also not allowed to perform recursive directory listings, which can help reduce bandwidth use.

The PASV data connection method is enabled to let external users know the IP address of the FTP server. This is a common problem when using FTP from behind a firewall/gateway using IP masquerading or when incoming data connections are disabled because without passive mode, the remote server tries to form a connection to your local host and gets blocked. For example, here is a connection to an FTP server (running ProFTPD), an attempt to view a directory listing, and the resulting need to use `ftp`'s internal `passive` command:

```
$ ftp ftp.tux.org
Connected to gwyn.tux.org.
220 ProFTPD 1.2.5rc1 Server (ProFTPD on ftp.tux.org) [gwyn.tux.org]
500 AUTH not understood.
KERBEROS_V4 rejected as an authentication type
Name (ftp.tux.org:gbush): gbush
331 Password required for gbush.
Password:
230 User gbush logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd public_html
250 CWD command successful.
ftp> ls
500 Illegal PORT command.
ftp: bind: Address already in use
ftp>
ftp> pass
Passive mode on.
ftp> ls
227 Entering Passive Mode (204,86,112,12,187,89).
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 gbush    gbush        8470 Jan 10  2000 LinuxUnleashed.gif
-rw-r--r--   1 gbush    gbush        4407 Oct  4  2001 RHU72ed.gif
-rw-r--r--   1 gbush    gbush        6732 May 18  2000 SuSEUnleashed.jpg
-rw-r--r--   1 gbush    gbush        6175 Jan 10  2000 TYSUSE.gif
-rw-r--r--   1 gbush    gbush        3135 Jan 10  2000 TZones.gif
...
```

NOTE

Browse to http://slacksite.com/other/ftp.html for a detailed discussion regarding active and passive FTP modes and the effect of firewall blocking of service ports on FTP server and client connections.

Other default settings are that specific user login controls are not set, but you can configure the controls to deny access to one or more users.

The data transfer rate for anonymous client access is unlimited, but you can set a maximum rate (in bytes per second) by using the `anon_max_rate` setting in `vsftpd.conf`. This can be useful for throttling bandwidth use during periods of heavy access. Another default is that remote clients will be logged out after five minutes of idle activity or a stalled data transfer. You can set idle and transfer timeouts (stalled connections) separately.

Other settings that might be important for managing your system's resources (networking bandwidth or memory) when offering FTP access include the following:

- `dirlist_enable`—Toggles directory listings on or off.
- `dirmessage_enable`—Toggles display of a message when the user enters a directory. A related setting is `ls_recurse_enable`, which can be used to disallow recursive directory listings.
- `download_enable`—Toggles downloading on or off.
- `max_clients`—Sets a limit on the maximum number of connections.
- `max_per_ip`—Sets a limit on the number of connections from the same IP address.

Configuring the Server

`Wu-FTP` uses a number of configuration files to control how it operates, including the following:

- `ftpaccess`—Contains the majority of server configuration settings
- `ftpconversions`—Contains definitions of file conversions during transfers
- `ftphosts`—Settings to control user access from specific hosts

These files may be created in the `/etc` directory during installation, or may be created by a system administrator. The following sections describe each of these files and how to use the commands they contain to configure the `Wu-FTP` server so that it is accessible to all incoming requests.

CAUTION

When configuring an anonymous FTP server, it is extremely important to ensure that all security precautions are taken to prevent malicious users from gaining privileged-level access to the server. Although this chapter shows you how to configure your FTP server for secure use, all machines connected to the Internet are potential targets for malicious attacks. Vulnerable systems can be a source of potential liability, especially if anyone accesses and uses them to store illegal copies of proprietary software—even temporarily.

Using Commands in the `ftpaccess` File to Configure `wu-ftpd`

The `ftpaccess` file contains most of the server configuration details. Each line contains a definition or parameter that is passed to the server to specify how the server is to operate. The directives can be broken down into the following categories, including:

- **Access Control**—Settings that determine who can access the FTP server and how it is accessed
- **Information**—Settings that determine what information is provided by the server or displayed to a user
- **Logging**—Settings that determine whether logging is enabled and what information is logged
- **Permission Control**—Settings that control the behavior of users when accessing the server; in other words, what actions users are allowed to perform, such as create a directory, upload a file, delete a file or directory, and so on

TIP

Many more options can be specified for the `wu-ftpd` FTP server in its `ftpaccess` file. The most common commands have been covered here. A full list of configuration options can be found in the `ftpaccess` man page after you install the server.

You can edit the `ftpaccess` file at the command line to make configuration changes in any of these categories. The following sections describe some configuration changes and how to edit these files to accomplish them.

Configure Access Control

Controlling which users can access the FTP server and how they can access it is a critical part of system security. Use the following entries in the `ftpaccess` file to specify to which group the user accessing the server is assigned.

Limit Access for Anonymous Users

This command imposes increased security on the anonymous user:

```
autogroup <groupname> <class> [<class>]
```

If the anonymous user is a member of a group, he will only be allowed access to files and directories owned by him or his group. The group must be a valid group from `/etc/groups`.

Define User Classes

This command defines a class of users by the address to which the user is connected:

```
class <class> <typelist> <addrglob> [<addrglob>]
```

There might be multiple members for a class of users, and multiple classes might apply to individual members. When multiple classes apply to one user, the first class that applies will be used.

The `typelist` field is a comma-separated list of the keywords `anonymous`, `guest`, and `real`. `anonymous` applies to the anonymous user, and `guest` applies to the guest access account, as specified in the `guestgroup` directive. `real` defines those users who have a valid entry in the `/etc/passwd` file.

The `addrglob` field is a regular expression that specifies addresses to which the class is to be applied. The (`*`) entry specifies all hosts.

Block a Host's Access to the Server

Sometimes it is necessary to block access to the server to entire hosts. This can be useful to protect the system from individual hosts or entire blocks of IP addresses, or to force the use of other servers. Use this command to do so:

```
deny <addrglob> <message_file>
```

`deny` will always deny access to hosts that match a given address.

`addrglob` is a regular expression field that contains a list of addresses, either numeric or a DNS name. This field can also be a file reference that contains a listing of addresses. If an address is a file reference, it must be an absolute file reference; that is, starting with a `/`. To ensure that IP addresses can be mapped to a valid domain name, use the `!nameserver` parameter.

A sample `deny` line resembles the following:

```
deny *.exodous.net /home/ftp/.message_exodous_deny
```

This entry denies access to the FTP server from all users who are coming from the exodous.net domain, and displays the message contained in the `.message_exoduous_deny` file in the `/home/ftp` directory.

ftpusers File Purpose Now Implemented in ftpaccess

Certain accounts for the system to segment and separate tasks with specific permissions are created during Linux installation. The `ftpusers` file (located in `/etc/ftpusers`) is where accounts for system purposes are listed. It is possible that the version of `wu-ftp` you use with Ubuntu deprecates the use of this file, and instead implements the specific functionality of this file in the `ftpaccess` file with the commands of `deny-uid/deny-gid`.

Restrict Permissions Based on Group IDs

The `guestgroup` line assigns a given group name or group names to behave exactly like the anonymous user. Here is the command:

```
guestgroup <groupname> [<groupname>]
```

This command confines the users to a specific directory structure in the same way anonymous users are confined to `/var/ftp`. This command also limits these users to access files for which their assigned group has permissions.

The `groupname` parameter can be the name of a group or that group's corresponding Group ID (*GID*). If you use a GID as the `groupname` parameter, put a percentage symbol (`%`) in front of it. You can use this command to assign permissions to a range of group IDs, as in this example:

```
guestgroup %500-550
```

This entry restricts all users with the group IDs 500–550 to being treated as a guest group, rather than individual users. For `guestgroup` to work, you must set up the user's home directories with the correct permissions, exactly like the anonymous FTP user.

Limit Permissions Based on Individual ID

The `guestuser` line works exactly like the `guestgroup` command you just read about, except it specifies a User ID (*UID*) instead of a group ID. Here's the command:

```
guestuser <username> [<username>]
```

This command limits the guest user to files for which the user has privileges. Generally, a user has more privileges than a group, so this type of assignment can be less restrictive than the `guestgroup` line.

Restrict the Number of Users in a Class

The `limit` command restricts the number of users in a class during given times. Here is the command, which contains fields for specifying a class, a number of users, a time range, and the name of a text file that contains an appropriate message:

```
limit <class> <n> <times> <message_file>
```

If the specified number of users from the listed class is exceeded during the given time period, the user sees the contents of the file given in the `message_file` parameter.

The `times` parameter is somewhat terse. Its format is a comma-delimited string in the form of days, hours. Valid day strings are `Su`, `Mo`, `Tu`, `We`, `Th`, `Fr`, `Sa`, and `Any`. The hours are formatted in a 24-hour format. An example is as follows:

```
limit anonymous 10 MoTuWeThFr,Sa0000-2300 /home/ftp/.message_limit_anon_class
```

This line limits the anonymous class to 10 concurrent connections on Monday through Friday, and on Saturday from midnight to 11:00 p.m. If the number of concurrent connections is exceeded or at 11:00 p.m. on Saturday, the users will see the contents of the file `/home/ftp/.message_limit_anon_class`.

Syntax for finer control over limiting user connections can be found in the `ftpaccess` man page.

Limit the Number of Invalid Password Entries

This line allows control over how many times a user can enter an invalid password before the FTP server terminates the session:

```
loginfails <number>
```

The default for `loginfails` is set to `5`. This command prevents users without valid passwords from experimenting until they "get it right."

Configure User Information

Providing users with information about the server and its use is a good practice for any administrator of a public FTP server. Adequate user information can help prevent user problems and eliminate tech support calls. You also can use this information to inform users of restrictions governing the use of your FTP server. User information gives you an excellent way to document how your FTP server should be used.

You can use the commands detailed in the following sections to display messages to users as they log in to the server or as they perform specific actions. The following commands enable messages to be displayed to users when logging in to the server or when an action is performed.

Display a Prelogin Banner

This command is a reference to a file that is displayed before the user receives a login prompt from the FTP server:

```
banner <path>
```

This file generally contains information to identify the server. The path is an absolute pathname relative to the system root (`/`), not the base of the anonymous FTP user's home. The entry might look like this:

```
banner /etc/uftp.banner
```

This example uses the file named `uftp.banner` under the `/etc` directory. The file can contain one or more lines of text, such as:

```
Welcome to Widget, Inc.'s Red Hat Linux FTP server.
This server is only for use of authorized users.
Third-party developers should use a mirror site.
```

When an FTP user attempts to log in, the banner is displayed like so:

```
$ ftp shuttle2
Connected to shuttle2.home.org.
220-Welcome to Widget, Inc.'s FTP server.
220-This server is only for use of authorized users.
220-Third-party developers should use a mirror site.
220-
220-
220 shuttle2 FTP server (Version wu-2.6.2-8) ready.
504 AUTH GSSAPI not supported.
504 AUTH KERBEROS_V4 not supported.
KERBEROS_V4 rejected as an authentication type
Name (shuttle2:phudson):
```

NOTE

Note that the banner does not replace the greeting text that, by default, displays the hostname and server information, such as:

```
220 shuttle2 FTP server (Version wu-2.6.2-8) ready.
```

To hide version information, use the `greeting` command in `ftpaccess` with a keyword, such as `terse`, like so:

```
greeting terse
```

FTP users then see a short message like this as part of the login text:

```
220 FTP server ready.
```

Also, not all FTP clients can handle multiline responses from the FTP server. The `banner <path>` command is how the banner line passes the file contents to the client. If clients cannot interrupt multiline responses, the FTP server is useless to them. You should also edit the default banner to remove identity and version information.

Display a File

This line specifies a text file to be displayed to the user during login and when the user issues the `cd` command:

```
message <path> {<when> {<class> ...}}
```

The optional `when` clause can be `LOGIN` or `CWD=(dir)`, where `dir` is the name of a directory that is current. The optional `class` parameter enables messages to be shown to only a given class or classes of users.

Using messages is a good way to give information about where things are on your site as well as information that is system dependent, such as alternative sites, general policies regarding available data, server availability times, and so on.

You can use magic cookies to breathe life into your displayed messages. *Magic cookies* are symbolic constants that are replaced by system information. Table 18.1 lists the `message` command's valid magic cookies and their representations.

TABLE 18.1 Magic Cookies and Their Descriptions

Cookie	**Description**
`%T`	Local time (form `Thu Nov 15 17:12:42 1990`)
`%F`	Free space in partition of CWD (kilobytes) [Not supported on all systems]
`%C`	Current working directory
`%E`	Maintainer's email address as defined in `ftpaccess`
`%R`	Remote hostname
`%L`	Local hostname
`%u`	Username as determined via RFC931 authentication
`%U`	Username given at login time
`%M`	Maximum allowed number of users in this class
`%N`	Current number of users in this class
`%B`	Absolute limit on disk blocks allocated
`%b`	Preferred limit on disk blocks
`%Q`	Current block count
`%I`	Maximum number of allocated inodes (+1)
`%i`	Preferred inode limit
`%q`	Current number of allocated inodes
`%H`	Time limit for excessive disk use
`%h`	Time limit for excessive files
Ratios	
`%xu`	Uploaded bytes
`%xd`	Downloaded bytes
`%xR`	Upload/download ratio (1:*n*)
`%xc`	Credit bytes
`%xT`	Time limit (minutes)
`%xE`	Elapsed time since login (minutes)
`%xL`	Time left
`%xU`	Upload limit
`%xD`	Download limit

To understand how this command works, imagine that you want to display a welcome message to everyone who logs in to the FTP server. An entry of:

```
message /home/ftp/welcome.msg  login
message /welcome.msg           login
```

shows the contents of the `welcome.msg` file to all real users who log in to the server. The second entry shows the same message to the anonymous user.

The `welcome.msg` file is not created with the installation of the package, but you can create it using a text editor. Type the following:

```
Welcome to the anonymous ftp service on %L!

There are %N out of %M users logged in.

Current system time is %T

Please send email to %E if there are
any problems with this service.

Your current working directory is %C
```

Save this file as `/var/ftp/welcome.msg`. Verify that it works by connecting to the FTP server:

```
220 FTP server ready.
504 AUTH GSSAPI not supported.
504 AUTH KERBEROS_V4 not supported.
KERBEROS_V4 rejected as an authentication type
Name (shuttle:phudson): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230-Welcome to the anonymous ftp service on shuttle.home.org!
230-
230-There are 1 out of unlimited users logged in.
230-
230-Current system time is Mon Nov  3 10:57:06 2003
230-
230-Please send email to root@localhost if there are
230-any problems with this service.
230-Your current working directory is /
```

Display Administrator's Email Address

This line sets the email address for the FTP administrator:

```
email <name>
```

This string is printed whenever the `%E` magic cookie is specified. This magic cookie is used in the messages line or in the shutdown file. You should display this string to users in the login banner message so that they know how to contact you (the administrator) in case of problems with the FTP server.

CAUTION

Do not use your live email address in the display banner; you want others to be able to access user emails as necessary. Instead, use an alias address that routes the messages to the appropriate IT department or other address.

Notify User of Last Modification Date

The `readme` line tells the server whether a notification should be displayed to the user when a specific file was last modified. Here's the command:

```
readme <path> {<when {<class>}}
```

The `path` parameter is any valid path for the user. The optional `when` parameter is exactly as seen in the message line. `class` can be one or more classes as defined in the class file. The `path` is absolute for real users. For the anonymous user, the `path` is relative to the anonymous home directory, which is `/var/ftp` by default.

Configure System Logging

Part of any system administration involves reviewing log files for what the server is doing, who accessed it, what files were transferred, and other pieces of important information. You can use a number of commands within `/etc/ftpacess` to control your FTP server logging actions.

Redirect Logging Records

This line allows the administrator to redirect where logging information from the FTP server is recorded:

```
log <syslog>{+<xferlog>}
```

By default, the information for commands is stored in `/var/log/messages`, although the man pages packaged in some packages state that this information is written to `/var/log/xferlog`. Check your server's settings for information regarding the location of your file transfer logs.

Log All User-Issued Commands

This line enables logging for all commands issued by the user:

```
log commands [<typelist>]
```

`typelist` is a comma-separated list of `anonymous`, `guest`, and `real`. If no `typelist` is given, commands are logged for all users. Some `wu-ftpd` packages set the logging of all file transfers to `/var/log/xferlog` (see the next section). However, you can add the `log` command to `ftpaccess` with the `commands` keyword to capture user actions. Logging is then turned on and user actions are captured in `/var/log/messages`. Here is a sample log file:

```
Oct  6 12:21:42 shuttle2 ftpd[5229]: USER anonymous
Oct  6 12:21:51 shuttle2 ftpd[5229]: PASS phudson@widget.com
Oct  6 12:21:51 shuttle2 ftpd[5229]: ANONYMOUS FTP LOGIN FROM 192.168.2.31
➥[192.168.2.31], phudson@widget.com
Oct  6 12:21:51 shuttle2 ftpd[5229]: SYST
Oct  6 12:21:54 shuttle2 ftpd[5229]: CWD pub
Oct  6 12:21:57 shuttle2 ftpd[5229]: PASV
Oct  6 12:21:57 shuttle2 ftpd[5229]: LIST
Oct  6 12:21:59 shuttle2 ftpd[5229]: QUIT
Oct  6 12:21:59 shuttle2 ftpd[5229]: FTP session closed
```

The sample log shows the username and password entries for an anonymous login. The `CWD` entry shows that a `cd` command is used to navigate to the `pub` directory. Note that the commands shown do not necessarily reflect the syntax the user typed, but instead list corresponding system calls the FTP server received. For example, the `LIST` entry is actually the `ls` command.

Log Security Violations and File Transfers

Two other logging commands are useful in the `/etc/ftpaccess` configuration file. This line enables the logging of security violations:

```
log security [<typelist>]
```

Violations are logged for anonymous, guest, and real users, as specified in the `typelist`—the same as other `log` commands. If you do not specify a `typelist`, security violations for all users are logged.

This line writes a log of all files transferred to and from the server:

```
log transfers [<typelist> [<directions>]]
```

`typelist` is the same as seen in log commands and log security lines. `directions` is a comma-separated list of the keywords `inbound` for uploaded files and `outbound` for downloaded files. If no `directions` are given, both uploaded and downloaded files are logged. Inbound and outbound logging is turned on by default.

Configure Permission Control

Controlling user activity is an important component of securing your system's server. The `ftpaccess` file includes a number of commands that enable you to determine what users can and cannot execute during an FTP session. You can use these permission controls to allow users to change file permissions, delete or overwrite files, rename files, and create new files with default permissions. You learn how to use all these `ftpaccess` file command lines in the following sections.

NOTE

By default, all the `ftpaccess` file command lines prohibit anonymous users from executing actions and enable authorized users to do so.

Allow Users to Change File Permissions

The `chmod` line determines whether a user has the capability to change a file's permissions. Here is the command line:

```
chmod <yes¦no> <typelist>
```

This command acts the same as the standard `chmod` command.

The `yes¦no` parameter designates whether the command can be executed. `typelist` is a comma-delimited string of the keywords `anonymous`, `guest`, and `real`. If you do not specify a `typelist` string, the command is applied to all users. An exhaustive description of its purpose and parameters can be found in the man page.

Assign Users File-Delete Permission

The `delete` line determines whether the user can delete files with the `rm` command. Here's the command line:

```
delete<yes¦no> <typelist>
```

The `yes¦no` parameter is used to turn this permission on or off, and `typelist` is the same as the `chmod` command.

Assign Users File-Overwrite Permission

This command line of the `ftpaccess` file allows or denies users the ability to overwrite an existing file. Here's the command line:

```
overwrite <yes¦no> <typelist>
```

The FTP client determines whether users can overwrite files on their own local machines; this line specifically controls overwrite permissions for uploads to the server. The `yes|no` parameter toggles the permission on or off, and `typelist` is the same as seen in the `chmod` line.

Allow Users to Rename Files

You can enable or prevent a user from renaming files by using this command line:

```
rename <yes¦no> <typelist>
```

The `yes¦no` parameter toggles the permission on or off, and `typelist` is the same comma-delimited string as seen in `chmod`.

Allow Users to Compress Files

This line determines whether the user is able to use the `compress` command on files:

```
compress <yes¦no> [<classglob>]
```

The `yes|no` parameter toggles the permission on or off, and `classglob` is a regular expression string that specifies one or more defined classes of users. The conversions that result from the use of this command are specified in the `ftpconversions` file, which contains directions on what compression or extraction command is to be used on a file with a specific extension, such as `.Z` for the `compress` command, `.gz` for the `gunzip` command, and so on. See the section "Configuring FTP Server File-Conversion Actions" later in this chapter.

Assign or Deny Permission to Use `tar`

This line determines whether the user is able to use the `tar` (tape archive) command on files:

```
tar <yes¦no> [<classglob> ...]
```

The `yes¦no` parameter toggles the permission on or off, and `classglob` is a regular expression string that specifies one or more defined classes of users. Again, the conversions that result from the use of this command are specified in the `ftpconversions` file.

Determine What Permissions Can Apply to User-Created Upload Files

This line is a bit different from the other commands in the permission control section. The `umask` command determines with what permissions a user can create new files; here it is.

```
umask <yes¦no> <typelist>
```

The `yes¦no` parameter toggles based on whether a user is allowed to create a file with his default permissions when uploading a file. Like the `overwrite` command you read about earlier in this section, this command line is specific to uploaded files because the client machine determines how new files are created from a download.

Configure Commands Directed Toward the `cdpath`

This `alias` command allows the administrator to provide another name for a directory other than its standard name:

```
alias <string> <dir>
```

The `alias` line applies to only the `cd` command. This line is particularly useful if a popular directory is buried deep within the anonymous FTP user's directory tree. The following is a sample entry:

```
alias linux-386 /pub/slackware/11/i386/
```

This line would allow the user to type `cd linux-386` and be automatically taken to the `/pub/redhat/7.3/en/i386` directory.

The `cdpath <dir>` line specifies the order in which the `cd` command looks for a given user-entered string. The search is performed in the order in which the `cdpath` lines are entered in the `ftpacess` file.

For example, if the following `cdpath` entries are in the `ftpaccess` file,

```
cdpath /pub/slackware/
cdpath /pub/linux/
```

and the user types `cd i386`, the server searches for an entry in any defined aliases, first in the `/pub/slackware` directory and then in the `/pub/linux` directory. If a large number of aliases are defined, it is recommended that symbolic links to the directories be created instead of aliases. Doing so reduces the amount of work on the FTP server and decreases the wait time for the user.

Structure of the `shutdown` File

The `shutdown` command tells the server where to look for the `shutdown` message generated by the `ftpshut` command or by the user. The `shutdown` command is used with a pathname to a shutdown file, such as:

```
shutdown /etc/uftpshutdown
```

If this file exists, the server checks the file to see when the server should shut down. The syntax of this file is as follows:

```
<year> <month> <day> <hour> <minute> <deny_offset> <disc_offset> <text>
```

`year` can be any year after 1970 (called the *epoch*), `month` is from 0–11, `hour` is 0–23, and `minute` is 0–59. `deny_offset` is the number of minutes before shutdown in which the server will disable new connections. `disc_offset` is the number of minutes before connected users are disconnected, and `text` is the message displayed to the users at login. In addition to the valid magic cookies defined in the messages section, those listed in Table 18.2 are also available.

TABLE 18.2 Magic Cookies for the `Shutdown` File

Cookie	Description
`%s`	The time the system will be shut down
`%r`	The time new connections will be denied
`%d`	The time current connections will be dropped

Configuring FTP Server File-Conversion Actions

The FTP server can convert files during transfer to compress and uncompress files automatically. Suppose that the user is transferring a file to his Microsoft Windows machine that was TARed and GZIPed on a Linux machine. If the user does not have an archive utility installed to uncompress these files, he cannot access or use the files.

As the FTP server administrator, you can configure the FTP server to automatically unarchive these files before download should the site support users who might not have unarchive capabilities. Additionally, you can configure an upload area for the users, and then configure the FTP server to automatically compress any files transferred to the server.

The structure of the format of the `ftpconversions` file is:

```
1:2:3:4:5:6:7:8
```

where `1` is the strip prefix, `2` is the strip postfix, `3` is the add-on prefix, `4` is the add-on postfix, `5` is the external command, `6` is the types, `7` is the options, and `8` is the description.

Strip Prefix

The *strip prefix* is one or more characters at the beginning of a filename that should be automatically removed by the server when the file is requested. By specifying a given prefix to strip in a conversions rule, such as `devel`, the user can request the file `devel_procman.tar.gz` by the command `get procman.tar.gz`, and the FTP server performs any other rules that apply to that file and retrieve it from the server. Although this feature is documented, as of version 2.6.2, it has yet to be implemented.

Strip Postfix

The *strip postfix* works much the same as the strip prefix, except that one or more characters are taken from the end of the filename. This feature is typically used to strip the `.gz` extension from files that have been TARed and GZIPed when the server is performing automatic decompression before sending the file to the client.

Add-On Prefix

The *add-on prefix* conversion instructs the server to insert one or more characters to a filename before it is transferred to the server or client. For example, a user requests the file `procman.tar.gz`. The server has a conversion rule to add a prefix of `gnome` to all `.tar.gz` files; therefore the server would append this string to the file before sending it to the client. The user would receive a file called `gnome_procman.tar.gz`. Keywords such as `uppercase` and `lowercase` can be used in this function to change the case of the filename for those operating systems in which case makes a difference. Similar to the strip prefix conversion, this feature is not yet implemented in version 2.6.2.

Add-On Postfix

An *add-on postfix* instructs the server to append one or more characters to the end of a filename during the transfer or reception of a file. A server can contain TARed packages of applications that are uncompressed. If an add-on postfix conversion was configured on the server, the server could compress the file, append a `.gz` extension after the file was compressed, and then send that file to the client. The server could also do the same action for uncompressed files sent to the server. This would have the effect of conserving disk space on the server.

External Command

The *external command* entries in the `ftpconversions` file contain the bulk of the FTP server conversion rules. The external command entry tells the server what should be done with a file after it is transferred to the server. The specified conversion utility can be any command on the server, although generally it is a compression utility. As the file is sent, the server passes the file through the external command. If the file is being uploaded to the server, the command needs to send the result to standard in, whereas a download sends the command to standard out. For example, here is an entry specifying the `tar` command:

```
:   : :.tar:/bin/tar -c -f - %s:T_REG¦T_DIR:O_TAR:TAR
```

The following sections describe the fields in a conversion entry.

Types

You must use the types field of the `ftpconversions` file to tell the server what types of files the conversion rules apply to. Separate the file type entries with the (¦) character, and give each `type` a value of `T_REG`, `T_ASCII`, and `T_DIR`.

`T_REG` signifies a regular file, `T_ASCII` an ASCII file, and `T_DIR` a directory. A typical entry is `T_REG ¦ T_ASCII`, which signifies a regular ASCII file.

Options

The options field informs the server what action is being done to the file. Similar to the types file, options are separated by the (¦) character. Here are the valid ranges you can assign to items in the options field:

- `O_COMPRESS` to compress the file
- `O_UNCOMPRESS` to uncompress the file
- `O_TAR` to tar the file

An example of this field is `O_COMPRESS ¦ O_TAR`, where files are both compressed and TARed.

Description

The description field allows an administrator to quickly understand what the rule is doing. This field does not have any syntax restriction, although it is usually a one-word entry—such as `TAR`, `TAR+COMPRESS`, or `UNCOMPRESS`—which is enough to get the concept across.

An Example of Conversions in Action

Crafting complex conversion entries is a task perhaps best left to the Linux/Unix expert, but the sample `ftpconversions` file included with `wu-ftpd` provides more than enough examples for the average administrator. Building your own simple conversion entry is not really too difficult, so let's examine and decode an example:

```
:.Z:  :  :/bin/compress -d -c %s:T_REG¦T_ASCII:O_UNCOMPRESS:UNCOMPRESS
```

In this example, the strip prefix (field 1) is `null` because it is not yet implemented, so this rule does not apply to prefixes. The second field of this rule contains the `.Z` postfix; therefore it deals with files that have been compressed with the `compress` utility. The rule does not address the add-on prefix or postfix, so fields 3 and 4 are `null`. Field 5, the external command field, tells the server to run the `compress` utility to decompress all files that have the `.Z` extension, as the `-d` parameter signifies. The `-c` option tells compress to write the output of the `compress` utility to the standard out, which is the server in this case. The `%s` is the name of the file against which the rule was applied. Field 6 specifies that this file is a regular file in ASCII format. Field 7, the options field, tells the server that this command uncompresses the file. Finally, the last field is a comment that gives the administrator a quick decode of what the conversion rule is doing—that is, uncompressing the file.

Examples

Several conversion rules may be specified in `wu-ftpd`'s default `ftpconversions` file. Additional examples, such as for Sun's Solaris operating system, might also be available in additional `wu-ftpd` documentation.

Using the `ftphosts` File to Allow or Deny FTP Server Connection

The purpose of the `ftphosts` file is to allow or deny specific users or addresses from connecting to the FTP server. The format of the file is the word `allow` or `deny`, optionally followed by a username, followed by an IP or a DNS address.

```
allow username address
deny username address
```

Listing 18.3 shows a sample configuration of this file.

LISTING 18.3 `ftphosts` Configuration File for Allowing or Denying Users

```
     # Example host access file
     #
     # Everything after a '#' is treated as comment,
     # empty lines are ignored
     allow tdc 128.0.0.1
     allow tdc 192.168.101.*
     allow tdc insanepenguin.net
     allow tdc *.exodous.net
     deny anonymous 201.*
deny anonymous *.pilot.net
```

The `*` is a wildcard that matches any combination of that address. For example, `allow tdc *.exodous.net` allows the user `tdc` to log in to the FTP server from any address that contains the domain name exodous.net. Similarly, the anonymous user is not allowed to access the FTP if he is coming from a 201 public class C IP address.

Changes made to your system's FTP server configuration files only become active after you restart `inetd` because configuration files are only parsed at startup. To restart `inetd` as root, issue the command `/etc/init.d/inetutils-inetd restart`. This makes a call to the same shell script that is called at system startup and shutdown for any runlevel to start or stop the `inet` daemon. `inetd` should report its status as

```
# /etc/init.d/inetutils-inetd restart
Stopping internet superserver inetd:                                          [  OK  ]
Starting internet superserver inetd:                                          [  OK  ]
```

Once restarted, the FTP server is accessible to all incoming requests.

Using Commands for Server Administration

wu-ftp provides a few commands to aid in server administration. Those commands are:

- ftpwho—Displays information about current FTP server users
- ftpcount—Displays information about current server users by class
- ftpshut—Provides automated server shutdown and user notification
- ftprestart—Provides automated server restart and shutdown message removal

Each of these commands must be executed with superuser privileges because they reference the ftpaccess configuration file to obtain information about the FTP server.

Display Information About Connected Users

The ftpwho command provides information about who are users currently connected to the FTP server. Here's the command line:

```
/usr/bin/ftpwho
```

Table 18.3 shows the format of the output ftpwho displays.

TABLE 18.3 ftpwho Fields

Name	Description
Process ID	The process ID of the FTP server process.
TTY	The terminal ID of the process. This is always a ? because the FTP daemon is not an interactive login.
Status	The status of the FTP process. The values are
	S: Sleeping
	Z: Zombie, indicating a crash
	R: Running
	N: Normal process
Time	The elapsed processor time the process has used in minutes and seconds.
Details	Tells from what host the process is connecting, the user who connected, and the currently executing command.

Listing 18.4 shows a typical output of this command. It lists the process ID for the ftp daemon handling requests, the class to which the particular user belongs, the total time connected, the connected username, and the status of the session.

In addition to the information given about each connected user, `ftpwho` also displays the total number of users connected out of any maximum that might have been set in the `ftpaccess` file. This information can be used to monitor the use of your FTP server.

You can pass one parameter to `ftpwho`. (You can find the parameter by using the `ftpwho--help` command.) The single parameter you can pass to `ftpwho` is `-v`. This parameter prints out version and licensing information for `wu-ftp`, as shown here:

```
# ftpwho
Service class all:
10447 ?        SN     0:00 ftpd: localhost: anonymous/winky@disney.com: IDLE
1 users (no maximum)
```

The output of `ftpwho`, using the `-V` option, which shows version information, is shown in Listing 18.4.

LISTING 18.4 `ftpwho -V` Command Output

```
Copyright © 1999,2000,2001 WU-FTPD Development Group.
All rights reserved.

Portions Copyright © 1980, 1985, 1988, 1989, 1990, 1991, 1993, 1994
  The Regents of the University of California.
Portions Copyright © 1993, 1994 Washington University in Saint Louis.
Portions Copyright © 1996, 1998 Berkeley Software Design, Inc.
Portions Copyright © 1989 Massachusetts Institute of Technology.
Portions Copyright © 1998 Sendmail, Inc.
Portions Copyright © 1983, 1995, 1996, 1997 Eric P.  Allman.
Portions Copyright © 1997 by Stan Barber.
Portions Copyright © 1997 by Kent Landfield.
Portions Copyright © 1991, 1992, 1993, 1994, 1995, 1996, 1997
Free Software Foundation, Inc.

Use and distribution of this software and its source code are governed
by the terms and conditions of the WU-FTPD Software License ("LICENSE").

If you did not receive a copy of the license, it may be obtained online
at http://www.wu-ftpd.org/license.html.
Version wu-2.6.2-8
```

Count the Number of Connections

`/usr/bin/ftpcount` counts the number of connected users to the FTP server and the maximum number of users allowed. This same information is found at the end of the output for the `ftpwho` command. This command takes only one parameter, `-V`, which displays the same output as the previous `ftpwho` example.

```
# ftpcount
Service class all                 -   4 users (no maximum)
```

Use `/usr/sbin/ftpshut` to Schedule FTP Server Downtime

As with any public server administration, it is always good practice to let users of the FTP server know about upcoming outages, when the server will be updated, and other relevant site information. The `ftpshut` command allows the administrator to let the FTP server do much of this automatically.

The `ftpshut` command enables the administrator to take down the FTP server at a specific time, based on some parameters passed to it. The format of the command is as follows and is documented in the `ftpshut` man page:

```
ftpshut  [ -V ] [ -l min] [ -d min] time [ warning-message ... ]
```

The `-V` parameter displays the version information of the command. The `time` parameter is the time when the `ftpshut` command will stop the FTP servers. This parameter takes either a + number for the number of minutes from the current time, or a specific hour and minute in 24-hour clock format with the syntax of *`HH:MM`*.

The `-l` parameter enables the FTP server administrator to specify how long, in minutes, before shutdown the server will disable new connections. The default for this is 10 minutes. If the time given to shut down the servers is less than 10 minutes, new connections will be disabled immediately.

The `-d` parameter is similar to the `-l` parameter, but controls when the FTP server terminates the current connections. By default, this occurs five minutes before the server shuts down. If the shutdown time is less than five minutes, the server terminates the current connections immediately.

When you execute this command, the FTP server creates a file containing the shutdown information in the location specified under the shutdown section in the `ftpaccess` file. The default configuration for this file is `/etc/shutmsg`. If you execute the `ftpshut` command with warning messages, the messages are displayed when the user logs in to the server.

```
Name (pheniox:tdc): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
```

```
230-system doing down at Mon Sep  3 06:23:00 2001
230-0 users of unlimited on pheniox.
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Here is a sample `ftpshut` command:

```
ftpshut -l 5 -d 5 +10 "system going down at %s %N users of %M on %R"
```

This command tells the FTP server to disconnect new connections in 5 minutes, drop all current connections in 5 minutes, shut down the server in 10 minutes, and display a warning message to the users at login. The message can be a mixture of text and magic cookies defined in Table 18.4. It is important to keep in mind that the message can only be a maximum of 75 characters in length. Additionally, it is not important to know how many characters the magic cookies will take because the system knows this information and truncates the message at 75 characters.

TABLE 18.4 Magic Cookies for the `ftpshut` Command

Cookie	Description
`%s`	Time the system will be shut down
`%r`	Time new connections will be denied
`%d`	Time current connections will be dropped
`%C`	Current working directory
`%E`	Server administrator's email address as specified in the `ftpaccess` file
`%F`	Available free space in the current working directories partition, in kilobytes
`%L`	Local host time
`%M`	Maximum number of allowed connections in this user class
`%N`	Current number of connections for this user class
`%R`	Remote hostname
`%T`	Local time, in the form of `Fri Aug 31 21:04:00 2001`
`%U`	Username given at login

When `ftpshut` is issued to the system, it creates a file that stores the necessary information. The `ftprestart` command removes this file for all servers, either canceling the impending shutdown or removing the shutdown file and restarting the FTP server. The `ftprestart` has only one optional argument, `-V`, to show version information.

Use `/var/log/xferlog` to View a Log of Server Transactions

The `xferlog` file gives a log of what transactions have occurred with the FTP server. Depending on the settings in the `/etc/ftpaccess` file, the contents of this file can contain the files sent or received, by whom, with a date stamp. Table 18.5 lists the fields of this file. The same information can also be found in the corresponding man page included in the `wu-ftp`.

TABLE 18.5 `/var/log/xferlog` Fields

Field	Description
current-time	Current local time in the form of *DDD MMM dd hh:mm:ss YYYY,* where *DDD* is the day of the week, *MMM* is the month, *dd* is the day of the month, *hh* is the hour, *mm* is the minutes, *ss* is the seconds, and *YYYY* is the year.
transfer-time	Total time in seconds for the transfer.
remote-host	Remote hostname.
file-size	Size of the transferred file in bytes.
filename	Name of the file.
transfer-type	A single character indicating the transfer type. The types are `a` for ASCII transfers `b` for binary transfers
special-action-flag	One or more character flags indicating any special action taken by the server. The values are `C` for compressed files `U` for uncompressed files `T` for TARed files `-` for no special action taken
direction	Indicates whether the file was sent from or received by the server.
access-mode	The way in which the user logged in to the server. The values are `a` for an anonymous guest user `g` for a guest user, corresponding to the `guestgroup` command in the `/etc/ftpaccess` file `r` for a real user on the local machine
username	If logged in as a real user, the username. If the access mode was `guest`, the password is given.
service-name	The name of the service used, usually FTP.
authentication-method	Type of authentication used. The values are `0` for none `1` for RFC931 authentication (a properly formed email address)
authenticated-user-id	This is the user ID returned to the server based on the authentication method used to access the server. An `*` is used when an authenticated user ID cannot be found.
completion-status	A single-character field indicating the status of the transfer. The values are `c` for a completed transfer `i` for an incomplete transfer

An example of this file is seen in Listing 18.5.

LISTING 18.5 Sample /var/log/xferlog File with Inbound and Outbound Logging

```
Mon Sep  3 07:13:05 2001 1 localhost.localdomain 100
 /var/ftp/pub/README b  o a testing@test.com ftp 0 * c
Mon Sep  3 02:35:35 2001 1 helios 8 /var/ftp/pub/configuration a
_ o a testing@test.com ftp 0 * c
Mon Sep  3 02:35:35 2001 1 helios 8 /var/ftp/pub/temp.txt a  o a
testing@test.com ftp 0 * c
Mon Sep  3 02:35:35 2001 1 helios 8 /var/ftp/pub/tftp-server-
0.17-14.i386.rpm a  o a testing@test.com ftp 0 * c
Mon Sep  3 02:35:35 2001 1 helios 8 /var/ftp/pub/wu-ftpd-2.6.1-
22.i386.rpm a  o a testing@test.com ftp 0 * c
```

Related Ubuntu and Linux Commands

You use these commands to install, configure, and manage FTP services in Ubuntu:

- `epiphany`—A graphical GNOME browser supporting FTP
- `ftp`—A text-based interactive FTP command
- `ftpcopy`—Copy directories and files from an FTP server
- `ftpcp`—Retrieve data from a remote FTP server, but do not overwrite existing local files
- `gftp`—A graphical FTP client for GNOME
- `konqueror`—KDE's graphical web browser
- `lftp`—An advanced text-based FTP program
- `nautilus`—Gnome's graphical file explorer and browser
- `ncftp`—A sophisticated, text-based FTP program
- `sftp`—Secure file transfer program
- `smbclient`—Samba FTP client to access SMB/CIFS resources on servers
- `vsftpd`—The Very Secure FTP daemon
- `webcam`—A webcam-oriented FTP client included with `xawtv`

Reference

- http://www.wu-ftpd.org/—Official Wu-FTP website.
- http://www.cert.org/—Computer Emergency Response Team.

- http://www.openssh.com/—The OpenSSH home page and source for the latest version of OpenSSH and its component clients, such as sftp.
- http://www.cert.org/tech_tips/anonymous_ftp_config.html—CERT Anonymous FTP configuration guidelines.
- http://vsftpd.beasts.org/—Home page for the `vsftd` FTP server.
- ftp://vsftpd.beasts.org/users/cevans/—Download site for the latest releases of the `vsftpd` server.

This page intentionally left blank

CHAPTER 19

Handling Electronic Mail

IN THIS CHAPTER

Email is still the dominant form of communication over the Internet. It is fast, free, and easy to use. However, much of what goes on behind the scenes is extremely complicated and would appear scary to anyone who does not know much about how email is handled. Ubuntu comes equipped with a number of powerful applications that will help you build anything from a small email server, right through to large servers able to handle thousands of messages.

This chapter shows you how to configure Ubuntu to act as an email server. We look at the options available in Ubuntu and examine the pros and cons of each one. You will also learn how mail is handled in Linux, and to a lesser extent, UNIX.

How Email Is Sent and Received

Email is transmitted as plain text across networks around the world using the *SMTP* protocol (*Simple Mail Transfer Protocol*). As the name implies, the protocol itself is fairly basic, and it has been extended to add further authentication and error reporting/messaging to satisfy the growing demands of modern email. *Mail transfer agents*, or *MTAs*, work in the background transferring email from server to server allowing emails to be sent all over the world. You may have come across such MTA software such as Sendmail, Postfix, Fetchmail, Exim, or Qmail.

SMTP allows each computer that the email passes through to forward it in the right direction to the final destination. When you consider the millions of email servers across the world, you have to marvel at how simple it all seems.

Here is a simplified example of how email is successfully processed and sent to its destination:

1. andrew@hudson.org composes and sends an email message to paul@hudzilla.org.
2. The MTA at hudson.org receives Andrew's email message and queues it for delivery behind any other messages that are also waiting to go out.
3. The MTA at hudson.org contacts the MTA at hudzilla.org on port 24. After hudzilla.org acknowledges the connection, the MTA at hudson.org sends the mail message. After hudzilla.org accepts and acknowledges receipt of the message, the connection is closed.
4. The MTA at hudzilla.org places the mail message into Paul's incoming mailbox; Paul is notified that he has new mail the next time he logs on.

Of course, several things can go wrong during this process. Here are a few examples:

What if Paul does not exist at hudzilla.org? In this case, the MTA at hudzilla.org will reject the email and notify the MTA at hudson.org of what the problem is. The MTA at hudson.org will then generate an email message and send it to andrew@hudson.org, informing him that no Paul exists at hudzilla.org (or perhaps just silently discard the message and give the sender no indication of the problem, depending on how the email server is configured).

What happens if hudzilla.org doesn't respond to hudson.org's connection attempts? (Perhaps the server is down for maintenance.) The MTA at hudson.org notifies the sender that the initial delivery attempt has failed. Further attempts will be made at intervals decided by the server administrator until the deadline is reached, and the sender will be notified that the mail is undeliverable.

The Mail Transport Agent

Several MTAs are available for Ubuntu, each with its pros and cons. Normally they are hidden under the skin of Ubuntu, silently moving mail between servers all over the world with need for little or no maintenance. Some MTAs are extremely powerful, being able to cope with hundreds of thousands of messages each day, whereas some are more geared toward smaller installations. Other MTAs are perhaps not as powerful, but are packed full with features. In the next section, we take a look at some of the more popular MTAs available for Ubuntu.

Postfix

Postfix has its origins as the IBM Secure Mailer, but was released to the community by IBM. Compared to Sendmail, it is much easier to administer and has a number of speed advantages. Postfix offers a pain-free replacement for Sendmail, and you are able to literally replace Sendmail with Postfix without the system breaking a sweat. In fact, the applications that rely on Sendmail will automatically use Postfix instead and carry on working correctly (because Postfix uses a Sendmail wrapper, which deceives other programs into thinking that Postfix is Sendmail). This wrapper, or more correctly interface, makes switching to Postfix extremely easy if you are already running Sendmail. Postfix also happens to be the MTA of choice for Ubuntu, so it is this one that we spend more time on later in this chapter.

For enhanced security, many Postfix processes used to use the `chroot` facility (which restricts access to only specific parts of the file system) for improved security, and there are no `setuid` components in Postfix. With the current release of Ubuntu, a `chroot` configuration *is no longer used* and is, in fact, discouraged by the Postfix author. You can manually reconfigure Postfix to a `chroot` configuration, but that is no longer supported by Ubuntu.

If you are starting from scratch, Postfix is considered a better choice than Sendmail.

Sendmail

Sendmail handles the overwhelming majority of emails transmitted over the Internet today. It is extremely popular across the Linux/UNIX/BSD world and is well supported. A commercial version is available that has a GUI interface for ease of configuration.

As well as being popular, Sendmail is particularly powerful compared to some of the other MTAs. However, it is not without its downsides, and you will find that other MTAs can handle more email per second in a larger environment. The other issue with Sendmail is that it can be extremely complicated to set it up exactly as you want it. A few books are available specifically for Sendmail, but the most popular one has more than a thousand pages, reflecting the complex nature of Sendmail configuration.

We can be thankful, however, that the default configuration for Sendmail works fine for most basic installations out of the box, making further configurations unnecessary. Even if you want to use it as a basic email server, you only need to do some minor tweaks. The level of complexity associated with Sendmail often leads to system administrators replacing it with one of the other alternatives that is easier to configure.

Qmail and Exim

Qmail is a direct competitor to Postfix but is not provided with Ubuntu. Qmail is designed to be easier to use than Sendmail, as well as faster and more secure. However, Qmail isn't a drop-in replacement for Sendmail, so migrating an existing Sendmail installation to Qmail is not quite as simple as migrating from Sendmail to Postfix. Qmail is relatively easy to administer, and it integrates with a number of software add-ons, including web mail systems and POP3 servers. Qmail is available from http://www.qmail.org/.

Exim is yet another MTA, and it is available at http://www.exim.org/. Exim is considered faster and more secure that Sendmail or Postfix, but is much different to configure that either of those. Exim and Qmail use the `maildir` format rather than `mbox`, so both are considered "NFS safe" (see the following sidebar).

MDIR Versus Mailbox

Qmail also introduced `maildir`, which is an alternative to the standard UNIX method of storing incoming mail. `maildir` is a more versatile system of handling incoming email, but it requires your email clients to be reconfigured, and it is not compatible with the traditional UNIX way of storing incoming mail. You will need to use mail programs that recognize the `maildir` format. (The modern programs do.)

The traditional `mbox` format keeps all mail assigned to a folder concatenated as a single file and maintains an index of individual emails. With `maildir`, each mail folder has three subfolders: `/cur`, `/new`, and `/tmp`. Each email is kept in a separate, unique file. If you are running a mail server for a large number of people, you should select a file system that can efficiently handle a large number of small files.

`mbox` does offer one major disadvantage. While you are accessing the monolithic `mbox` file that contains all your email, suppose that some type of corruption occurs, either to the file itself or to the index. Recovery from this problem can prove difficult. The `mbox` files are especially prone to problems if the files are being accessed over a network and can result in file corruption; one should avoid accessing `mbox` mail mounted over `NFS`, the Network File System, because file corruption can occur.

Depending on how you access your mail, `maildir` does permit the simultaneous access of `maildir` files by multiple applications; `mbox` does not.

The choice of a *mail user agent (MUA)*, or email client, also affects your choice of mail directory format. For example, the `pine` program does not cache any directory information and must reread the mail directory any time it accesses it. If you are using `pine`, `maildir` is a poor choice. More-advanced email clients perform caching, so `maildir` might be a good choice, although the email client cache can get out of synchronization. It seems that no perfect choice exists.

Ubuntu provides you with mail alternatives that have both strong and weak points. Be aware of the differences among the alternatives and frequently reevaluate your selection to make certain that it is the best one for your circumstances.

Choosing an MTA

Other MTAs are available for use with Ubuntu, but those discussed in the previous sections are the most popular. Which one should you choose? That depends on what you need to do. Postfix's main strengths is that it scales well and can handle large volumes of email at high speeds, not to mention that it is much easier to configure than the more cryptic Sendmail. However, you may find that there are specific things that you need that only Sendmail can provide. It is easy to switch between MTAs when you need to.

The Mail Delivery Agent

SMTP is a server-to-server protocol that was designed to deliver mail to systems that are always connected to the Internet. Dial-up systems only connect at the user's command; they connect for specific operations, and are frequently disconnected. To accommodate this difference, many mail systems also include a *mail delivery agent*, or *MDA*. The MDA transfers mail to systems without permanent Internet connections. The MDA is similar to an MTA (see the following note), but does not handle deliveries between systems and does not provide an interface to the user.

NOTE

Procmail or Spamassassin are examples of MTAs; both provide filtering services to the MTA while they store messages locally and then make them available to the MUA or email client for reading by the user.

The MDA uses the POP3 or IMAP protocols for this process. In a manner similar to a post office box at the post office, POP3 and IMAP implement a "store and forward" process that alleviates the need to maintain a local mail server if all you want to do is read your mail. For example, dial-up Internet users can intermittently connect to their ISP's mail server to retrieve mail using Fetchmail—the MDA recommended by Ubuntu (see the section "Using Fetchmail to Retrieve Mail" later in this chapter).

The Mail User Agent

The *mail user agent*, or *MUA*, is another necessary part of the email system. The MUA is a mail client, or mail reader, that allows the user to read and compose email and provides the user interface. (It is the email application itself that most users are familiar with as "email.") Some popular UNIX command-line MUAs are `elm`, `pine`, and `mutt`. Ubuntu also provides modern GUI MUAs: Evolution, Thunderbird, Mozilla Mail, Balsa, Sylpheed, and KMail. For comparison, common non-UNIX MUAs are Microsoft Outlook, Outlook Express, Pegasus, Eudora, and Netscape Messenger.

The Microsoft Windows and Macintosh MUAs often include some MTA functionality; UNIX does not. For example, Microsoft Outlook can connect to your Internet provider's mail server to send messages. On the other hand, UNIX MUAs generally rely on an external MTA such as Sendmail. This might seem like a needlessly complicated way to do things, and it is if used to connect a single user to her ISP. For any other situation, however, using an external MTA allows you much greater flexibility because you can use any number of external programs to handle and process your email functions and customize the service. Having the process handled by different applications gives you great control over how you provide email service to users on your network, as well as to individual and *small office/home office (SOHO)* users.

For example, you could do the following:

- Use Evolution to read and compose mail
- Use Sendmail to send your mail
- Use `xbiff` to notify you when you have new mail
- Use Fetchmail to retrieve your mail from a remote mail server

- Use Procmail to automatically sort your incoming mail based on sender, subject, or many other variables
- Use Spamassassin to eliminate the unwanted messages before you read them

Basic Postfix Configuration and Operation

Because Postfix is the Ubuntu-recommended MTA, the following sections provide a brief explanation and examples for configuring and operating your email system. As mentioned earlier, however, Postfix is an extremely complex program with many configuration options. As such, this chapter only covers some of the basics. For more information on Postfix, as well as other MTAs, see the "Reference" section at the end of this chapter.

Postfix configuration is handled by files in the `/etc/postfix` directory with much of the configuration being handled by the file `main.cf`. The actual syntax of the configuration file, `main.cf`, is fairly easy to read (see the following example):

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete version
# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree:${queue_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${queue_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

myhostname = optimus
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
```

```
mydestination = optimus, localhost.localdomain, , localhost
relayhost =
mynetworks = 127.0.0.0/8
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
```

Complicated email server setup is beyond the scope of this book, and we would point you in the direction of *Postfix: The Definitive Guide* by Dent. This is a great reference, and rather unusual because it only runs to just under 300 pages. However, if you want to know something about Postfix, this is the book to read.

However, the following five sections address some commonly used advanced options.

Configuring Masquerading

Sometimes you might want to have Postfix masquerade as a host other than the actual hostname of your system. Such a situation could occur if you have a dial-up connection to the Internet and your ISP handles all your mail for you. In this case, you will want Postfix to masquerade as the domain name of your ISP. For example

```
masquerade_domains = hudson.com
```

will strip any messages that come from `andrew.hudson.com` to just `hudson.com`.

Using Smart Hosts

If you do not have a full-time connection to the Internet, you will probably want to have Postfix send your messages to your ISP's mail server and let it handle delivery for you. Without a full-time Internet connection, you could find it difficult to deliver messages to some locations (such as some underdeveloped areas of the world where email services are unreliable and sporadic). In those situations, you can configure Postfix to function as a smart host by passing email on to another sender instead of attempting to deliver the email directly. You can use a line such as the following in the `main.cf` file to enable a smart host:

```
relayhost = mail.isp.net
```

This line causes Postfix to pass any mail it receives to the server `mail.isp.net` rather than attempt to deliver it directly. Smart hosting will not work for you if your ISP, like many others, blocks any mail relaying. Some ISPs block relaying because it is frequently used to disseminate spam.

Setting Message Delivery Intervals

As mentioned earlier, Postfix typically attempts to deliver messages as soon as it receives them, and again at regular intervals after that. If you have only periodic connections to the Internet, as with a dial-up connection, you likely would prefer that Sendmail hold all messages in the queue and attempt to deliver them whenever you connect to your ISP. You can configure Postfix to do so by adding the following line to `/etc/ppp/peers/ppp0`:

```
/usr/sbin/sendmail =q
```

This line causes Postifix to automatically send all mail when connecting to your ISP.

However, Postfix will still attempt to send mail regardless of whether the computer is on or off line, meaning that your computer may dial out just to send email. To disable this, you need to enter the following line into `mail.cf`:

```
defer_transports = smtp
```

This stops any unwanted telephone calls from being placed!

TIP

If you use networking over a modem, there is a configuration file for `pppd` called `ppp0`, which is located in `/etc/ppp/peers`. Any commands in this file automatically run each time the PPP daemon is started. You can add the line `sendmail -q` to this file to have your mail queue automatically processed each time you dial up your Internet connection.

Mail Relaying

By default, Postfix will not relay mail that did not originate from the local domain. This means that if a Postfix installation running at hudson.org receives mail intended for hudzilla.org, and that mail did not originate from hudson.org, the mail will be rejected and will not be relayed. If you want to allow selected domains to relay through you, add an entry for the domain to the `main.cf` file like so:

```
mynetworks = 192.168.2.0/24, 10.0.0.2/24, 127.0.0.0/8
```

The IP address needs to be specified in CIDR format. For a handy calculator, head on over to http://www.subnet-calculator/cidr.php. You must restart Postfix for this change to take effect.

CAUTION

You need a good reason to relay mail; otherwise, do not do it. Allowing all domains to relay through you will make you a magnet for spammers who will use your mail server to send spam. This can lead to your site being blacklisted by many other sites, which then will not accept any mail from you or your site's users—even if the mail is legitimate!

Forwarding Email with Aliases

Aliases allow you to have an infinite number of valid recipient addresses on your system, without having to worry about creating accounts or other support files for each address. For example, most systems have "postmaster" defined as a valid recipient, but do not have an actual login account named "postmaster." Aliases are configured in the file `/etc/aliases`. Here is an example of an alias entry:

```
postmaster: root
```

This entry forwards any mail received for "postmaster" to the root user. By default, almost all the aliases listed in the `/etc/aliases` file forward to root.

CAUTION

Reading email as root is a security hazard; a malicious email message can exploit an email client and cause it to execute arbitrary code as the user running the client. To avoid this danger, you can forward all of root's mail to another account and read it from there. You can choose one of two ways for doing this.

You can add an entry to the `/etc/aliases` file that sends root's mail to a different account. For example, `root: foobar` would forward all mail intended for root to the account foobar.

The other way is to create a file named `.forward` in root's home directory that contains the address that the mail should forward to.

Anytime you make a change to the `/etc/aliases` file, you will need to rebuild the aliases database before that change will take effect. This is done with the following:

```
 $ sudo newaliases
```

Using Fetchmail to Retrieve Mail

SMTP is designed to work with systems that have a full-time connection to the Internet. What if you are on a dial-up account? What if you have another system store your email for you and then you log in to pick it up once in awhile? (Most users who are not setting up servers will be in this situation.) In this case, you cannot easily receive email using SMTP, and you need to use a protocol, such as POP3 or IMAP, instead.

NOTE

Remember when we said that some mail clients can include some MTA functionality? Microsoft Outlook and Outlook Express can be configured to use SMTP and, if you use a dial-up connection, will offer to start the connection and then use SMTP to send your mail, so a type of MTA functionality is included in those mail clients.

Unfortunately, many MUAs do not know anything about POP3 or IMAP. To eliminate that problem, you can use a program called Fetchmail to contact mail servers using POP3 or IMAP, download mail off the servers, and then inject those messages into the local MTA just as if they had come from a standard SMTP server. The following sections explain how to install, configure, and use the Fetchmail program.

Installing Fetchmail

Similar to other packages, Fetchmail can be installed using either `synaptic` or `apt-get`.

You can get the latest version of Fetchmail at http://www.catb.org/~esr/fetchmail.

Configuring Fetchmail

After you have installed Fetchmail, you must create the file `.fetchmailrc` in your home directory, which provides the configuration for the Fetchmail program.

You can create and subsequently edit the `.fetchmailrc` file by using any text editor. The configuration file is straightforward and quite easy to create; the following sections explain the manual method for creating and editing the file. The information presented in the following sections does not discuss all the options available in the `.fetchmailrc` file, but covers the most common ones needed to get a basic Fetchmail installation up and running. You must use a text editor to create the file to include entries like the ones shown as examples—modified for your personal information, of course. For advanced configuration, see the man page for Fetchmail. The man page is well written and documents all the configuration options in detail.

CAUTION

The `.fetchmailrc` file is divided into three different sections: global options, mail server options, and user options. It is important that these sections appear in the order listed. Do not add options to the wrong section. Putting options in the wrong place is one of the most common problems that new users make with Fetchmail configuration files.

Configuring Global Options

The first section of `.fetchmailrc` contains the global options. These options affect all the mail servers and user accounts that you list later in the configuration file. Some of these global options can be overridden with local configuration options, as you learn later in this section. Here is an example of the options that might appear in the global section of the `.fetchmailrc` file:

```
set daemon 600
set postmaster foobar
set logfile ./.fetchmail.log
```

The first line in this example tells Fetchmail that it should start in daemon mode and check the mail servers for new mail every 600 seconds, or 10 minutes. Daemon mode means that after Fetchmail starts, it will move itself into the background and continue running. Without this line, Fetchmail would check for mail once when it started and would then terminate and never check again.

The second option tells Fetchmail to use the local account foobar as a last-resort address. In other words, any email that it receives and cannot deliver to a specified account should be sent to foobar.

The third line tells Fetchmail to log its activity to the file `./.fetchmail.log`. Alternatively, you can use the line `set syslog`—in which case, Fetchmail will log through the `syslog` facility.

Configuring Mail Server Options

The second section of the `.fetchmailrc` file contains information on each of the mail servers that should be checked for new mail. Here is an example of what the mail section might look like:

```
poll mail.samplenet.org
proto pop3
no dns
```

The first line tells Fetchmail that it should check the mail server mail.samplenet.org at each poll interval that was set in the global options section (which was 600 seconds in our example). Alternatively, the first line can begin with `skip`. If a mail server line begins with `skip`, it will not be polled as the poll interval, but will only be polled when it is specifically specified on the Fetchmail command line.

The second line specifies the protocol that should be used when contacting the mail server. In this case, we are using the POP3 protocol. Other legal options are IMAP, APOP, and KPOP. You can also use `AUTO` here—in which case, Fetchmail will attempt to automatically determine the correct protocol to use with the mail server.

The third line tells Fetchmail that it should not attempt to do a *Dynamic Name Server (DNS)* lookup. You will probably want to include this option if you are running over a dial-up connection.

Configuring User Accounts

The third and final section of `.fetchmailrc` contains information about the user account on the server specified in the previous section. Here is an example:

```
user foobar
pass secretword
fetchall
flush
```

The first line, of course, simply specifies the username that is used to log in to the email server, and the second line specifies the password for that user. Many security conscious people cringe at the thought of putting clear-text passwords in a configuration file, and they should if it is group or world readable. The only protection for this information is to make certain that the file is readable only by the owner; that is, with file permissions of `600`.

The third line tells Fetchmail that it should fetch all messages from the server, even if they have already been read.

The fourth line tells Fetchmail that it should delete the messages from the mail server after it has completed downloading them. This is the default, so we would not really have to specify this option. If you want to leave the messages on the server after downloading them, use the option `no flush`.

The configuration options you just inserted configured the entire `.fetchmailrc` file to look like this:

```
set daemon 600
set postmaster foobar
set logfile ./.fetchmail.log

poll mail.samplenet.org
proto pop3
no dns

user foobar
pass secretword
fetchall
flush
```

This file tells Fetchmail to do the following:

- Check the POP3 server mail.samplenet.org for new mail every 600 seconds.
- Log in using the username foobar and the password secretword.
- Download all messages off the server.
- Delete the messages from the server after it has finished downloading them.
- Send any mail it receives that cannot be delivered to a local user to the account foobar.

As mentioned before, many more options can be included in the `.fetchmailrc` file than are listed here. However, these options will get you up and running with a basic configuration.

For additional flexibility, you can define multiple `.fetchmailrc` files to retrieve mail from different remote mail servers while using the same Linux user account. For example, you can define settings for your most often used account and save them in the default `.fetchmailrc` file. Mail can then quickly be retrieved like so:

```
$ fetchmail –a
1 message for andrew at mail.ourphotos.me.uk (1108 octets).
reading message 1 of 1 (1108 octets) . flushed
```

By using Fetchmail's `-f` option, you can specify an alternative resource file and then easily retrieve mail from another server, as follows:

```
$ fetchmail –f .myothermailrc
2 messages for andrew at anrew.hudson.com (5407 octets).
reading message 1 of 2 (3440 octets) ... flushed
reading message 2 of 2 (1967 octets) . flushed
You have new mail in /var/spool/mail/andrew
```

By using the `-d` option, along with a time interval (in seconds), you can use Fetchmail in its daemon, or background mode. The command will launch as a background process and retrieve mail from a designated remote server at a specified interval. For more-advanced options, see the Fetchmail man page, which is well written and documents all options in detail.

CAUTION

Because the `.fetchmailrc` file contains your mail server password, it should be readable only by you. This means that it should be owned by you and should have permissions no greater than `600`. Fetchmail will complain and refuse to start if the `.fetchmailrc` file has permissions greater than this.

Choosing a Mail Delivery Agent

Because of the modular nature of mail handling, it is possible to use multiple applications to process mail and accomplish more than simply deliver it. Getting mail from the storage area and displaying it to the user is the purpose of the MDA. MDA functionality can be found in some of the mail clients (MUAs), which can cause some confusion to those still unfamiliar with the concept of UNIX mail. As an example, the Procmail MDA provides filtering based on rulesets; KMail and Evolution, both MUAs, provide filtering, but the MUAs `pine`, `mutt`, and Balsa do not. Some MDAs perform simple sorting, and other MDAs are designed to eliminate unwanted emails, such as spam and viruses.

You would choose an MDA based on what you want to do with your mail. We will look at five MDAs that offer functions you might find useful in your particular situation. If you have simple needs (just organizing mail by rules), one of the MUAs that offers filtering might be better for your needs. Ubuntu provides the Evolution MUA as the default selection (and it contains some MDA functionality as previously noted), so try that first and see whether it meets your needs. If not, investigate one of the following MDAs provided by Ubuntu.

Unless otherwise noted, all the MDA software is provided through `synaptic` and `apt-get`. Chapter 31, “Managing Software,” details the general installation of any software.

Procmail

As a tool for advanced users, the Procmail application acts as a filter for email as it is retrieved from a mail server. It uses rulesets (known as *recipes*) as it reads each email message. No default configuration is provided; you must manually create a `~/.procmail` file for each user, or each user can create her own.

There is no systemwide default configuration file. The creation of the rulesets is not trivial and requires an understanding of the use of regular expressions that is beyond the scope of this chapter. Ubuntu does provide three examples of the files in `/usr/share/doc/procmail/examples`, as well as a fully commented example in the `/usr/share/doc/procmail` directory, which also contains a `README` and `FAQ`. Details for the rulesets can be found in the man page for Procmail as well as the man pages for `procmailrc`, `procmailsc`, and `procmailex`, which contain examples of Procmail recipes.

Spamassassin

If you have used email for any length of time, you have likely been subjected to spam—unwanted email sent to thousands of people at the same time. Ubuntu provides an MDA named Spamassassin to assist you in reducing and eliminating unwanted emails. Easily integrated with Procmail and Sendmail, it can be configured for both systemwide and individual use. It uses a combination of rulesets and blacklists (Internet domains known to mail spam).

Enabling Spamassassin is simple. You must first have installed and configured Procmail. The `README` file found in `/usr/share/doc/spamassasin` provides details on configuring the `.procmail` file to process mail through Spamassassin. It will tag probable spam with a unique header; you can then have Procmail filter the mail in any manner you choose. One interesting use of Spamassasin is to use it to tag email received at special email accounts established solely for the purpose of attracting spam. This information is then shared with the Spamassassin site where these “spam trap” generated hits help the authors fine-tune the rulesets.

Squirrelmail

Perhaps you do not want to read your mail in an MUA. If you use your web browser often, it might make sense to read and send your mail via a web interface, such as the one used by Hotmail or Yahoo! Mail. Ubuntu provides Squirrelmail for just that purpose. Squirrelmail is written in the PHP 4 language and supports IMAP and SMTP with all pages rendering in HTML 4.0 without using Java. It supports MIME attachments, as well as an address book and folders for segregating email.

You must configure your web server to work with PHP 4. Detailed installation instructions can be found in `/usr/share/doc/squirrelmail/INSTALL`. Once configured, point your web browser to http://www.yourdomain.com/squirellmail/ to read and send email.

Virus Scanners

Although the currently held belief is that Linux is immune to email viruses targeted at Microsoft Outlook users, it certainly makes no sense for UNIX mail servers to permit infected email to be sent through them. Although Ubuntu does not provide a virus scanner, one of the more popular of many such scanners is MailScanner, available from http://www.sng.ecs.soton.ac.uk/mailscanner/; a Ubuntu DEB package is available as well as the source code. It supports Sendmail and Exim, but not Postfix or Qmail. A hunt through `synaptic` using the search terms "virus" and "scanner" should yield some other options for you to try out.

Mail Daemons

Biff is a small daemon that monitors your mail folder and notifies you when a message has been placed there. It is common to include `biff y` in the `.login` or `.profile` files to automatically start it upon user login if you want to use Biff.

NOTE

Autoresponders automatically generate replies to received messages; they are commonly used to notify others that the recipient is out of the office. Mercifully, Ubuntu does not include one, but you can find and install an autoresponder at Freshmeat.net. If you are subscribed to a mailing list, be aware that automatic responses from your account can be very annoying to others on the list. Please unsubscribe from mail lists before you leave the office with your autoresponder activated.

Alternatives to Microsoft Exchange Server

One of the last areas in which a Microsoft product has yet to be usurped by open-source software is a replacement for Microsoft Exchange Server. Many businesses use Microsoft Outlook and Microsoft Exchange Server to access email, as well as to provide calendaring, notes, file sharing, and other collaborative functions. General industry complaints about Exchange Server center around scalability, administration (backup and restore in particular), and licensing fees.

A "drop-in" alternative needs to have compatibility with Microsoft Outlook because it's intended to replace Exchange Server in an environment in which there are Microsoft desktops in existence using Outlook. A "work-alike" alternative provides similar features to Exchange Server, but does not offer compatibility with the Microsoft Outlook client itself; the latter is typical of many of the open-source alternatives.

Several "drop-in" alternatives exist, none of which are fully open source because some type of proprietary connector is needed to provide the services to Microsoft Outlook clients (or provide Exchange services to the Linux Evolution client). For Outlook compatibility, the key seems to be the realization of a full, open implementation of *MAPI*, the Microsoft *Messaging Application Program Interface*. That goal is going to be difficult to achieve because MAPI is a poorly documented Microsoft protocol. For Linux-only solutions, the missing ingredient for many alternatives is a usable group calendaring/scheduling system similar in function to that provided by Exchange Server/Outlook.

Of course, independent applications for these functions abound in the open-source world, but one characteristic of "groupware" is its central administration; another is that all components can share information.

The following sections examine several of the available servers, beginning with Microsoft Exchange Server itself and moving toward those applications that have increasing incompatibility with it. None of these servers are provided with Ubuntu.

Microsoft Exchange Server/Outlook Client

Exchange Server and Outlook seem to be the industry benchmark because of their widespread deployment. They offer a proprietary server providing email, contacts, scheduling, public folders, task lists, journaling, and notes using Microsoft Outlook as the client and MAPI as the API. If you consider what Microsoft Exchange offers as the "full" set of features, no other replacement offers 100% of the features exactly as provided by Microsoft Exchange Server—even those considered "drop-in" replacements. The home page for the Microsoft Exchange server is http://www.microsoft.com/exchange/.

CommuniGate Pro

CommuniGate Pro is a proprietary, drop-in alternative to Microsoft Exchange Server, providing, email, webmail, LDAP directories, a web server, file server, contacts, calendaring (third party), Voice over IP, and a list server. The CommuniGate Pro MAPI Connector provides access to the server from Microsoft Outlook and other MAPI-enabled clients. The home page for this server is http://www.stalker.com/.

Oracle Collaboration Suite

This is probably the closest that you will get to an Exchange replacement, allowing you to collaborate by instant messaging, email, sharing files (workspaces), calendaring, and other tools. It provides an Outlook Connector for users who have to have the familiarity of Outlook. OCS is available for Linux platforms and its homepage is http://www.oracle.com/collabsuite/.

Bynari

Bynari provides a proprietary group of servers to act as a drop-in replacement for Microsoft Exchange Server for email, calendaring, public folders, scheduling, address book, webmail, and contacts. Although it runs on Linux, it offers no Linux clients although it can be used with Evolution and Thunderbird, and the connector provides services to Microsoft Outlook only. The home page is http://www.bynari.net/.

Open-Xchange

Open-Xchange has a great pedigree having been owned and developed by Novell/SUSE until being spun off by itself into its own company. Working with open standards, it provides a number of collaboration options and is firmly based on Linux. It can work with a wide variety of protocols, making it one of the best connected suites available. You can get the open source version at http://www.open-xchange.com.

phpgroupware

phpgroupware is an open-source application written in PHP (and used with MySQL or postgresql plus a web server and an IMAP mail server). phpgroupware provides a web-based calendar, task list, address book, email, news headlines, and a file manager. Its modular nature enables you to plug in various components (around 50 at the last count) as you need them. The home page is http://www.phpgroupware.org/.

PHProjekt

PHProjekt is open-source software written in PHP (used with MySQL, postgresql, Oracle, Informix, or MS-sql). PHProjekt provides calendaring, contact manager, time card system, project management, online chat, threaded discussion forum, trouble ticket system, email, public files, notes bookmarks, voting system, task lists, reminders, site search, and integration with the PostNuke news site application. It provides no Exchange/Outlook compatibility whatsoever. The home page is http://www.PHProjekt.com/.

Horde

Horde is a PHP-based application framework. When combined with an HTTP server (Apache, Microsoft IIS, Netscape) and MySQL database, IMP/Horde offers modules that provide webmail, contact manager, calendar, CVS viewer, file manager, time tracking, email filter rules manager, notes, tasks, chat, newsgroups, forms, bug tracking, FAQ repository, and presentations. The home page is http://www.horde.org/.

Relevant Ubuntu and Linux Commands

You will use the following commands to manage electronic mail in Ubuntu:

`balsa`—A GNOME mail user agent for X

`biff`—A console-based mail notification utility

`evolution`—A comprehensive and capable Ximian GNOME mail PIM for X

`fetchmail`—A console-based and daemon-mode mail retrieval command for Linux

`fetchmailconf`—A graphical `fetchmail` configuration client for X

`kmail`—A graphical mail user client for KDE and X

`korn`—A `biff` applet for KDE and X

`mutt`—A console-based mail user agent

`sendmail`—A comprehensive mail transport agent for UNIX and Linux

`xbiff`—A mail notification X client

Reference

The following references are recommended reading for email configuration. Of course, not all references will apply to you. Select the ones that apply to the email server that you are using.

Web Resources

- http://www.sendmail.org/—This is the Sendmail home page. Here you will find configuration information and FAQs regarding the Sendmail MTA.
- http://www.postfix.org/—This is the Postfix home page. If you are using the Postfix MTA, you can find documentation and sample configurations at this site.
- http://www.qmail.org/—This is the home page for the Qmail MTA. It contains documentation and links to other resources on Qmail.
- http://www.linuxgazette.com/issue35/jao.html—IMAP on Linux: A Practical Guide. The Internet Mail Access Protocol allows a user to access his email stored on a remote server rather than a local disk.
- http://www.imap.org/about/whatisIMAP.html—This page describes IMAP.
- http://www.rfc-editor.org/—A repository of RFCs—Request for Comments—that define the technical "rules" of modern computer usage.
- http://www.procmail.org/—The Procmail home page.
- http://www.moongroup.com/docs/procmail/—The Procmail FAQ, which includes suggestions for troubleshooting problems. The page also references a page of links that, in turn, reference other link collections as well as tidbits from the Procmail mail list.
- http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Qmail-VMailMgr-Courier-imap-HOWTO.html—If you want some help configuring a mail system based on the lesser-used applications, this HOWTO will help.

Books

- *Sendmail* (O'Reilly Publishing)—This is the de facto standard guide for everything Sendmail. It is loaded with more than 1,000 pages, which gives you an idea of how complicated Sendmail really is.
- *Postfix* (Sams Publishing)—An excellent book from Sams Publishing that covers the Postfix MTA.
- *Running Qmail* (Sams Publishing)—This is similar to the Postfix book from Sams Publishing except that it covers the Qmail MTA.

CHAPTER 20

Proxying and Reverse Proxying

IN THIS CHAPTER

You can never have enough of two things in this world: time and bandwidth. Ubuntu comes with a proxy server—Squid—that enables you to cache web traffic on your server so that websites load faster and users consume less bandwidth.

What Is a Proxy Server?

A *proxy server* lies between client machines—the desktops in your company—and the Internet. As clients request websites, they do not connect directly to the Web and send the HTTP request. Instead, they connect to the local proxy server. The proxy then forwards their request on to the Web, retrieves the result, and hands it back to the client. At its simplest, a proxy server really is just an extra layer between client and server, so why bother?

The three main reasons for deploying a proxy server are as follows:

- **Content control**—You want to stop people whiling away their work hours reading the news or downloading MP3s.
- **Speed**—You want to cache common sites to make the most of your bandwidth.
- **Security**—You want to monitor what people are doing.

Squid is capable of all of these and more.

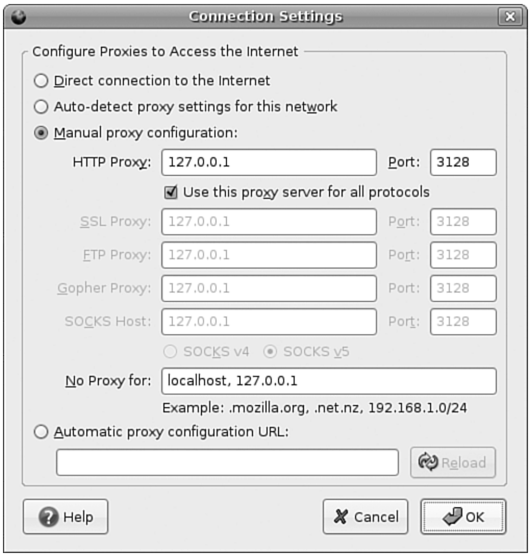
Installing Squid

Like most Ubuntu software, Squid installation is handled through `synaptic`. After Squid is installed, it is automatically enabled for each boot up. You can check this by running `ps aux ¦ grep squid` when the machine boots. If for some reason you see nothing there, run `/etc/init.d/squid start`.

Configuring Clients

Before you configure your new Squid server, set up the local web browser to use it for its web access. Doing so enables you to test your rules as you are working with the configuration file.

To configure Firefox, select Preferences from the Edit menu. From the dialog that appears, click the Connection Settings button (near the bottom on the General tab) and select the option Manual Proxy Configuration. Check the box beneath it, Use the Same Proxy for All Protocols; then enter **`127.0.0.1`** as the IP address and **`3128`** as the port number. See Figure 20.1 for how this should look. If you are configuring a remote client, specify the IP address of the Squid server rather than 127.0.0.1.



Connection Settings

Configure Proxies to Access the Internet

- ○ Direct connection to the Internet
- ○ Auto-detect proxy settings for this network
- [✓ Manual proxy configuration:]

HTTP Proxy:	127.0.0.1	Port:	3128
	☑ Use this proxy server for all protocols		
SSL Proxy:	127.0.0.1	Port:	3128
FTP Proxy:	127.0.0.1	Port:	3128
Gopher Proxy:	127.0.0.1	Port:	3128
SOCKS Host:	127.0.0.1	Port:	3128
	SOCKS v4 [✓ SOCKS v5]		
No Proxy for:	localhost, 127.0.0.1		

Example: .mozilla.org, .net.nz, 192.168.1.0/24

- ○ Automatic proxy configuration URL:

FIGURE 20.1 Setting up Firefox to use 127.0.0.1 routes all its web requests through Squid.

For Konqueror, go to the Settings menu and select Configure Konqueror. From the left tabs, scroll down to Proxy, select Manually Specify the Proxy Settings, and then click Setup. Enter **`127.0.0.1`** as the proxy IP address and **`3128`** as the port. As with Firefox, if you are configuring a remote client, specify the IP address of the Squid server rather than 127.0.0.1.

Internet Explorer proxy settings are in Tools/Internet Options. From the Connections tab, click the LAN Settings button and enable the Use a Proxy Server for Your LAN option. Enter the address as the IP of your Squid machine, and then specify **`3128`** as the port.

Access Control Lists

The main Squid configuration file is `/etc/squid/squid.conf`, and the default Ubuntu configuration file is full of comments to help guide you. The default configuration file allows full access to the local machine but denies the rest of your network. This is a secure place to start; we recommend you try all the rules on yourself (`localhost`) before rolling them out to other machines.

Before you start, open two terminal windows. In the first, change to the directory `/var/log/squid` and run this command:

```
sudo tail -f access.log cache.log
```

That reads the last few lines from both files and (thanks to the `-f` flag) follows them so that any changes appear in there. This allows you to watch what Squid is doing as people access it. We refer to this window as the "log window," so keep it open. In the other window (again, with `sudo`), bring up the file `/etc/squid/squid.conf` in your favorite editor. This window is referred to as the "config editor," and you should keep it open, too.

To get started, search for the string `acl all`—this brings you to the access control section, which is where most of the work needs to be done. You can configure a lot elsewhere, but unless you have unusual requirements, you can leave the defaults in place.

NOTE

The default port for Squid is `3128`, but you can change that by editing the `http_port` line. Alternatively, you can have Squid listen on multiple ports by having multiple `http_port` lines: **`80`**, **`8000`**, and **`8080`** are all popular ports for proxy servers.

The `acl` lines make up your *access control lists (ACLs)*. The first 16 or so define the minimum recommended configuration that set up ports to listen to, and so on. You can safely ignore these. If you scroll down further (past another short block of comments), you come to the `http_access` lines, which are combined with the `acl` lines to dictate who can do what. You can (and should) mix and match `acl` and `http_access` lines to keep your configuration file easy to read.

Just below the first block of `http_access` lines is a comment like `# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS`. This is just what we are going to do. First, though, scroll just a few lines further; you should see these two lines:

```
http_access allow localhost
http_access deny all
```

They are self-explanatory: The first says, "Allow HTTP access to the local computer, but deny everyone else." This is the default rule, as mentioned earlier. Leave that in place for now, and run `service squid start` to start the server with the default settings. If you have not yet configured the local web browser to use your Squid server, do so now so that you can test the default rules.

In your web browser (Firefox is assumed from here on, but it makes little difference), go to the URL http://www.ubuntulinux.org. You should see it appear as normal in the browser, but in the log window you should see a lot of messages scroll by as Squid downloads the site for you and stores it in its cache. This is all allowed because the default configuration allows access to the `localhost`.

Go back to the config editor window and add this before the last two `http_access` lines:

```
http_access deny localhost
```

So, the last three lines should look like this:

```
http_access deny localhost
http_access allow localhost
http_access deny all
```

Save the file and quit your editor. Then, run this command:

```
kill -SIGHUP `cat /var/run/squid.pid`
```

That looks for the *process ID (PID)* of the Squid daemon and then sends the `SIGHUP` signal to it, which forces it to reread its configuration file while running. You should see a string of messages in the log window as Squid rereads its configuration files. If you now go back to Firefox and enter a new URL, you should see the Squid error page informing you that you do not have access to the requested site.

The reason you are now blocked from the proxy is because Squid reads its ACL lines in sequence, from top to bottom. If it finds a line that conclusively allows or denies a request, it stops reading and takes the appropriate action. So, in the previous lines, `localhost` is being denied in the first line and then allowed in the second. When Squid sees `localhost` asking for a site, it reads the `deny` line first and immediately sends the error page—it does not even get to the `allow` line. Having a `deny all` line at the bottom is highly recommended so that only those you explicitly allow are able to use the proxy.

Go back to editing the configuration file and remove the `deny localhost` and `allow localhost` lines. This leaves only `deny all`, which blocks everyone (including the `localhost`) from accessing the proxy. Now we are going to add some conditional allow statements: We want to allow `localhost` only if it fits certain criteria.

Defining access criteria is done with the `acl` lines, so above the `deny all` line, add this:

```
acl newssites dstdomain news.bbc.co.uk slashdot.org
http_access allow newssites
```

The first line defines an access category called `newssites`, which contains a list of domains (`dstdomain`). The domains are news.bbc.co.uk and slashdot.org, so the full line reads, “Create a new access category called newssites, that should filter on domain, and contain the two domains listed.” It does *not* say whether access should be granted or denied to that category; that comes in the next line. The line `http_access allow newssites` means, “Allow access to the category `newssites` with no further restrictions.” It is not limited to `localhost`, which means that applies to every computer connecting to the proxy server.

Save the configuration file and rerun the `kill -SIGHUP` line from before to restart Squid; then go back to Firefox and try loading http://www.ubuntulinux.org. You should see the same error as before because that was not in our `newssites` category. Now try http://news.bbc.co.uk, and it should work. However, if you try http://www.slashdot.org, it will *not* work, and you might also have noticed that the images did not appear on the BBC News website either. The problem here is that specifying slashdot.org as the website is specific: It means that http://slashdot.org will work, whereas http://www.slashdot.org will not. The BBC News site stores its images on the site http://newsimg.bbc.co.uk, which is why they do not appear.

Go back to the configuration file, and edit the `newssites` ACL to this:

```
acl newssites dstdomain .bbc.co.uk .slashdot.org
```

Putting the period in front of the domains (and in the BBC’s case, taking the `news` off, too) means that Squid will allow any subdomain of the site to work, which is usually what you will want. If you want even more vagueness, you can just specify `.com` to match `*.com` addresses.

Moving on, you can also use time conditions for sites. For example, if you want to allow access to the news sites in the evenings, you can set up a time category using this line:

```
acl freetime time MTWHFAS 18:00-23:59
```

This time, the category is called `freetime` and the condition is `time`, which means we need to specify what time the category should contain. The seven characters following that are the days of the week: Monday, Tuesday, Wednesday, tHursday, Friday, sAturday, and Sunday. Thursday and Saturday use capital *H* and *A* so they do not clash with Tuesday and Sunday.

With that category defined, you can change the `http_access` line to include it, like this:

```
http_access allow newssites freetime
```

For Squid to allow access now, it must match both conditions—the request must be for either *.bbc.co.uk or slashdot.org, and during the time specified. If either condition does not match, the line is not matched and Squid continues looking for other matching rules beneath it. The times you specify here are inclusive on both sides, which means users in the `freetime` category will be able to surf from 18:00:00 until 23:59:59.

You can add as many rules as you like, although you should be careful to try to order them so that they make sense. Keep in mind that all conditions in a line must be matched for the line to be matched. Here is a more complex example:

- You want a category `newssites` that contains serious websites people need for their work.
- You want a category `playsites` that contains websites people do not need for their work.
- You want a category `worktime` that stretches from 09:00 to 18:00.
- You want a category `freetime` that stretches from 18:00 to 20:00, when the office closes.
- You want people to be able to access the news sites, but not the play sites, during working hours.
- You want people to be able to access both the news sites and the play sites during the free time hours.

To do that, you need the following rules:

```
acl newssites dstdomain .bbc.co.uk .slashdot.org
acl playsites dstdomain .tomshardware.com ubuntulinux.org
acl worktime time MTWHF 9:00-18:00
acl freetime time MTWHF 18:00-20:00
http_access allow newssites worktime
http_access allow newssites freetime
http_access allow playsites freetime
```

NOTE

The letter `D` is equivalent to `MTWHF` in meaning “all the days of the working week.”

Notice that there are two `http_access` lines for the `newssites` category: one for `worktime` and one for `freetime`. This is because all the conditions must be matched for a line to be matched. Alternatively, you can write this:

```
http_access allow newssites worktime freetime
```

However, if you do that and someone visits news.bbc.co.uk at 2:30 p.m. (14:30) on a Tuesday, Squid will work like this:

- Is the site in the `newssites` category? Yes, continue.
- Is the time within the `worktime` category? Yes, continue.
- Is the time within the `freetime` category? No; do not match rule, and continue searching for rules.

It is because of this that two lines are needed for the `worktime` category.

One particularly powerful way to filter requests is with the `url_regex` ACL line. This allows you to specify a regular expression that is checked against each request: If the expression matches the request, the condition matches.

For example, if you want to stop people downloading Windows executable files, you would use this line:

```
acl noexes url_regex -i exe$
```

The dollar sign means "end of URL," which means it would match http://www.somesite.com/virus.exe but not http://www.executable.com/innocent.html. The `-i` part means "not case-sensitive," so the rule will match `.exe`, `.Exe`, `.EXE`, and so on. You can use the caret sign (^) for "start of URL."

For example, you could stop some pornography sites using this ACL:

```
acl noporn url_regex -i sex
```

Do not forget to run the `kill -SIGHUP` command each time you make changes to Squid; otherwise, it will not reread your changes. You can have Squid check your configuration files for errors by running `squid -k parse` as root. If you see no errors, it means your configuration is fine.

NOTE

It is critical that you run the command `kill -SIGHUP` and provide it the PID of your Squid daemon each time you change the configuration; without this, Squid will not reread its configuration files.

Specifying Client IP Addresses

The configuration options so far have been basic, and you can use many more to enhance the proxying system you want.

After you are past deciding which rules work for you locally, it is time to spread them out to other machines. You do so by specifying IP ranges that should be allowed or disallowed access, and you enter these into Squid using more ACL lines.

If you want to, you can specify all the IP addresses on your network, one per line. However, for networks of more than about 20 people or using *Dynamic Host Control Protocol (DHCP)*, that is more work than necessary. A better solution is to use *classless interdomain routing (CIDR)* notation, which allows you to specify addresses like this:

```
192.0.0.0/8
192.168.0.0/16
192.168.0.0/24
```

Each line has an IP address, followed by a slash and then a number. That last number defines the range of addresses you want covered and refers to the number of bits in an IP address. An IP address is a 32-bit number, but we are used to seeing it in dotted-quad notation: A.B.C.D. Each of those quads can be between 0 and 255 (although in practice some of these are reserved for special purposes), and each is stored as an 8-bit number.

The first line in the previous code covers IP addresses starting from 192.0.0.0; the `/8` part means that the first 8 bits (the first quad, 192) is fixed and the rest is flexible. So, Squid treats that as addresses 192.0.0.0, 192.0.0.1, through to 192.0.0.255, then 192.0.1.0, 192.0.1.1, all the way through to 192.255.255.255.

The second line uses `/16`, which means Squid will allow IP addresses from 192.168.0.0 to 192.168.255.255. The last line has `/24`, which allows from 192.168.0.0 to 192.168.0.255.

These addresses are placed into Squid using the `src` ACL line, as follows:

```
acl internal_network src 10.0.0.0/24
```

That line creates a category of addresses from 10.0.0.0 to 10.0.0.255. You can combine multiple address groups together, like this:

```
acl internal_network src 10.0.0.0/24 10.0.3.0/24 10.0.5.0/24 192.168.0.1
```

That example allows 10.0.0.0 through 10.0.0.255, then 10.0.3.0 through 10.0.3.255, and finally the single address 192.168.0.1.

Keep in mind that if you are using the local machine and you have the web browser configured to use the proxy at 127.0.0.1, the client IP address will be 127.0.0.1, too. So, make sure you have rules in place for `localhost`.

As with other ACL lines, you need to enable them with appropriate `http_access allow` and `http_access deny` lines.

Example Configurations

To help you fully understand how Squid access control works, and to give you a head start developing your own rules, the following are some ACL lines you can try. Each line is preceded with one or more comment lines (starting with a `#`) explaining what it does:

```
# include the domains news.bbc.co.uk and slashdot.org
# and not newsimg.bbc.co.uk or www.slashdot.org.
acl newssites dstdomain news.bbc.co.uk slashdot.org

# include any subdomains or bbc.co.uk or slashdot.org
acl newssites dstdomain .bbc.co.uk .slashdot.org

# only include sites located in Canada
acl canadasites dstdomain .ca
```

```
# only include working hours
acl workhours time MTWHF 9:00-18:00

# only include lunchtimes
acl lunchtimes time MTWHF 13:00-14:00

# only include weekends
acl weekends time AS 00:00-23:59

# include URLs ending in ".zip". Note: the \ is important,
# because "." has a special meaning otherwise
acl zipfiles url_regex -i \.zip$

# include URLs starting with https
acl httpsurls url_regex -i ^https

# include all URLs that match "hotmail"
url_regex hotmail url_regex -i hotmail

# include three specific IP addresses
acl directors src 10.0.0.14 10.0.0.28 10.0.0.31

# include all IPs from 192.168.0.0 to 192.168.0.255
acl internal src 192.168.0.0/24

# include all IPs from 192.168.0.0 to 192.168.0.255
# and all IPs from 10.0.0.0 to 10.255.255.255
acl internal src 192.168.0.0/24 10.0.0.0/8
```

When you have your ACL lines in place, you can put together appropriate `http_access` lines. For example, you might want to use a multilayered access system so that certain users (for example, company directors) have full access, whereas others are filtered. For example:

```
http_access allow directors
http_access deny hotmail
http_access deny zipfiles
http_access allow internal lunchtimes
http_access deny all
```

Because Squid matches those in order, directors will have full, unfiltered access to the Web. If the client IP address is not in the directors list, the two `deny` lines are processed so that the user cannot download `.zip` files or read online mail at Hotmail. After blocking those two types of requests, the `allow` on line four allows internal users to access the Web, as long as they do so only at lunchtime. The last line (which is highly recommended) blocks all other users from the proxy.

Reference

http://www.squid-cache.org/—The home page of the Squid Web Proxy Cache.

http://squid-docs.sourceforge.net/latest/book-full.html—The home page of Squid: A User's Guide, a free online book about Squid.

http://www.faqs.org/docs/securing/netproxy-squid.html—A brief online guide to configuring a local Squid server.

http://squid.visolve.com/squid/index.htm/—The home page of a company that can provide commercial support and deployment of Squid.

http://squid.visolve.com/squid/reverseproxy.htm—ViSolve's guide to setting up Squid to reverse proxy to cache a local web server for external visitors.

As well as these URLs, there are two excellent books on the topic of web caching. The first is *Squid: The Definitive Guide* (O'Reilly) by Duane Wessels, ISBN: 0-596-00162-2. The second is *Web Caching* (O'Reilly) also by Duane Wessels, ISBN: 1-56592-536-X.

Of the two, the former is more practical and covers the Squid server in depth. The latter is more theoretical, discussing how caching is implemented. Wessels is one of the leading developers on Squid, so both books are of impeccable technical accuracy.

CHAPTER 21

Administering Database Services

IN THIS CHAPTER

This chapter is an introduction to MySQL and PostgreSQL, two database systems that are included with Ubuntu. You will learn what these systems do, how the two programs compare, and how to consider their advantages and disadvantages. This information can help you choose and deploy which one to use for your organization's database needs.

The database administrator (DBA) for an organization has several responsibilities, which vary according to the size and operations of the organization, supporting staff, and so on. Depending on the particular organization's structure, if you are the organization's DBA, your responsibilities might include the following:

- **Installing and maintaining database servers**—You might install and maintain the database software. Maintenance can involve installing patches as well as upgrading the software at the appropriate times. As DBA, you might need to have root access to your system and know how to manage software (see Chapter 7, "Managing Software"). You also need to be aware of kernel, file system, and other security issues.

- **Installing and maintaining database clients**—The database client is the program used to access the database (you'll learn more on that later in this chapter, in the section "Database Clients"), either locally or remotely over a network. Your responsibilities might include installing and maintaining these client programs on users' systems. This chapter discusses how to install and work with the clients from both the Linux command line and through its graphical interface database tools.

- **Managing accounts and users**—Account and user management include adding and deleting users from the database, assigning and administering passwords, and so on. In this chapter, you will learn how to grant and revoke user privileges and passwords for MySQL and PostgreSQL while using Ubuntu.
- **Ensuring database security**—To ensure database security, you need to be concerned with things such as access control, which ensures that only authorized people can access the database, and permissions, which ensure that people who can access the database cannot do things they should not do. In this chapter, you will learn how to manage Secure Shell (SSH), web, and local GUI client access to the database. Planning and overseeing the regular backup of an organization's database and restoring data from those backups is another critical component of securing the database.
- **Ensuring data integrity**—Of all the information stored on a server's hard disk storage, chances are the information in the database is the most critical. Ensuring data integrity involves planning for multiple-user access and ensuring that changes are not lost or duplicated when more than one user is making changes to the database at the same time.

A Brief Review of Database Basics

Database services under Linux that use the software discussed in this chapter are based on a *client/server* model. Database clients are often used to input data and to query or display query results from the server. You can use the command line or a graphical client to access a running server. Databases generally come in two forms: flat file and relational. A *flat file* database can be as simple as a text file with a space, tab, or some other character delimiting different parts of the information. One example of a simple flat file database is the Ubuntu `/etc/passwd` file. Another example could be a simple address book that might look something like this:

```
Doe~John~505 Some Street~Anytown~NY~12345~555-555-1212
```

You can use standard UNIX tools such as grep, awk, and perl to search for and extract information from this primitive database. Although this might work well for a small database such as an address book that only one person uses, flat file databases of this type have several limitations:

- **They do not scale well**—Flat file databases cannot perform random access on data. They can only perform sequential access. This means they have to scan each line in the file, one by one, to look for specific information. As the size of the database grows, access times increase and performance decreases.

- **Flat file databases are unsuitable for multi-user environments**—Depending on how the database is set up, it either enables only one user to access it at a time or allows two users to make changes simultaneously, and the changes could end up overwriting each other and cause data loss.

These limitations obviously make the flat file database unsuitable for any kind of serious work in even a small business—much less in an enterprise environment. Relational databases, or relational database management systems (RDBMSs) to give them their full name, are good at finding the relationships between individual pieces of data. An RDBMS stores data in tables with fields much like those in spreadsheets, making the data searchable and sortable. RDBMSs are the focus of this chapter.

Oracle, DB2, Microsoft SQL Server, and the freely available PostgreSQL and MySQL are all examples of RDBMSs. The following sections discuss how relational databases work and provide a closer look at some of the basic processes involved in administering and using databases. You will also learn about SQL, the standard language used to store, retrieve, and manipulate database data.

How Relational Databases Work

An RDBMS stores data in tables, which you can visualize as spreadsheets. Each column in the table is a field; for example, a column might contain a name or an address. Each row in the table is an individual record. The table itself has a name you use to refer to that table when you want to get data out of it or put data into it. Figure 21.1 shows an example of a simple relational database that stores name and address information.

last_name	first_name	address	city	state	zip	phone
Doe	John	501 Somestreet	Anytown	NY	55011	555-555-1212
Doe	Jane	501 Somestreet	Anytown	NY	55011	555-555-1212
Palmer	John	205 Anystreet	Sometown	NY	55055	123-456-7890
Johnson	Robert	100 Easystreet	Easytown	CT	12345	111-222-3333

FIGURE 21.1 In this visualization of how an RDBMS stores data, the database stores four records (rows) that include name and address information, divided into seven fields (columns) of data.

In the example shown in Figure 21.1, the database contains only a single table. Most RDBMS setups are much more complex than this, with a single database containing multiple tables. Figure 21.2 shows an example of a database named `sample_database` that contains two tables.

phonebook

last_name	first_name	phone
Doe	John	555-555-1212
Doe	Jane	555-555-1212
Palmer	John	555-123-4567
Johnson	Richard	555-111-4321

cd_collection

id	title	artist	year	rating
1	Mindbomb	The The	1989	4
2	For All You've Done	Hillsong	2004	
3	Trouser Jazz	Mr Scruff	2002	5
4	Natural Elements	Acoustic Alchemy	1988	3
5	Combat Rock	The Clash	1982	4
6	Life for Rent	Dido	2003	5
7	Adiemus 4	Karl Jenkins	2000	4
8	The Two Towers	Howard Shore	2002	5

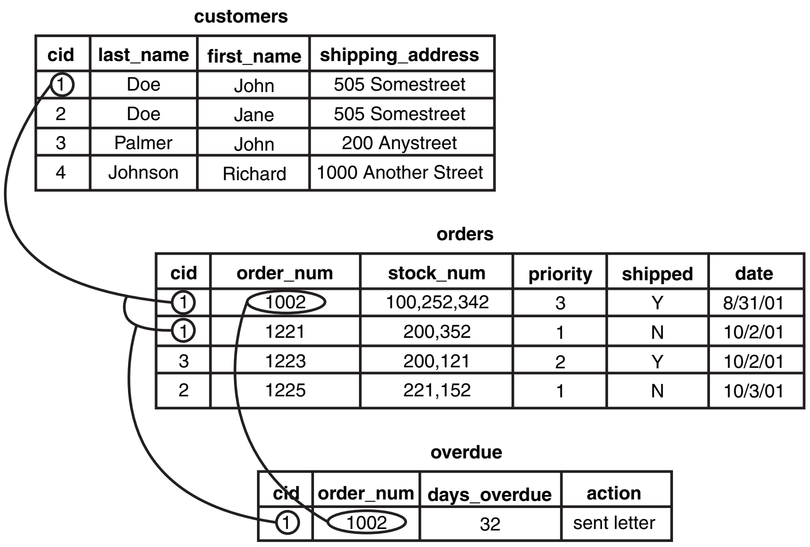
FIGURE 21.2 A single database can contain two tables—in this case, `phonebook` and `cd_collection`.

In the `sample_database` example, the `phonebook` table contains four records (rows) and each record hold three fields (columns) of data. The `cd_collection` table holds eight records, divided into five fields of data.

If you are thinking that there is no logical relationship between the `phonebook` table and the `cd_collection` table in the `sample_database` example, you are correct. In a relational database, users can store multiple tables of data in a single database—even if the data in one table is unrelated to the data in others.

For example, suppose you run a small company that sells widgets and you have a computerized database of customers. In addition to storing each customer's name, address, and phone number, you want to be able to look up outstanding order and invoice information for any of your customers. You could use three related tables in an RDBMS to store and organize customer data for just those purposes. Figure 21.3 shows an example of such a database.

In the example in Figure 21.3, we have added a customer ID field to each customer record. This field holds a customer ID number that is the unique piece of information that can be used to link all other information for each customer to track orders and invoices. Each customer is given an ID unique to him; two customers might have the same data in their name fields, but their ID field values will never be the same. The Customer ID field data in the Orders and Overdue tables replaces the Last Name, First Name, and Shipping Address field information from the `Customers` table. Now, when you want to run a search for any customer's order and invoice data, you can search based on one key rather than multiple keys. You get more accurate results in faster, easier-to-conduct data searches.



customers

cid	last_name	first_name	shipping_address
1	Doe	John	505 Somestreet
2	Doe	Jane	505 Somestreet
3	Palmer	John	200 Anystreet
4	Johnson	Richard	1000 Another Street

orders

cid	order_num	stock_num	priority	shipped	date
1	1002	100,252,342	3	Y	8/31/01
1	1221	200,352	1	N	10/2/01
3	1223	200,121	2	Y	10/2/01
2	1225	221,152	1	N	10/3/01

overdue

cid	order_num	days_overdue	action
1	1002	32	sent letter

FIGURE 21.3 You can use three related tables to track customers, orders, and outstanding invoices.

Now that you have an idea of how data is stored in an RDBMS and how the RDBMS structure enables you to work with that data, you are ready to learn how to input and output data from the database. This is where SQL comes in.

Understanding SQL Basics

SQL (pronounced "S-Q-L") is a database query language understood by virtually all RDBMSs available today. You use SQL statements to get data into and retrieve data from a database. As with statements in any language, SQL statements have a defined structure that determines their meanings and functions.

As a DBA, you should understand the basics of SQL, even if you will not be doing any of the actual programming yourself. Fortunately, SQL is similar to standard English, so learning the basics is simple.

Creating Tables

As mentioned previously, an RDBMS stores data in tables that look similar to spreadsheets. Of course, before you can store any data in a database, you need to create the necessary tables and columns to store the data. You do this by using the CREATE statement.

For example, the cd_collection table from Figure 21.2 has 5 columns, or fields: id, title, artist, year, and rating.

SQL provides several column types for data that define what kind of data will be stored in the column. Some of the available types are INT, FLOAT, CHAR, and VARCHAR. Both CHAR and VARCHAR hold text strings, with the difference being that CHAR holds a fixed-length string, whereas VARCHAR holds a variable-length string.

There are also special column types, such as DATE, that only take data in a date format, and ENUMs (enumerations), which can be used to specify that only certain values are allowed. If, for example, you wanted to record the genre of your CDs, you could use an ENUM column that accepts only the values POP, ROCK, EASY_LISTENING, and so on. You will learn more about ENUM later in this chapter.

Looking at the cd_collection table, you can see that three of the columns hold numerical data and the other two hold string data. In addition, the character strings are of variable length. Based on this information, you can discern that the best type to use for the text columns is type VARCHAR, and the best type to use for the others is INT. You should notice something else about the cd_collection table: One of the CDs is missing a rating, perhaps because we have not listened to it yet. This value, therefore, is optional; it starts empty and can be filled in later.

You are now ready to create a table. As mentioned before, you do this by using the CREATE statement, which uses the following syntax:

```
CREATE TABLE table_name (column_name column_type(parameters) options, ...);
```

You should know the following about the CREATE statement:

- **SQL commands are not case sensitive**—For example, CREATE TABLE, create table, and Create Table are all valid.
- **Whitespace is generally ignored**—This means you should use it to make your SQL commands clearer.

The following example shows how to create the table for the cd_collection database:

```
CREATE TABLE cd_collection
(
id INT NOT NULL,
title VARCHAR(50) NOT NULL,
artist VARCHAR(50) NOT NULL,
year VARCHAR(50) NOT NULL,
rating VARCHAR(50) NULL
);
```

Notice that the statement terminates with a semicolon. This is how SQL knows you are finished with all the entries in the statement. In some cases, the semicolon can be omitted, and we will point out these cases when they arise.

TIP

SQL has a number of reserved keywords that cannot be used in table names or field names. For example, if you keep track of CDs you want to take with you on vacation, you would not be able to use the field name `select` because that is a reserved keyword. Instead, you should either choose a different name (`selected`?) or just prefix the field name with an `f`, such as `fselect`.

Inserting Data into Tables

After you create the tables, you can put data into them. You can insert data manually with the `INSERT` statement, which uses the following syntax:

```
INSERT INTO table_name VALUES('value1', 'value2', 'value3', ...);
```

This statement inserts *`value1`*, *`value2`*, and so on into the table *`table_name`*. The values that are inserted constitute one row, or *record*, in the database. Unless specified otherwise, values are inserted in the order in which the columns are listed in the database table. If, for some reason, you want to insert values in a different order (or if you want to insert only a few values and they are not in sequential order), you can specify which columns you want the data to go in by using the following syntax:

```
INSERT INTO table_name (column1,column4) VALUES('value1', 'value2');
```

You can also fill multiple rows with a single `INSERT` statement, using syntax such as the following:

```
INSERT INTO table_name VALUES('value1', 'value2'),('value3', 'value4');
```

In this statement, *`value1`* and *`value2`* are inserted into the first row and *`value3`* and *`value4`* are inserted into the second row.

The following example shows how you would insert the `Nevermind` entry into the `cd_collection` table:

```
INSERT INTO cd_collection VALUES(9, 'Nevermind', ''Nirvana', '1991,
''NULL);
```

MySQL requires the `NULL` value for the last column (`rating`) if you do not want to include a rating. PostgreSQL, on the other hand, lets you get away with just omitting the last column. Of course, if you had columns in the middle that were null, you would need to explicitly state `NULL` in the `INSERT` statement.

Normally, `INSERT` statements are coded into a front-end program so users adding data to the database do not have to worry about the SQL statements involved.

Retrieving Data from a Database

Of course, the main reason for storing data in a database is so you can later look up, sort, and generate reports on that data. Basic data retrieval is done with the `SELECT` statement, which has the following syntax:

```
SELECT column1, column2, column3 FROM table_name WHERE search_criteria;
```

The first two parts of the statement—the `SELECT` and `FROM` parts—are required. The `WHERE` portion of the statement is optional. If it is omitted, all rows in the table *`table_name`* are returned.

The *`column1`*, *`column2`*, *`column3`* indicates the name of the columns you want to see. If you want to see all columns, you can also use the wildcard `*` to show all the columns that match the search criteria. For example, the following statement displays all columns from the `cd_collection` table:

```
SELECT * FROM cd_collection;
```

If you wanted to see only the titles of all the CDs in the table, you would use a statement such as the following:

```
SELECT title FROM cd_collection;
```

To select the title and year of a CD, you would use the following:

```
SELECT title, year FROM cd_collection;
```

If you wanted something a little fancier, you can use SQL to print the CD title followed by the year in parentheses, as is the convention. Both MySQL and PostgreSQL provide string concatenation functions to handle problems such as this. However, the syntax is different in the two systems.

In MySQL, you can use the `CONCAT()` function to combine the `title` and `year` columns into one output column, along with parentheses. The following statement is an example:

```
SELECT CONCAT(title," (",year, ")") AS TitleYear FROM cd_collection;
```

That statement lists both the title and year under one column that has the label `TitleYear`. Note that there are two strings in the `CONCAT()` function along with the fields—these add whitespace and the parentheses.

In PostgreSQL, the string concatenation function is simply a double pipe (¦¦). The following command is the PostgreSQL equivalent of the preceding MySQL command:

```
SELECT (genus¦¦'' ('¦¦species¦¦')') AS TitleYear FROM cd_collection;
```

Note that the parentheses are optional, but they make the statement easier to read. Once again, the strings in the middle and at the end (note the space between the quotes) are used to insert spacing and parentheses between the title and year.

Of course, more often than not, you do not want a list of every single row in the database. Rather, you only want to find rows that match certain characteristics. For this, you add the `WHERE` statement to the `SELECT` statement. For example, suppose you want to find all the CDs in the `cd_collection` table that have a rating of 5. You would use a statement like the following:

```
SELECT * FROM cd_collection WHERE rating = 5;
```

Using the table from Figure 21.2, you can see that this query would return the rows for *Trouser Jazz*, *Life for Rent*, and *The Two Towers*. This is a simple query, and SQL is capable of handling queries much more complex than this. Complex queries can be written using logical `AND` and logical `OR` statements. For example, suppose you want to refine the query so it lists only those CDs that were not released in 2003. You would use a query like the following:

```
SELECT * FROM cd_collection WHERE rating = 5 AND year != 2003;
```

In SQL, `!=` means "is not equal to." So once again looking at the table from Figure 21.2, you can see that this query returns the rows for *Trouser Jazz* and *The Two Towers* but does not return the row for *Life for Rent* because it was released in 2003.

So, what if you want to list all the CDs that have a rating of 3 or 4 except those released in the year 2000? This time, you combine logical `AND` and logical `OR` statements:

```
SELECT * FROM cd_collection WHERE rating = 3 OR rating = 4 AND year != 2000;
```

This query would return entries for *Mind Bomb*, *Natural Elements*, and *Combat Rock*. However, it wouldn't return entries for *Adiemus 4* because it was released in 2000.

TIP

One of the most common errors among new database programmers is confusing logical `AND` and logical `OR`. For example, in everyday speech, you might say "Find me all CDs released in 2003 and 2004." At first glance, you might think that if you fed this statement to the database in SQL format, it would return the rows for *For All You've Done* and *Life for Rent*. In fact, it would return no rows at all. This is because the database interprets the statement as "Find all rows in which the CD was released in 2003 and was released in 2004." It is, of course, impossible for the same CD to be released twice, so this statement would never return any rows, no matter how many CDs were stored in the table. The correct way to form this statement is with an `OR` statement instead of an `AND` statement.

SQL is capable of far more than is demonstrated here. But as mentioned before, this section is not intended to teach you all there is to know about SQL programming; rather, it teaches you the basics so you can be a more effective DBA.

Choosing a Database: MySQL Versus PostgreSQL

If you are just starting out and learning about using a database with Linux, the first logical step is to research which database will best serve your needs. Many database software packages are available for Linux; some are free, and others cost hundreds of thousands of dollars. Expensive commercial databases, such as Oracle, are beyond the scope of this book. Instead, this chapter focuses on two freely available databases: MySQL and PostgreSQL.

Both of these databases are quite capable, and either one could probably serve your needs. However, each database has a unique set of features and capabilities that might serve your needs better or make developing database applications easier for you.

Speed

Until recently, the speed choice was simple: If the speed of performing queries was paramount to your application, you used MySQL. MySQL has a reputation for being an extremely fast database. Until recently, PostgreSQL was quite slow by comparison.

Newer versions of PostgreSQL have improved in terms of speed (when it comes to disk access, sorting, and so on). In certain situations, such as periods of heavy simultaneous access, PostgreSQL can be significantly faster than MySQL, as you will see in the next section. However, MySQL is still extremely fast when compared to many other databases.

Data Locking

To prevent data corruption, a database needs to put a lock on data while it is being accessed. As long as the lock is on, no other process can access the data until the first process has released the lock. This means that any other processes trying to access the data have to wait until the current process completes. The next process in line then locks the data until it is finished, and the remaining processes have to wait their turn, and so on.

Of course, operations on a database generally complete quickly, so in environments with a small number of users simultaneously accessing the database, the locks are usually of such short duration that they do not cause any significant delays. However, in environments in which many people are accessing the database simultaneously, locking can create performance problems as people wait their turn to access the database.

Older versions of MySQL lock data at the table level, which can be considered a bottleneck for updates during periods of heavy access. This means that when someone writes a row of data in the table, the entire table is locked so no one else can enter data. If your table has 500,000 rows (or records) in it, all 500,000 rows are locked any time 1 row is accessed. Once again, in environments with a relatively small number of simultaneous users, this doesn't cause serious performance problems because most operations complete so quickly that the lock time is extremely short. However, in environments in which many people are accessing the data simultaneously, MySQL's table-level locking can be a significant performance bottleneck.

PostgreSQL, on the other hand, locks data at the row level. In PostgreSQL, only the row currently being accessed is locked. The rest of the table can be accessed by other users. This row-level locking significantly reduces the performance impact of locking in environments that have a large number of simultaneous users. Therefore, as a general rule, PostgreSQL is better suited for high-load environments than MySQL.

The MySQL release bundled with Ubuntu gives you the choice of using tables with table-level or row-level locking. In MySQL terminology, MyISAM tables use table-level locking and InnoDB tables use row-level locking.

NOTE

MySQL's data locking methods are discussed in more depth at http://www.mysql.com/doc/en/Internal_locking.html.

You can find more information on PostgreSQL's locking at http://www.postgresql.org/idocs/index.php?locking-tables.html.

ACID Compliance in Transaction Processing to Protect Data Integrity

Another way MySQL and PostgreSQL differ is in the amount of protection they provide for keeping data from becoming corrupted. The acronym ACID is commonly used to describe several aspects of data protection:

- **Atomicity**—This means that several database operations are treated as an indivisible (atomic) unit, often called a *transaction*. In a transaction, either all unit operations are carried out or none of them are. In other words, if any operation in the atomic unit fails, the entire atomic unit is canceled.
- **Consistency**—Ensures that no transaction can cause the database to be left in an inconsistent state. Inconsistent states can be caused by database client crashes, network failures, and similar situations. Consistency ensures that, in such a situation, any transaction or partially completed transaction that would cause the database to be left in an inconsistent state is *rolled back*, or undone.
- **Isolation**—Ensures that multiple transactions operating on the same data are completely isolated from each other. This prevents data corruption if two users try to write to the same record at the same time. The way isolation is handled can generally be configured by the database programmer. One way that isolation can be handled is through locking, as discussed previously.

- **Durability**—Ensures that, after a transaction has been committed to the database, it cannot be lost in the event of a system crash, network failure, or other problem. This is usually accomplished through transaction logs. Durability means, for example, that if the server crashes, the database can examine the logs when it comes back up and it can commit any transactions that were not yet complete into the database.

PostgreSQL is ACID-compliant, but again MySQL gives you the choice of using ACID-compliant tables or not. MyISAM tables are not ACID-compliant, whereas InnoDB tables are. Note that ACID compliancy is no easy task: All the extra precautions incur a performance overhead.

SQL Subqueries

Subqueries allow you to combine several operations into one atomic unit, and they enable those operations to access each other's data. By using SQL subqueries, you can perform some extremely complex operations on a database. In addition, using SQL subqueries eliminates the potential problem of data changing between two operations as a result of another user performing some operation on the same set of data. Both PostgreSQL and MySQL have support for subqueries in this release of Ubuntu, but this was not true in earlier releases.

Procedural Languages and Triggers

A *procedural language* is an external programming language that can be used to write functions and procedures. This allows you to do things that aren't supported by simple SQL. A *trigger* allows you to define an event that will invoke the external function or procedure you have written. For example, a trigger can be used to cause an exception if an `INSERT` statement containing an unexpected or out-of-range value for a column is given.

For example, in the CD tracking database, you could use a trigger to cause an exception if a user entered data that did not make sense. PostgreSQL has a procedural language called PL/pgSQL. Although MySQL has support for a limited number of built-in procedures and triggers, it does not have any procedural language. This means you cannot create custom procedures or triggers in MySQL, although the same effects can often be achieved through creative client-side programming.

Configuring MySQL

A free and stable version of MySQL is included with Ubuntu. MySQL is also available from the website http://www.mysql.com. The software is available in source code, binary, and APT format for Linux. You can use the `Synaptic` client to add the software to your system. See Chapter 7 for the details on adding (or removing) software.

After you have MySQL installed, you need to initialize the *grant tables* or permissions to access any or all databases and tables and column data within a database. You can do this by issuing `mysql_install_db` as root. This command initializes the grant tables and creates a MySQL root user.

CAUTION

The MySQL data directory needs to be owned by the user as which MySQL will run (changing ownership using the `chown` command). In addition, only this user should have any permissions on this directory. (In other words, the permissions should be set to `700` by using `chmod`.) Setting up the data directory any other way creates a security hole.

Running `mysql_install_db` should generate output similar to the following:

```
sudo mysql_install_db
Preparing db table
Preparing host table
Preparing user table
Preparing func table
Preparing tables_priv table
Preparing columns_priv table
Installing all prepared tables
020916 17:39:05 /usr/libexec/mysqld: Shutdown Complete
...
```

The command prepares MySQL for use on the system and reports helpful information. The next step is to set the password for the MySQL root user, which is discussed in the following section.

CAUTION

By default, the MySQL root user is created with no password. This is one of the first things you must change because the MySQL root user has access to all aspects of the database. The following section explains how to change the password of the user.

Setting a Password for the MySQL Root User

To set a password for the root MySQL user, you need to connect to the MySQL server as the root MySQL user; you can use the command `mysql -u root` to do so. This command connects you to the server with the MySQL client. When you have the MySQL command prompt, issue a command like the following to set a password for the root user:

```
mysql> SET PASSWORD FOR root = PASSWORD("secretword");
```

`secretword` should be replaced by whatever you want to be the password for the root user. You can use this same command with other usernames to set or change passwords for other database users.

After you enter a password, you can exit the MySQL client by typing `exit` at the command prompt.

Creating a Database in MySQL

In MySQL you create a database by using the `CREATE DATABASE` statement. To create a database, you connect to the server by typing **`mysql -u root -p`** and pressing Enter. After you do so, you are connected to the database as the MySQL root user and prompted for a password. After you enter the password, you are placed at the MySQL command prompt. Then you use the `CREATE DATABASE` command. For example, the following commands create a database called `animals`:

```
sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.23.58

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE animals;
Query OK, 1 row affected (0.00 sec)
mysql>
```

Another way to create a database is to use the `mysqladmin` command, as the root user, with the `create` keyword and the name of a new database. For example, to create a new database named `reptiles`, you use a command line like this:

```
# mysqladmin -u root -p create reptiles
```

Granting and Revoking Privileges in MySQL

You probably want to grant yourself some privileges, and eventually you will probably want to grant privileges to other users. Privileges, also known as *rights*, are granted and revoked on four levels:

- **Global-level**—These rights allow access to any database on a server.
- **Database-level**—These rights allow access to all tables in a database.
- **Table-level**—These rights allow access to all columns within a table in a database.
- **Column-level**—These rights allow access to a single column within a database's table.

NOTE

Listing all the available privileges is beyond the scope of this chapter. See the MySQL documentation for more information.

To add a user account, you connect to the database by typing `mysql -u root -p` and pressing Enter. You are then connected as the root user and prompted for a password. (You did set a password for the root user, as instructed in the last section, right?) After you enter the root password, you are placed at the MySQL command prompt.

To grant privileges to a user, you use the `GRANT` statement, which has the following syntax:

```
grant what_to_grant ON where_to_grant TO user_name IDENTIFIED BY 'password';
```

The first option, *`what_to_grant`*, is the privileges you are granting to the user. These privileges are specified with keywords. For example, the `ALL` keyword is used to grant global-, database-, table-, and column-level rights for a specified user.

The second option, *`where_to_grant`*, specifies the resources on which the privileges should be granted. The third option, *`user_name`*, is the username to which you want to grant the privileges. Finally, the fourth option, *`password`*, is a password that should be assigned to this user. If this is an existing user who already has a password and you are modifying permissions, you can omit the `IDENTIFIED BY` portion of the statement.

For example, to grant all privileges on a database named `sampledata` to a user named `foobar`, you could use the following command:

```
GRANT ALL ON animals.* TO foobar IDENTIFIED BY 'secretword';
```

The user `foobar` can now connect to the database `sampledata` by using the password `secretword`, and `foobar` has all privileges on the database, including the ability to create and destroy tables. For example, the user `foobar` can now log in to the server (by using the current hostname—`shuttle2`, in this example), and access the database like so:

```
$ mysql -h shuttle2 -u foobar -p animals
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 43 to server version: 3.23.58

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

NOTE

See the section "The MySQL Command-Line Client" for additional command-line options.

Later, if you need to revoke privileges from `foobar`, you can use the `REVOKE` statement. For example, the following statement revokes all privileges from the user `foobar`:

```
REVOKE ALL ON animals FROM foobar;
```

Advanced database administration, privileges, and security are very complex topics that are beyond the scope of this book. See the section "Reference" at the end of this chapter for links to online documentation. You can also check out Luke Welling's and Laura Thompson's book *PHP and MySQL Development* from Sams Publishing.

Configuring PostgreSQL

If you do not want to use the version of PostgreSQL bundled with Ubuntu, the latest PostgreSQL binary files and source are available at http://www.postgresql.org. The PostgreSQL packages are distributed as several files. At a minimum, you probably want the `postgresql`, `postgresql-server`, and `postgresql-libs` packages. You should see the `README` file in the FTP directory ftp://ftp.postgresql.org/pub/ to determine whether you need any other packages.

If you are installing from the Ubuntu package files, a necessary `postgres` user account (that is, an account with the name of the user running the server on your system) is created for you automatically:

```
$ fgrep postgres /etc/passwd
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
```

Otherwise, you need to create a user called `postgres` during the installation. This user shouldn't have login privileges because only root should be able to use `su` to become this user and no one will ever log in directly as the user. (See Chapter 10, "Managing Users," for more information on how to add users to an Ubuntu system.) After you have added the user, you can install each of the PostgreSQL packages you downloaded using the standard `dpkg -i` command for a default installation.

Initializing the Data Directory in PostgreSQL

After the packages have been installed, you need to initialize the data directory. To do so, you must first create the data directory and you must be the root user. The following example assumes that the data directory is `/usr/local/pgsql/data`.

Create the `/usr/local/pgsql/data` directory (using `mkdir`) and change the ownerships of the directory (using `chown` and `chgrp`) so it is owned by the user `postgres`. Then use `su` and, as the user `postgres`, issue the following commands:

```
# mkdir /usr/local/pgsql
# chown postgres /usr/local/pgsql
# chgrp postgres /usr/local/pgsql
# su - postgres
-bash-2.05b$ initdb -D /usr/local/pgsql/data
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale en_US.UTF-8.
This locale setting will prevent the use of indexes for pattern matching
```

```
operations.  If that is a concern, rerun initdb with the collation order
set to "C".  For more information see the Administrator's Guide.

creating directory /usr/local/pgsql/data... ok
creating directory /usr/local/pgsql/data/base... ok
creating directory /usr/local/pgsql/data/global... ok
creating directory /usr/local/pgsql/data/pg_xlog... ok
creating directory /usr/local/pgsql/data/pg_clog... ok
creating template1 database in /usr/local/pgsql/data/base/1... ok
creating configuration files... ok
initializing pg_shadow... ok
enabling unlimited row size for system tables... ok
initializing pg_depend... ok
creating system views... ok
loading pg_description... ok
creating conversions... ok
setting privileges on built-in objects... ok
vacuuming database template1... ok
copying template1 to template0... ok

Success. You can now start the database server using:

 /usr/bin/postmaster -D /usr/local/pgsql/data
or
 /usr/bin/pg_ctl -D /usr/local/pgsql/data -l logfile start
```

This initializes the database and sets the permissions on the data directory to their correct values.

CAUTION

The `initdb` program sets the permissions on the data directory to `700`. You should not change these permissions to anything else to avoid creating a security hole.

You can start the `postmaster` program with the following command (make sure you are still the user `postgres`):

```
$ postmaster -D /usr/local/pgsql/data &
```

If you have decided to use a directory other than `/usr/local/pgsql/data` as the data directory, you should replace the directory in the `postmaster` command line with whatever directory you are using.

TIP

By default, Ubuntu makes the PostgreSQL data directory `/var/lib/pgsql/data`. This isn't a very good place to store the data, however, because most people do not have the necessary space in the `/var` partition for any kind of serious data storage. Note that if you do change the data directory to something else (such as `/usr/local/pgsql/data`, as in the examples in this section), you need to edit the PostgreSQL startup file (named `postgres`) located in `/etc/init.d` to reflect the change.

Creating a Database in PostgreSQL

Creating a database in PostgreSQL is straightforward, but it must be performed by a user who has permissions to create databases in PostgreSQL—for example, initially the user named `postgres`. You can then simply issue the following command from the shell prompt (not the PSQL client prompt, but a normal shell prompt):

```
# su - postgres
-bash-2.05b$ createdb database
```

where *`database`* is the name of the database you want to create.

The `createdb` program is actually a wrapper that makes it easier to create databases without having to log in and use `psql`. However, you can also create databases from within `psql` with the `CREATE DATABASE` statement. Here's an example:

```
CREATE DATABASE database;
```

You need to create at least one database before you can start the `pgsql` client program. You should create this database while you're logged in as the user `postgres`. To log in as this user, you need to use `su` to become root and then use `su` to become the user `postgres`. To connect to the new database, you start the `psql` client program with the name of the new database as a command-line argument, like so:

```
$ psql sampledata
```

If you don't specify the name of a database when you invoke `psql`, the command attempts to connect to a database that has the same name as the user as which you invoke `psql` (that is, the default database).

Creating Database Users in PostgreSQL

To create a database user, you use `su` to become the user `postgres` from the Linux root account. You can then use the PostgreSQL `createuser` command to quickly create a user who is allowed to access databases or create new database users, like this:

```
$ createuser phudson
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) y
CREATE USER
```

In this example, the new user named phudson is created and allowed to create new databases and database users (you should carefully consider who is allowed to create new databases or additional users).

You can also use the PostgreSQL command-line client to create a new user by typing psql along with name of the database and then use the CREATE USER command to create a new user. Here is an example:

```
CREATE USER foobar ;
```

CAUTION

PostgreSQL allows you to omit the WITH PASSWORD portion of the statement. However, doing so causes the user to be created with no password. This is a security hole, so you should always use the WITH PASSWORD option when creating users.

NOTE

When you are finished working in the psql command-line client, you can type \q to get out of it and return to the shell prompt.

Deleting Database Users in PostgreSQL

To delete a database user, you use the dropuser command, along with the user's name, and the user's access is removed from the default database, like this:

```
$ dropuser msmith
DROP USER
```

You can also log in to your database by using psql and then use the DROP USER commands. Here's an example:

```
$ psql demodb
Welcome to psql, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
 \h for help with SQL commands
 \? for help on internal slash commands
 \g or terminate with semicolon to execute query
 \q to quit
```

```
demodb=# DROP USER msmith ;
DROP USER
demodb=# \q
$
```

Granting and Revoking Privileges in PostgreSQL

As in MySQL, granting and revoking privileges in PostgreSQL is done with the `GRANT` and `REVOKE` statements. The syntax is the same as in MySQL except that PostgreSQL doesn't use the `IDENTIFIED BY` portion of the statement because with PostgreSQL, passwords are assigned when you create the user with the `CREATE USER` statement, as discussed previously. Here is the syntax of the `GRANT` statement:

```
GRANT what_to_grant ON where_to_grant TO user_name;
```

The following command, for example, grants all privileges to the user `foobar` on the database `sampledata`:

```
GRANT ALL ON sampledata TO foobar;
```

To revoke privileges, you use the `REVOKE` statement. Here is an example:

```
REVOKE ALL ON sampledata FROM foobar;
```

This command removes all privileges from the user `foobar` on the database `sampledata`.

Advanced administration and user configuration are complex topics. This section cannot begin to cover all the aspects of PostgreSQL administration or of privileges and users. For more information on administering PostgreSQL, see the PostgreSQL documentation or consult a book on PostgreSQL, such as *PostgreSQL* (Sams Publishing).

Database Clients

Both MySQL and PostgreSQL use a client/server system for accessing databases. In the simplest terms, the database server handles the requests that come into the database and the database client handles getting the requests to the server as well as getting the output from the server to the user.

Users never interact directly with the database server even if it happens to be located on the same machine they are using. All requests to the database server are handled by a database client, which might or might not be running on the same machine as the database server.

Both MySQL and PostgreSQL have command-line clients. A command-line client is a very primitive way of interfacing with a database and generally isn't used by end users. As a DBA, however, you use the command-line client to test new queries interactively without having to write front-end programs for that purpose. In later sections of this chapter, you will learn a bit about the MySQL graphical client and the web-based database administration interfaces available for both MySQL and PostgreSQL.

The following sections examine two common methods of accessing a remote database, a method of local access to a database server, and the concept of web access to a database.

NOTE

You should consider access and permission issues when setting up a database. Should users be able to create and destroy databases? Or should they only be able to use existing databases? Will users be able to add records to the database and modify existing records? Or should users be limited to read-only access to the database? And what about the rest of the world? Will the general public need to have any kind of access to your database through the Internet? As DBA, you must determine the answers to these questions.

SSH Access to a Database

Two types of remote database access scenarios are briefly discussed in this section. In the first scenario, the user directly logs in to the database server through SSH (to take advantage of the security benefits of encrypted sessions) and then starts a program on the server to access the database. In this case, shown in Figure 21.4, the database client is running on the database server itself.

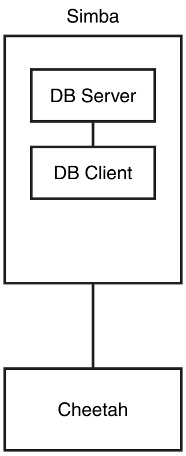


FIGURE 21.4 The user logs in to the database server located on host `simba` from the workstation (host `cheetah`). The database client is running on `simba`.

In the other scenario, shown in Figure 21.5, the user logs in to a remote host through SSH and starts a program on it to access the database, but the database is actually running on a different system. Three systems are now involved—the user's workstation, the remote host running the database client, and the remote host running the database server.

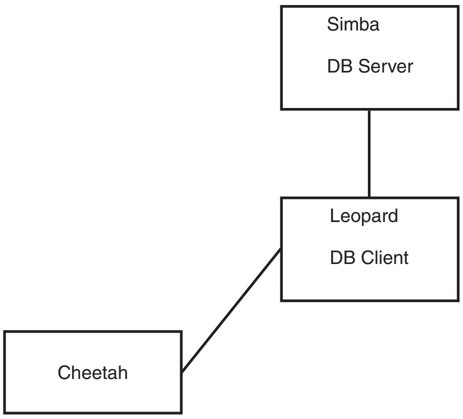


FIGURE 21.5 The user logs in to the remote host `leopard` from the workstation (host `cheetah`) and starts a database client on `leopard`. The client on `leopard` then connects to the database server running on host `simba`. The database client is running on `leopard`.

The important thing to note in Figure 21.5 is the middleman system `leopard`. Although the client is no longer running on the database server itself, it isn't running on the user's local workstation, either.

Local GUI Client Access to a Database

A user can log in to the database server by using a graphical client (which could be running on Windows, Macintosh, or a UNIX workstation). The graphical client then connects to the database server. In this case, the client is running on the user's workstation. Figure 21.6 shows an example.

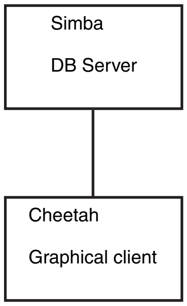


FIGURE 21.6 The user starts a GUI database program on his workstation (hostname `cheetah`). This program, which is the database client, then connects to the database server running on the host `simba`.

Web Access to a Database

In this section, we look at two basic examples of web access to the database server. In the first example, a user accesses the database through a form located on the World Wide Web. At first glance, it might appear that the client is running on the user's workstation. Of course, in reality it is not; the client is actually running on the web server. The web browser on the user's workstation simply provides a way for the user to enter the data that he wants to send to the database and a way for the results sent from the database to be displayed to the user. The software that actually handles sending the request to the database is running on the web server in the form of a CGI script; a Java servlet; or embedded scripting such as the PHP or Sun Microsystems, Inc.'s JavaServer Pages (JSP).

Often, the terms *client* and *front end* are used interchangeably when speaking of database structures. However, Figure 21.7 shows an example of a form of access in which the client and the front end aren't the same thing at all. In this example, the front end is the form displayed in the user's web browser. In such cases, the client is referred to as *middleware*.

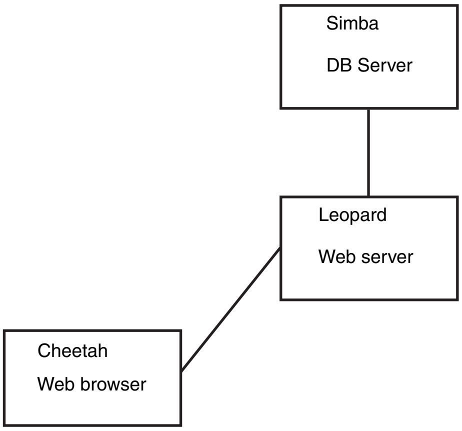


FIGURE 21.7 The user accesses the database through the World Wide Web. The front end is the user's web browser, the client is running on `leopard`, and the server is running on `simba`.

In another possible web access scenario, it could be said that the client is a two-piece application in which part of it is running on the user's workstation and the other part is running on the web server. For example, the database programmer can use JavaScript in the web form to ensure that the user has entered a valid query. In this case, the user's query is partially processed on her own workstation and partially on the web server. Error checking is done on the user's own workstation, which helps reduce the load on the server and also helps reduce network traffic because the query is checked for errors before being sent across the network to the server.

The MySQL Command-Line Client

The MySQL command-line client is `mysql`, and it has the following syntax:

```
mysql [options] [database]
```

Some of the available options for `mysql` are discussed in Table 21.1. *`database`* is optional, and if given, it should be the name of the database to which you want to connect.

TABLE 21.1 Command-Line Options to Use When Invoking `mysql`

Option	Action
`-h` *`hostname`*	Connects to the remote host *`hostname`* (if the database server isn't located on the local system).
`-u` *`username`*	Connects to the database as the user *`username`*.
`-p`	Prompts for a password. This option is required if the user you are connecting as needs a password to access the database. Note that this is a lowercase `p`.
`-P` *`n`*	Specifies *`n`* as the number of the port that the client should connect to. Note that this is an uppercase `P`.
`-?`	Displays a help message.

More options are available than are listed in Table 21.1, but these are the most common options. See the man page for `mysql` for more information on the available options.

CAUTION

Although `mysql` allows you to specify the password on the command line after the `-p` option, and thus allows you to avoid having to type the password at the prompt, you should never invoke the client this way. Doing so causes your password to display in the process list, and the process list can be accessed by any user on the system. This is a major security hole, so you should never give your password on the `mysql` command line.

You can access the MySQL server without specifying a database to use. After you log in, you use the `help` command to get a list of available commands, like this:

```
mysql> help

MySQL commands:
Note that all text commands must be first on line and end with ';'
help (\h) Display this help.
? (\?) Synonym for `help'.
clear (\c) Clear command.
connect (\r) Reconnect to the server. Optional arguments are db and host.
edit (\e) Edit command with $EDITOR.
ego (\G) Send command to mysql server, display result vertically.
```

```
exit (\q) Exit mysql. Same as quit.
go (\g) Send command to mysql server.
nopager (\n) Disable pager, print to stdout.
notee (\t) Don't write into outfile.
pager (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print (\p) Print current command.
quit (\q) Quit mysql.
rehash (\#) Rebuild completion hash.
source (\.) Execute a SQL script file. Takes a file name as an argument.
status (\s) Get status information from the server.
tee (\T) Set outfile [to_outfile]. Append everything into given outfile.
use (\u) Use another database. Takes database name as argument.
```

You can then access a database by using the `use` command and the name of a database that has been created (such as `animals`) and that you are authorized to connect to, like this:

```
mysql> use animals
Database changed
mysql>
```

The PostgreSQL Command-Line Client

You invoke the PostgreSQL command-line client with the command `psql`. Like `mysql`, `psql` can be invoked with the name of the database to which you would like to connect. Also like `mysql`, `psql` can take several options. These options are listed in Table 21.2.

TABLE 21.2 Command-Line Options to Use When Invoking `psql`

Option	Action
`-h` *`hostname`*	Connects to the remote host *`hostname`* (if the database server isn't located on the local system).
`-p` *`n`*	Specifies *n* as the number of the port that the client should connect to. Note that this is a lowercase p.
`-U` *`username`*	Connects to the database as the user *`username`*.
`-W`	Prompts for a password after connecting to the database. In PostgreSQL 7 and later, password prompting is automatic if the server requests a password after a connection has been established.
`-?`	Displays a help message.

Several more options are available in addition to those listed in Table 21.2. See the `psql`'s man page for details on all the available options.

Graphical Clients

If you'd rather interact with a database by using a graphical database client than with the command-line clients discussed in the previous section, you're in luck: A few options are available.

MySQL has an official graphical client, called MySQLGUI. MySQLGUI is available in both source and binary formats from the MySQL website at http://www.mysql.com.

Web-based administration interfaces are also available for MySQL and PostgreSQL. phpMyAdmin and phpPgAdmin are two such products. Both of these products are based on the PHP-embedded scripting language and therefore require you to have PHP installed. Of course, you also need to have a web server installed.

Related Ubuntu and Database Commands

The following commands are useful for creating and manipulating databases in Ubuntu:

`createdb`—Creates a new PostgreSQL database

`createuser`—Creates a new PostgreSQL user account

`dropdb`—Deletes a PostgreSQL database

`dropuser`—Deletes a PostgreSQL user account

`mysql`—Interactively queries the mysqld server

`mysqladmin`—Administers the mysqld server

`mysqldump`—Dumps or backs up MySQL data or tables

`pgaccess`—Accesses a PostgreSQL database server

`pg_ctl`—Controls a PostgreSQL server or queries its status

`psql`—Accesses PostgreSQL via an interactive terminal

Reference

The following are references for the databases mentioned in this chapter:

- http://www.mysql.com—This is the official website of the MySQL database server. Here you can find the latest versions as well as up-to-date information and online documentation for MySQL. You can also purchase support contracts here. You might want to look into this if you will be using MySQL in a corporate setting. (Many corporations balk at the idea of using software for which the company has no support contract in place.)
- http://www.postgresql.org—This is the official website of the PostgreSQL database server. You are asked to select a mirror when you arrive at this site. After you select a mirror, you are taken to the main site. From there, you can find information on the latest versions of PostgreSQL and read the online documentation.

- http://www.postgresql.org/idocs/index.php?tutorial-start.html—This interactive HTML documentation tree is a great place to get started with learning how to use PostgreSQL.
- http://www.pgsql.com—This is a commercial company that provides fee-based support contracts for the PostgreSQL database.

This page intentionally left blank

CHAPTER 22

LDAP

IN THIS CHAPTER

The *Lightweight Directory Access Protocol (LDAP, pronounced ell-dap)* is one of those technologies that, while hidden, forms part of the core infrastructure in enterprise computing. Its job is simple: It stores information about users. However, its power comes from the fact that it can be linked into dozens of other services. LDAP can power login authentication, public key distribution, email routing, and address verification and, more recently, has formed the core of the push toward single-sign-on technology.

TIP

Most people find the concept of LDAP easier to grasp when they think of it as a highly specialized form of database server. Behind the scenes, Ubuntu uses a database for storing all its LDAP information; however, LDAP does not offer anything as straightforward as SQL for data manipulation!

OpenLDAP uses Sleepycat Software's *Berkeley DB (BDB)*, and sticking with that default is highly recommended. That said, there are alternatives if you have specific needs.

This chapter looks at a relatively basic installation of an LDAP server, including how to host a companywide directory service that contains the names and email addresses of employees. LDAP is a client/server system, meaning that an LDAP server hosts the data, and an LDAP client queries it. Ubuntu comes with OpenLDAP as its LDAP server, along with several LDAP-enabled email clients, including Evolution and Mozilla Thunderbird. We cover all three of these applications in this chapter.

Because LDAP data is usually available over the Internet—or at least your local network—it is imperative that you make every effort to secure your server. This chapter gives specific instruction on password configuration for OpenLDAP, and we recommend you follow our instructions closely.

Configuring the Server

If you have been using LDAP for years, you will be aware of its immense power and flexibility. On the other hand, if you are just trying LDAP for the first time, it will seem like the most broken component you could imagine. LDAP has specific configuration requirements, is vastly lacking in graphical tools, and has a large number of acronyms to remember. On the bright side, all the hard work you put in will be worth it because, when it works, LDAP will hugely improve your networking experience.

The first step in configuring your LDAP server is to install the client and server applications. Start up Synaptic and install the `slapd` and `ldap-utils` packages. You'll be asked to enter an administrator password for `slapd`.

Now, use `sudo` to edit `/etc/ldap/slapd.conf` in the text editor of your choice. This is the primary configuration file for `slapd`, the OpenLDAP server daemon. Scroll down until you see the lines `database`, `suffix`, and `rootdn`.

This is the most basic configuration for your LDAP system. What is the name of your server? The `dc` stands for *domain component*, which is the name of your domain as stored in DNS—for example, example.com. For our examples, we used hudzilla.org. LDAP considers each part of a domain name (separated by a period) to be a domain component, so the domain hudzilla.org is made up of a domain component `hudzilla` and a domain component `org`.

Change the suffix line to match your domain components, separated by commas. For example:

```
suffix   "dc=hudzilla,dc=org"
```

The next line defines the root DN, which is another LDAP acronym meaning *distinguished name*. A DN is a complete descriptor of a person in your directory: her name and the domain in which she resides. For example

```
rootdn    "cn=root,dc=hudzilla,dc=org"
```

CN is yet another LDAP acronym, this time meaning common name. A *common name* is just that—the name a person is usually called. Some people have several common names. Andrew Hudson is a common name, but that same user might also have the common name Andy Hudson. In our `rootdn` line, we define a complete user: common name `root` at domain hudzilla.org. These lines are essentially read backward. LDAP goes to `org` first, searches `org` for `hudzilla`, and then searches `hudzilla` for `root`.

The `rootdn` is important because it is more than just another person in your directory. The root LDAP user is like the root user in Linux. It is the person who has complete control over the system and can make whatever changes he wants to.

Now comes a slightly more complex part: We need to give the LDAP root user a password. The easiest way to do this is to open a new terminal window alongside your existing one. Switch to root in the new terminal and type **`slappasswd`**. This tool generates password hashes for OpenLDAP using the SHA1 hash algorithm. Enter a password when it prompts you. When you have entered and confirmed your password, you should see output like this:

```
{SSHA}qMVxFT2K1UUmrA89Gd7z6EK3gRLDIo2W
```

That is the password hash generated from your password. Yours will differ from the one shown here, but what is important is that it has `{SSHA}` at the beginning to denote it uses SHA1. You now need to switch back to the other terminal (the one editing `slapd.conf`) and add this line below the `rootdn` line:

```
rootpw <your password hash>
```

You should replace `<your password hash>` with the full output from `slappasswd`, like this:

```
rootpw {SSHA}qMVxFT2K1UUmrA89Gd7z6EK3gRLDIo2W
```

That sets the LDAP root password to the one you just generated with `slappaswd`. That is the last change you need to make in the `slapd.conf` file, so save your changes and close your editor.

Back in the terminal, run the `slaptest` command. This checks your `slapd.conf` file for errors and ensures you edited it correctly. Presuming there are no errors, run `/etc/init.d/slapd` to start OpenLDAP.

Ubuntu automatically starts OpenLDAP each time you boot up, but that command starts it right now.

The final configuration step is to tell Ubuntu which DN it should use if none is specified. You do so by going to System Settings and selecting Authentication. In the dialog box that appears, check Enable LDAP Support in both the User Information tab and Authentication tab. Next, click the Configure LDAP button, enter your DCs (for example, `dc=hudzilla,dc=org`) for the LDAP Search Base DN, and enter **`127.0.0.1`** for the LDAP Server. Click OK, and then click OK again.

TIP

Checking Enable LDAP Support does not actually change the way in which your users log in. Behind the scenes, this forces Ubuntu to set up the `ldap.conf` file in `/etc/ldap` so that LDAP searches that do not specify a base search start point are directed to your DC.

Populating Your Directory

With LDAP installed, configured, and running, you can now fill the directory with people. This involves yet more LDAP acronyms and is by no means an easy task, so do not worry if you have to reread this several times before it sinks in.

First, create the file base.ldif. You will use this to define the base components of your system: the domain and the address book. LDIF is an acronym standing for *LDAP Data Interchange Format*, and it is the standard way of recording user data for insertion into an LDAP directory. Here are the contents we used for our example:

```
dn: dc=hudzilla,dc=org
objectClass: top
objectClass: dcObject
objectClass: organization
dc: hudzilla
o: Hudzilla Dot Org

dn: ou=People,dc=hudzilla,dc=org
ou: People
objectClass: top
objectClass: organizationalUnit
```

This file contains two individual entities, separated by an empty line. The first is our organization, hudzilla.org. The dn lines you know already, as they define each object uniquely in the scope of the directory. The objectClass directive specifies which attributes should be allowed for this entity and which attributes should be required. In this case, we use it to set the DC to hudzilla and to set o (the name of the organization) to Hudzilla Dot Org.

The next entity defines the address book, People, in which all our people will be stored. It is defined as an organizational unit, which is what the ou stands for. An *organizational unit* really is just an arbitrary partition of your company. You might have OUs for marketing, accounting, and management, for example.

You need to customize the file to your own requirements. Specifically, change the DCs to those you specified in your slapd.conf.

Next, create and edit a new file called people.ldif. This is where you will define entries for your address book, also using LDIF. Here are the people we used in our example:

```
dn: cn=Paul Hudson,ou=People,dc=hudzilla,dc=org
objectClass: inetOrgPerson
cn: Paul Hudson
cn: Hudzilla
mail: paul@hudzilla.org
jpegPhoto:< file:///home/paul/paulhudson.jpg
sn: Hudson
```

```
dn: cn=Andrew Hudson,ou=People,dc=hudzilla,dc=org
objectClass: inetOrgPerson
cn: Andrew Hudson
cn: IzAndy
mail: andrew@hudzilla.org
sn: Hudson

dn: cn=Nick Veitch,ou=People,dc=hudzilla,dc=org
objectClass: inetOrgPerson
cn: Nick Veitch
cn: CrackAttackKing
mail: nick@hudzilla.org
sn: Veitch
```

There are three entries there, again separated by empty lines. Each person has a DN that is made up of his common name (CN), organizational unit (OU), and domain components (DCs). He also has an `objectClass` definition, `inetOrgPerson`, which gives him standard attributes such as an email address, a photograph, and a telephone number. Entities of type `inetOrgPerson` must have a CN and an SN (*surname*) so you will see them in this code.

Note also that each person has two common names: his actual name and a nickname. Not all LDAP clients support more than one CN, but there is no harm in having several as long as the main one comes first and is listed in the DN.

TIP

Having multiple key/value pairs, like multiple CNs, is one of the defining features of LDAP. In today's interconnected world, few people can be defined using a single set of attributes, because people now have home phone numbers, work phone numbers, cell phone numbers, pager numbers, plus several email addresses, and potentially even a selection of offices where they hot desk. Using multiple CNs and other attributes allows you to properly record these complex scenarios.

The `jpegPhoto` attribute for the first entity has very particular syntax. Immediately after the colon, you use an opening angle bracket (<) followed by a space and then the location of the person's picture. Because the picture is local, it is prefixed with `file://`. It is in `/home/paul/paulhudson.jpg`, so the whole URL is file:///home/paul/paulhudson.jpg.

After you have edited the file to include the people in your organization, save it and close the editor. As root, issue these two commands:

```
ldapadd -x -W -D "cn=root,dc=hudzilla,dc=org" -f base.ldif
ldapadd -x -W -D "cn=root,dc=hudzilla,dc=org" -f people.ldif
```

The `ldapadd` command is used to convert LDIF into live directory content and, most important, can be executed while your LDAP server is running. The `-x` parameter means to use only basic authentication, which means you need to supply the root username and password. `-W` means to prompt you for the password. `-D` lets you specify a DN for your username; and immediately after the `-D`, we specify the root DN as set earlier in `slapd.conf`. Finally, `-f` means to use the LDIF from the following file.

When you run them, you are prompted for the root password you set earlier. Upon entering it, you should see confirmation messages as your entries are added, like this:

```
adding new entry "cn=Paul Hudson,ou=People,dc=hudzilla,dc=org"
```

If you see an error such as `ldap_bind: Can't contact LDAP server (-1)`, you need to start the LDAP server by typing `/etc/init.d/slapd start`. The most likely sources of other errors are typing errors. LDIF is a precise format, even down to its use of whitespace.

To test that the directory has been populated and that your configuration settings are correct, run this command:

```
ldapsearch -x 'objectclass=*'
```

The `ldapsearch` command does what you might expect: It queries the LDAP directory from the command line. Again, `-x` means to use simple authentication, although in this situation you do not need to provide any credentials because you are only reading from the directory. The `objectclass=*` search specifies to return any entry of any `objectclass`, so the search will return all the entries in your directory.

You can amend the search to be more specific. For example:

```
ldapsearch -x 'cn=Ni*'
```

This command returns all people with a common name that begins with *Ni*. If you get results for your searches, you are ready to configure your clients.

TIP

OpenLDAP needs specific permissions for its files. The `/var/lib/ldap` directory should be owned by user `ldap` and group `ldap`, with permissions `600`. If you experience problems, try running `chmod 600 /var/lib/ldap`.

Configuring Clients

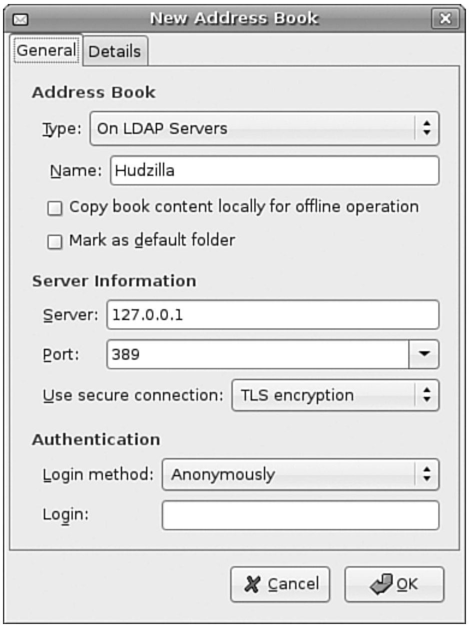
Although Ubuntu comes with a selection of email clients, there is not enough room here to cover them all. So, we discuss the two most frequently used clients: Evolution, the default, and Thunderbird. Both are powerful messaging solutions and so both work well with LDAP. Of the two, Thunderbird seems to be the easier to configure. We have had various problems with Evolution in the past in situations where Thunderbird has worked the first time.

Evolution

To configure Evolution for LDAP, click the arrow next to the New button and select Address Book. A new screen appears, the first option of which prompts you for the type of address book to create. Select On LDAP Servers.

For Name, just enter **`Address book`**, and for Server, enter the IP address of your LDAP server (or **`127.0.0.1`** if you are working on the server), as shown in Figure 22.1. Leave the port as `389`, which is the default for `slapd`. Switch to the Details tab, and set Search Base to be the DN for your address book—for example, `ou=People,dc=hudzilla,dc=org`. Set Search Scope to be Sub so that Evolution will perform a comprehensive search. To finish, click Add Address Book.

Although Evolution is now configured to use your directory, it will not use it for email address autocompletion just yet. To enable that, go to the Tools menu and click Settings. From the options that appear on the left, click Autocompletion and select your LDAP server from the list. Click Close, and then create a new email message. If everything has worked, typing part of someone's name should pop up a box with LDAP matches.


New Address Book

General | Details

Address Book

Type: On LDAP Servers

Name: Hudzilla

☐ Copy book content locally for offline operation

☐ Mark as default folder

Server Information

Server: 127.0.0.1

Port: 389

Use secure connection: TLS encryption

Authentication

Login method: Anonymously

Login:

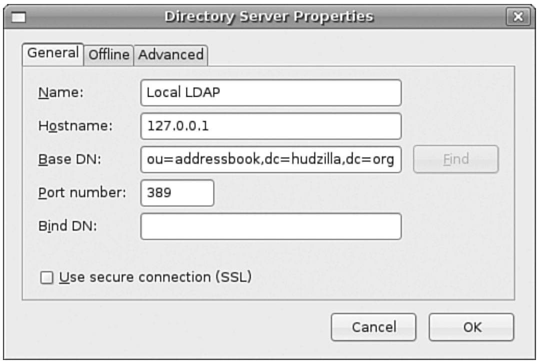
FIGURE 22.1 Configuring Evolution to use LDAP for addresses is easy for anonymous connections.

Thunderbird

Thunderbird is a little easier to configure than Evolution and tends to work better, particularly with entries that have multiple CNs. To enable autocompletion, go to the Edit menu, click Preferences, and then select Composition from the tabs along the top.

From the Addressing subtab, check the Directory Server box and click the Edit Directories button to its right. From the dialog box that appears, click Add to add a new directory. You can give it any name you want because this is merely for display purposes. As shown in Figure 22.2, set the Hostname field to be the IP address of your LDAP server (or `127.0.0.1` if you are working on the server). Set the Base DN to be the DN for your address book (for instance, `ou=People,dc=hudzilla,dc=org`), and leave the port number as `389`. Click OK three times to get back to the main interface.

Now, click Write to create a new email message, and type the first few letters of a user in the To box. If everything has worked, Thunderbird should pop up a box with LDAP matches.



Name: Local LDAP
Hostname: 127.0.0.1
Base DN: ou=addressbook,dc=hudzilla,dc=org
Port number: 389
Bind DN:
☐ Use secure connection (SSL)

FIGURE 22.2 Thunderbird's options are buried deeper than Evolution's, but it does allow you to download the LDAP directory for offline use.

Administration

After you have your LDAP server and clients set up, they require little maintenance until something changes externally. Specifically, if someone in your directory changes jobs, changes her phone number, gets married (changing her last name [surname]), quits, or so forth, you need to be able to update your directory to reflect the change.

OpenLDAP comes with a selection of tools for manipulating directories, of which you have already met `ldapadd`. To add to that, you can use `ldapdelete` for deleting entries in your directory and `ldapmodify` for modifying entries. Both are hard to use but come with moderate amounts of documentation in their man pages.

A much smarter option is to use phpLDAPadmin, which is an LDAP administration tool that allows you to add and modify entries entirely through your web browser. You can learn more and download the product to try at http://www.phpldapadmin.com.

Reference

http://www.openldap.org—The home page of the OpenLDAP project where you can download the latest version of the software and meet other users.

http://www.kingsmountain.com/ldapRoadmap.shtml—A great set of links and resources across the Internet that explain various aspects of LDAP and its parent protocol, X500.

http://ldap.perl.org/—The home of the Perl library for interacting with LDAP provides comprehensive documentation to get you started.

http://www.ldapguru.com/—A gigantic portal for LDAP administrators around the world. From forums dedicated to LDAP to jobs specifically for LDAP admins, this site could very well be all you need.

The definitive book on LDAP is *LDAP System Administration* (O'Reilly), ISBN: 1-56592-491-6. It is an absolute must for the bookshelf of any Linux LDAP administrator. For more general reading, try *LDAP Directories Explained* (Addison-Wesley), ISBN: 0-201-78792-X. It has a much stronger focus on the Microsoft Active Directory LDAP implementation, however.

This page intentionally left blank

PART V

Programming Linux

IN THIS PART

This page intentionally left blank

CHAPTER 23

Using Perl

IN THIS CHAPTER

Perl (the *Practical Extraction and Report Language,* or the *Pathologically Eclectic Rubbish Lister,* depending on who you speak to), a powerful scripting tool, enables you to manage files, create reports, edit text, and perform many other tasks when using Linux. Perl is included with Ubuntu and could be considered an integral part of the distribution because Ubuntu depends on Perl for many types of software services, logging activities, and software tools. If you do a full install from this book's disc, you will find nearly 150 software tools written in Perl installed under the `/usr/bin` and `/usr/sbin` directories.

Perl is not the easiest of programming languages to learn because it is designed for flexibility. This chapter shows how to create and use Perl scripts on your system. You will see what a Perl program looks like, how the language is structured, and where you can find modules of prewritten code to help you write your own Perl scripts. This chapter also includes several examples of Perl used to perform a few common functions on a computer system.

Using Perl with Linux

Although originally designed as a data-extraction and report-generation language, Perl appeals to many Linux system administrators because it can be used to create utilities that fill a gap between the capabilities of shell scripts and compiled C programs. Another advantage of Perl over other UNIX tools is that it can process and extract data from binary files, whereas `sed` and `awk` cannot.

NOTE

In Perl, "there is more than one way to do it." This is the unofficial motto of Perl, and it comes up so often that it is usually abbreviated as TIMTOWTDI.

You can use Perl at your shell's command line to execute one-line Perl programs, but most often the programs (usually ending in `.pl`) are run as a command. These programs generally work on any computer platform because Perl has been ported to just about every operating system. Perl is available by default when you install Ubuntu, and you will find its package files on the DVD included with this book.

Perl programs are used to support a number of Ubuntu services, such as system logging. For example, the `logwatch.pl` program is run every morning at 4:20 a.m. by the `crond` (scheduling) daemon on your system. Other Ubuntu services supported by Perl include the following:

- Amanda for local and network backups
- Fax spooling with the `faxrunqd` program
- Printing supported by Perl document filtering programs
- Hardware sensor monitoring setup using the `sensors-detect` Perl program

Perl Versions

As of this writing, the current production version of Perl is 5.8.8 (which is Perl version 5 point 8, patch level 8).

You can download the code from http://www.perl.com and build it yourself from source. You will occasionally find updated versions in APT format for Ubuntu, which can be installed by updating your system. See Chapter 7, "Managing Software," to see how to quickly get a list of available updates for Ubuntu.

You can determine what version of Perl you installed by typing **`perl -v`** at a shell prompt. If you are installing the latest Ubuntu distribution, you should have the latest version of Perl.

A Simple Perl Program

This section introduces a very simple sample Perl program to get you started using Perl. Although trivial for experienced Perl hackers, a short example is necessary for new users who want to learn more about Perl.

To introduce you to the absolute basics of Perl programming, Listing 23.1 illustrates a simple Perl program that prints a short message.

LISTING 23.1 A Simple Perl Program

```
#!/usr/bin/perl
print "Look at all the camels!\n";
```

Type that in and save it to a file called `trivial.pl`. Then make the file executable using the `chmod` command (see the following sidebar) and run it at the command prompt.

Command-Line Error

If you get the message `bash: trivial.pl: command not found` or `bash: ./trivial.pl: Permission denied`, you have either typed the command line incorrectly or forgotten to make `trivial.pl` executable (with the `chmod` command):

```
$ chmod +x trivial.pl
```

You can force the command to execute in the current directory as follows:

```
$ ./trivial.pl
```

Or you can use Perl to run the program like this:

```
$ perl trivial.pl
```

The sample program in the listing is a two-line Perl program. Typing in the program and running it (using Perl or making the program executable) shows how to create your first Perl program, a process duplicated by Linux users around the world every day!

NOTE

`#!` is often pronounced *she-bang*, which is short for *sharp* (the musicians name for the # character), and *bang*, which is another name for the exclamation point. This notation is also used in shell scripts. See Chapter 11, “Automating Tasks,” for more information about writing shell scripts.

The `#!` line is technically not part of the Perl code at all. The # character indicates that the rest of the screen line is a comment. The comment is a message to the shell, telling it where it should go to find the executable to run this program. The interpreter ignores the comment line.

Exceptions to this practice include when the # character is in a quoted string and when it is being used as the delimiter in a regular expression. Comments are useful to document your scripts, like this:

```
#!/usr/bin/perl
# a simple example to print a greeting
print "hello there\n";
```

A block of code, such as what might appear inside a loop or a branch of a conditional statement, is indicated with curly braces (`{}`). For example, here is an infinite loop:

```
#!/usr/bin/perl
# a block of code to print a greeting forever
while (1) {
        print "hello there\n";
};
```

Perl statements are terminated with a semicolon. A Perl statement can extend over several actual screen lines because Perl is not concerned about whitespace.

The second line of the simple program prints the text enclosed in quotation marks. `\n` is the escape sequence for a newline character.

TIP

Using the `perldoc` and `man` commands is an easy way to get more information about the version of Perl installed on your system. To learn how to use the `perldoc` command, enter the following:

```
$ perldoc perldoc
```

To get introductory information on Perl, you can use either of these commands:

```
$ perldoc perl
$ man perl
```

For an overview or table of contents of Perl's documentation, use the `perldoc` command like this:

```
$ perldoc perltoc
```

The documentation is extensive and well organized. Perl includes a number of standard Linux manual pages as brief guides to its capabilities, but perhaps the best way to learn more about Perl is to read its `perlfunc` document, which lists all the available Perl functions and their usage. You can view this document by using the `perldoc` script and typing `perldoc perlfunc` at the command line. You can also find this document online at http://www.cpan.org/doc/manual/html/pod/perlfunc.html.

Perl Variables and Data Structures

Perl is a *weakly typed* language, meaning that it does not require that you declare a data type, such as a type of value (data) to be stored in a particular variable. C, for example, makes you declare that a particular variable is an integer, a character, a structure, or whatever the case may be. Perl variables are whatever type they need to be, and can change type when you need them to.

Perl Variable Types

Perl has three variable types: *scalars*, *arrays*, and *hashes*. A different character is used to signify each variable type.

Scalar variables are indicated with the `$` character, as in `$penguin`. Scalars can be numbers or strings, and they can change type from one to the other as needed. If you treat a number like a string, it becomes a string. If you treat a string like a number, it is translated into a number if it makes sense to do so; otherwise, it usually evaluates to `0`. For example, the string `"76trombones"` evaluates as the number `76` if used in a numeric calculation, but the string `"polar bear"` evaluates to `0`.

Perl arrays are indicated with the `@` character, as in `@fish`. An *array* is a list of values referenced by index number, starting with the first element numbered `0`, just as in C and `awk`. Each element in the array is a scalar value. Because scalar values are indicated with the `$` character, a single element in an array is also indicated with a `$` character.

For example, `$fish[2]` refers to the third element in the `@fish` array. This tends to throw some people off, but is similar to arrays in C in which the first array element is `0`.

Hashes are indicated with the `%` character, as in `%employee`. A *hash* is a list of name and value pairs. Individual elements in the hash are referenced by name rather than by index (unlike an array). Again, because the values are scalars, the `$` character is used for individual elements.

For example, `$employee{name}` gives you one value from the hash. Two rather useful functions for dealing with hashes are *keys* and *values*. The `keys` function returns an array containing all the keys of the hash, and `values` returns an array of the values of the hash. Using this approach, the Perl program in Listing 23.2 displays all the values in your environment, much like typing the `bash` shell's `env` command.

LISTING 23.2 Displaying the Contents of the `env` Hash

```
#!/usr/bin/perl
foreach $key (keys %ENV)  {
  print "$key = $ENV{$key}\n";
}
```

Special Variables

Perl has a wide variety of special variables which usually look like punctuation— `$_`, `$!`, and `$]`—and are all extremely useful for shorthand code. `$_` is the default variable, `$!` is the error message returned by the operating system, and `$]` is the Perl version number.

`$_` is perhaps the most useful of these, and you will see that variable used often in this chapter. `$_` is the Perl default variable, which is used when no argument is specified. For example, the following two statements are equivalent:

```
chomp;
chomp($_);
```

The following loops are equivalent:

```
for $cow (@cattle) {
        print "$cow says moo.\n";
}
for (@cattle)        {
        print "$_ says moo.\n";
}
```

For a complete listing of the special variables, you should see the `perlvar` document that comes with your Perl distribution (such as in the `perlvar` manual page), or you can go online to http://theoryx5.uwinnipeg.ca/CPAN/perl/pod/perlvar.html.

Operators

Perl supports a number of operators to perform various operations. There are *comparison* operators (used to compare values, as the name implies), *compound* operators (used to combine operations or multiple comparisons), *arithmetic* operators (to perform math), and special string constants.

Comparison Operators

The comparison operators used by Perl are similar to those used by C, `awk`, and the `csh` shells, and are used to specify and compare values (including strings). Most frequently, a comparison operator is used within an `if` statement or loop. Perl has comparison operators for numbers and strings. Table 23.1 shows the numeric comparison operators and their behavior.

TABLE 23.1 Numeric Comparison Operators in Perl

Operator	Meaning
`==`	Is equal to
`<`	Less than
`>`	Greater than
`<=`	Less than or equal to
`>=`	Greater than or equal to
`!=`	Not equal to
`..`	Range of >= first operand to <= second operand
`<=>`	Returns `-1` if less than, `0` if equal, and `1` if greater than

Table 23.2 shows the string comparison operators and their behaviors.

TABLE 23.2 String Comparison Operators in Perl

Operator	Meaning
`eq`	Is equal to
`lt`	Less than
`gt`	Greater than
`le`	Less than or equal to
`ge`	Greater than or equal to
`ne`	Not equal to
`cmp`	Returns `-1` if less than, `0` if equal, and `1` if greater than
`=~`	Matched by regular expression
`!~`	Not matched by regular expression

Compound Operators

Perl uses compound operators, similar to those used by C or `awk`, which can be used to combine other operations (such as comparisons or arithmetic) into more complex forms of logic. Table 23.3 shows the compound pattern operators and their behavior.

TABLE 23.3 Compound Pattern Operators in Perl

Operator	Meaning
`&&`	Logical `AND`
`¦¦`	Logical `OR`
`!`	Logical `NOT`
`()`	Parentheses; used to group compound statements

Arithmetic Operators

Perl supports a wide variety of math operations. Table 23.4 summarizes these operators.

TABLE 23.4 Perl Arithmetic Operators

Operator	Purpose
`x**y`	Raises `x` to the `y` power (same as `x^y`)
`x%y`	Calculates the remainder of `x/y`
`x+y`	Adds `x` to `y`
`x-y`	Subtracts `y` from `x`
`x*y`	Multiplies `x` times `y`
`x/y`	Divides `x` by `y`
`-y`	Negates `y` (switches the sign of `y`); also known as the *unary minus*
`++y`	Increments `y` by 1 and uses value (prefix increment)
`y++`	Uses value of `y` and then increments by 1 (postfix increment)
`--y`	Decrements `y` by 1 and uses value (prefix decrement)
`y--`	Uses value of `y` and then decrements by 1 (postfix decrement)
`x=y`	Assigns value of `y` to `x`. Perl also supports operator-assignment operators (`+=`, `-=`, `*=`, `/=`, `%=`, `**=`, and others)

You can also use comparison operators (such as `==` or `<`) and compound pattern operators (`&&`, `¦¦`, and `!`) in arithmetic statements. They evaluate to the value `0` for false and `1` for true.

Other Operators

Perl supports a number of operators that don't fit any of the prior categories. Table 23.5 summarizes these operators.

TABLE 23.5 Other Perl Operators

Operator	Purpose
`~x`	Bitwise not (changes 0 bits to 1 and 1 bits to 0)
`x & y`	Bitwise and
`x ¦ y`	Bitwise or
`x ^ y`	Bitwise exclusive or (XOR)
`x << y`	Bitwise shift left (shifts x by y bits)
`x >> y`	Bitwise shift right (shifts x by y bits)
`x . y`	Concatenate y onto x
`a x b`	Repeats string a for b number of times
`x , y`	Comma operator—evaluates x and then y
`x ? y : z`	Conditional expression—if x is true, y is evaluated; otherwise, z is evaluated

Except for the comma operator and conditional expression, these operators can also be used with the assignment operator, similar to the way addition (+) can be combined with assignment (=), giving +=.

Special String Constants

Perl supports string constants that have special meaning or cannot be entered from the keyboard.

Table 23.6 shows most of the constants supported by Perl.

TABLE 23.6 Perl Special String Constants

Expression	Meaning
`\\`	The means of including a backslash
`\a`	The alert or bell character
`\b`	Backspace
`\cC`	Control character (like holding the Ctrl key down and pressing the C character)
`\e`	Escape
`\f`	Formfeed
`\n`	Newline
`\r`	Carriage return
`\t`	Tab
`\xNN`	Indicates that NN is a hexadecimal number
`\0NNN`	Indicates that NNN is an octal (base 8) number

Conditional Statements: `if/else` and `unless`

Perl offers two conditional statements, `if` and `unless`, which function opposite one another. `if` enables you to execute a block of code only if certain conditions are met so that you can control the flow of logic through your program. Conversely, `unless` performs the statements when certain conditions are not met.

The following sections explain and demonstrate how to use these conditional statements when writing scripts for Linux.

`if`

The syntax of the Perl `if/else` structure is as follows:

```
if (condition) {
  statement or block of code
} elsif (condition) {
  statement or block of code
} else {
  statement or block of code
}
```

`condition` can be a statement that returns a true or false value.

Truth is defined in Perl in a way that might be unfamiliar to you, so be careful. Everything in Perl is true except `0` (the digit zero), `"0"` (the string containing the number `0`), `""` (the empty string), and an undefined value. Note that even the string `"00"` is a true value because it is not one of the four false cases.

The statement or block of code is executed if the test condition returns a true value.

For example, Listing 23.3 uses the `if/else` structure and shows conditional statements using the `eq` string comparison operator.

LISTING 23.3 `if/elsif/else`

```
if  ($favorite eq "chocolate") {
      print "I like chocolate too.\n";
} elsif ($favorite eq "spinach") {
      print "Oh, I don't like spinach.\n";
} else {
     print "Your favorite food is $favorite.\n";
}
```

unless

`unless` works just like `if`, only backward. `unless` performs a statement or block if a condition is false:

```
unless ($name eq "Rich")       {
  print "Go away, you're not allowed in here!\n";
}
```

NOTE

You can restate the preceding example in more natural language like this:

```
print "Go away!\n" unless $name eq "Rich";
```

Looping

A *loop* is a way to repeat a program action multiple times. A very simple example is a countdown timer that performs a task (waiting for one second) 300 times before telling you that your egg is done boiling.

Looping constructs (also known as *control structures*) can be used to iterate a block of code as long as certain conditions apply, or while the code steps through (evaluates) a list of values, perhaps using that list as arguments.

Perl has four looping constructs: `for`, `foreach`, `while`, and `until`.

for

The `for` construct performs a *statement* (block of code) for a set of conditions defined as follows:

```
for (start condition; end condition; increment function) {
  statement(s)
}
```

The start condition is set at the beginning of the loop. Each time the loop is executed, the increment function is performed until the end condition is achieved. This looks much like the traditional `for/next` loop. The following code is an example of a `for` loop:

```
for ($i=1; $i<=10; $i++) {
      print "$i\n"
}
```

foreach

The `foreach` construct performs a statement block for each element in a list or array:

```
@names = ("alpha","bravo","charlie");
foreach $name (@names) {
  print "$name sounding off!\n";
}
```

The loop variable (`$name` in the example) is not merely set to the value of the array elements; it is aliased to that element. That means if you modify the loop variable, you're actually modifying the array. If no loop array is specified, the Perl default variable `$_` may be used:

```
@names = ("alpha","bravo","charlie");
foreach (@names) {
  print "$_ sounding off!\n";

}
```

This syntax can be very convenient, but it can also lead to unreadable code. Give a thought to the poor person who'll be maintaining your code. (It will probably be you.)

NOTE

`foreach` is frequently abbreviated as `for`.

while

`while` performs a block of statements as long as a particular condition is true:

```
while ($x<10) {
     print "$x\n";
     $x++;
}
```

Remember that the condition can be anything that returns a true or false value. For example, it could be a function call:

```
while ( InvalidPassword($user, $password) )        {
        print "You've entered an invalid password. Please try again.\n";
        $password = GetPassword;
}
```

until

until is the exact opposite of the while statement. It performs a block of statements as long as a particular condition is false—or, rather, until it becomes true:

```
until (ValidPassword($user, $password))  {
  print "You've entered an invalid password. Please try again.\n";
  $password = GetPassword;
}
```

last **and** next

You can force Perl to end a loop early by using a last statement. last is similar to the C break command—the loop is exited. If you decide you need to skip the remaining contents of a loop without ending the loop itself, you can use next, which is similar to the C continue command. Unfortunately, these statements don't work with do ... while.

On the other hand, you can use redo to jump to a loop (marked by a label) or inside the loop where called:

```
$a = 100;
while (1) {
  print "start\n";
  TEST: {
    if (($a = $a / 2) > 2) {
      print "$a\n";
      if (--$a < 2) {
        exit;
      }
      redo TEST;
    }
  }
}
```

In this simple example, the variable $a is repeatedly manipulated and tested in an endless loop. The word *start* will only be printed once.

do ... while **and** do ... until

The while and until loops evaluate the conditional first. The behavior is changed by applying a do block before the conditional. With the do block, the condition is evaluated last, which results in the contents of the block always executing at least once (even if the condition is false). This is similar to the C language do ... while (*conditional*) statement.

Regular Expressions

Perl's greatest strength is in text and file manipulation, which is accomplished by using the regular expression (regex) library. Regexes, which are quite different from the wild-card-handling and filename-expansion capabilities of the shell (see Chapter 15), allow complicated pattern matching and replacement to be done efficiently and easily.

For example, the following line of code replaces every occurrence of the string `bob` or the string `mary` with `fred` in a line of text:

```
$string =~ s/bob¦mary/fred/gi;
```

Without going into too many of the details, Table 23.7 explains what the preceding line says.

TABLE 23.7 Explanation of `$string =~ s/bob¦mary/fred/gi;`

Element	Explanation
`$string =~`	Performs this pattern match on the text found in the variable called `$string`.
`s`	Substitute.
`/`	Begins the text to be matched.
`bob¦mary`	Matches the text `bob` or `mary`. You should remember that it is looking for the text `mary`, not the word `mary`; that is, it will also match the text `mary` in the word `maryland`.
`/`	Ends text to be matched; begins text to replace it.
`fred`	Replaces anything that was matched with the text `fred`.
`/`	Ends replace text.
`g`	Does this substitution globally; that is, replaces the match text wherever in the string you match it (and any number of times).
`i`	The search text is not case sensitive. It matches `bob`, `Bob`, or `bOB`.
`;`	Indicates the end of the line of code.

If you are interested in the details, you can get more information using the regex (7) section of the manual.

Although replacing one string with another might seem a rather trivial task, the code required to do the same thing in another language (for example, C) is rather daunting unless supported by additional subroutines from external libraries.

Access to the Shell

Perl can perform for you any process you might ordinarily perform by typing commands to the shell through the `` ` ` `` syntax. For example, the code in Listing 23.4 prints a directory listing.

LISTING 23.4 Using Backticks to Access the Shell

```
$curr_dir = `pwd`;
@listing = `ls -al`;
print "Listing for $curr_dir\n";
foreach $file (@listing) {
     print "$file";
}
```

NOTE

The `` notation uses the backtick found above the Tab key (on most keyboards), not the single quotation mark.

You can also use the `Shell` module to access the shell. `Shell` is one of the standard modules that comes with Perl; it allows creation and use of a shell-like command line. Look at the following code for an example:

```
use Shell qw(cp);
cp ("/home/httpd/logs/access.log", "/tmp/httpd.log");
```

This code almost looks like it is importing the command-line functions directly into Perl. Although that is not really happening, you can pretend that the code is similar to a command line and use this approach in your Perl programs.

A third method of accessing the shell is via the `system` function call:

```
$rc = 0xffff & system('cp /home/httpd/logs/access.log /tmp/httpd.log');
if ($rc == 0) {
  print "system cp succeeded \n";
} else {
  print "system cp failed $rc\n";
}
```

The call can also be used with the `or die` clause:

```
system('cp /home/httpd/logs/access.log /tmp/httpd.log') == 0
  or die "system cp failed: $?"
```

However, you can't capture the output of a command executed through the `system` function.

Modules and CPAN

A great strength of the Perl community (and the Linux community) is the fact that it is an open-source community. This community support is expressed for Perl via the

Comprehensive Perl Archive Network (CPAN), which is a network of mirrors of a repository of Perl code.

Most of CPAN is made up of *modules*, which are reusable chunks of code that do useful things, similar to software libraries containing functions for C programmers. These modules help speed development when building Perl programs and free Perl hackers from repeatedly reinventing the wheel when building a bicycle.

Perl comes with a set of standard modules installed. Those modules should contain much of the functionality that you will initially need with Perl. If you need to use a module not installed with Ubuntu, use the CPAN module (which is one of the standard modules) to download and install other modules onto your system. At http://www.perl.com/CPAN, you will find the CPAN Multiplex Dispatcher, which will attempt to direct you to the CPAN site closest to you.

Typing the following command will put you into an interactive shell that gives you access to CPAN. You can type `help` at the prompt to get more information on how to use the CPAN program:

```
$ perl -MCPAN -e shell
```

After installing a module from CPAN (or writing one of your own), you can load that module into memory where you can use it with the `use` function:

```
use Time::CTime;
```

`use` looks in the directories listed in the variable `@INC` for the module. In this example, `use` looks for a directory called `Time`, which contains a file called `CTime.pm`, which in turn is assumed to contain a package called `Time::CTime`. The distribution of each module should contain documentation on using that module.

For a list of all the standard Perl modules (those that come with Perl when you install it), see `perlmodlib` in the Perl documentation. You can read this document by typing **`perldoc perlmodlib`** at the command prompt.

Code Examples

The following sections contain a few examples of things you might want to do with Perl.

Sending Mail

You can get Perl to send email in several ways. One method that you see frequently is opening a pipe to the `sendmail` command and sending data to it (shown in Listing 23.5). Another method is using the `Mail::Sendmail` module (available through CPAN), which uses socket connections directly to send mail (as shown in Listing 23.6). The latter method is faster because it does not have to launch an external process. Note that `sendmail` must be running on your system for the Perl program in Listing 23.5 to work.

LISTING 23.5 Sending Mail Using Sendmail

```
#!/usr/bin/perl
open (MAIL, "¦ /usr/sbin/sendmail -t"); # Use -t to protect from users
print MAIL <<EndMail;
To: dpitts\@mk.net
From: rbowen\@mk.net
Subject: Email notification

David,
 Sending email from Perl is easy!
Rich
.
EndMail
close MAIL;
```

NOTE

Note that the @ sign in the email addresses must be escaped so that Perl does not try to evaluate an array of that name. That is, dpitts@mk.net will cause a problem, so you need to use dpitts\@mk.net.

The syntax used to print the mail message is called a *here document*. The syntax is as follows:

```
print <<EndText;
.....
EndText
```

The `EndText` value must be identical at the beginning and at the end of the block, including any whitespace.

LISTING 23.6 Sending Mail Using the `Mail::Sendmail` Module

```
#!/usr/bin/perl
use Mail::Sendmail;
%mail = ('To' => 'dpitts@mk.net',
  'From' => 'rbowen@mk.net'
  'Subject' => 'Email notification',
  'Message' => 'Sending email from Perl is easy!',
);
sendmail(%mail);
```

Perl ignores the comma after the last element in the hash. It is convenient to leave it there; if you want to add items to the hash, you don't need to add the comma. This is purely a style decision.

Using Perl to Install a CPAN Module

You can use Perl to interactively download and install a Perl module from the CPAN archives by using the `-M` and `-e` commands. Start the process by using a Perl like this:

```
# perl -MCPAN -e shell
```

After you press Enter, you will see some introductory information and be asked to choose an initial automatic or manual configuration, which is required before any download or install takes place. Type `no` and press Enter to have Perl automatically configure for the download and install process; or, if desired, simply press Enter to manually configure for downloading and installation. If you use manual configuration, you must answer a series of questions regarding paths, caching, terminal settings, program locations, and so on. Settings are saved in a directory named `.cpan` in current directory.

When finished, you will see the CPAN prompt:

```
cpan>
```

To have Perl examine your system and then download and install a large number of modules, use the `install` keyword, specify `Bundle` at the prompt, and then press Enter like this:

```
cpan> install Bundle::CPAN
```

To download a desired module (using the example in Listing 23.6), use the `get` keyword like so:

```
cpan> get Mail::Sendmail
```

The source for the module will be downloaded into the `.cpan` directory. You can then build and install the module using the `install` keyword, like this:

```
cpan> install Mail::Sendmail
```

The entire process of retrieving, building, and installing a module can also be accomplished at the command line by using Perl's `-e` option like this:

```
# perl -MCPAN  -e "install Mail::Sendmail"
```

Note also that the `@` sign did not need to be escaped within single quotation marks (`''`). Perl does not *interpolate* (evaluate variables) within single quotation marks, but does within double quotation marks and `here` strings (similar to << shell operations).

Purging Logs

Many programs maintain some variety of logs. Often, much of the information in the logs is redundant or just useless. The program shown in Listing 23.7 removes all lines from a file that contain a particular word or phrase, so lines that you know are not important can be purged. For example, you might want to remove all the lines in the Apache error log that originate with your test client machine because you know those error messages were produced during testing.

LISTING 23.7 Purging Log Files

```
#!/usr/bin/perl
# Be careful using this program!
# This will remove all lines that contain a given word
# Usage:  remove <word> <file>
$word=@ARGV[0];
$file=@ARGV[1];
if ($file)  {
  # Open file for reading
  open (FILE, "$file") or die "Could not open file: $!";      @lines=<FILE>;
  close FILE;
  # Open file for writing
  open (FILE, ">$file") or die "Could not open file for writing: $!";
  for (@lines)  {
    print FILE unless /$word/;
  } # End for
  close FILE;
} else {
  print "Usage:  remove <word> <file>\n";
}  #  End if...else
```

The code uses a few idiomatic Perl expressions to keep it brief. It reads the file into an array using the `<FILE>` notation; it then writes the lines back out to the file unless they match the pattern given on the command line.

The `die` function kills program operation and displays an error message if the `open` statements fail. `$!` in the error message, as mentioned in the section on special variables, is the error message returned by the operating system. It will likely be something like `'file not found'` or `'permission denied'`.

Posting to Usenet

If some portion of your job requires periodic postings to Usenet—a FAQ listing, for example—the following Perl program can automate the process for you. In the sample code, the posted text is read in from a text file, but your input can come from anywhere.

The program shown in Listing 23.8 uses the `Net::NNTP` module, which is a standard part of the Perl distribution. You can find more documentation on the `Net::NNTP` module by typing **`'perldoc Net::NNTP'`** at the command line.

LISTING 23.8 Posting an Article to Usenet

```
#!/usr/bin/perl
# load the post data into @post
open (POST, "post.file");
@post = <POST>;
close POST;
```

LISTING 23.8 Continued

```
# import the NNTP module
use Net::NNTP;
$NNTPhost = 'news';
# attempt to connect to the remote host;
# print an error message on failure
$nntp = Net::NNTP->new($NNTPhost)
  or die "Cannot contact $NNTPhost: $!";
# $nntp->debug(1);
$nntp->post()
  or die "Could not post article: $!";
# send the header of the post
$nntp->datasend("Newsgroups: news.announce\n");
$nntp->datasend("Subject: FAQ - Frequently Asked Questions\n");
$nntp->datasend("From: ADMIN <root\@rcbowen.com>\n");
$nntp->datasend("\n\n");
# for each line in the @post array, send it
for (@post)     {
  $nntp->datasend($_);
} #  End for
$nntp->quit;
```

One-Liners

One medium in which Perl excels is the one-liner. Folks go to great lengths to reduce tasks to one line of Perl code. Perl has the rather undeserved reputation of being unreadable. The fact is that you can write unreadable code in any language. Perl allows for more than one way to do something, and this leads rather naturally to people trying to find the most arcane way to do things.

Named for Randal Schwartz, a *Schwartzian Transform* is a way of sorting an array by something that is not obvious. The sort function sorts arrays alphabetically; that is pretty obvious. What if you want to sort an array of strings alphabetically by the third word? Perhaps you want something more useful, such as sorting a list of files by file size? A Schwartzian Transform creates a new list that contains the information that you want to sort by, referencing the first list. You then sort the new list and use it to figure out the order that the first list should be in. Here's a simple example that sorts a list of strings by length:

```
@sorted_by_length =
  map { $_ => [0] }            # Extract original list
  sort { $a=>[1] <=> $b=>[1] } # Sort by the transformed value
  map { [$_, length($_)] }     # Map to a list of element lengths
  @list;
```

Because each operator acts on the thing immediately to the right of it, it helps to read this from right to left (or bottom to top, the way it is written here).

The first thing that acts on the list is the `map` operator. It transforms the list into a hash in which the keys are the list elements and the values are the lengths of each element. This is where you put in your code that does the transformation by which you want to sort.

The next operator is the `sort` function, which sorts the list by the values.

Finally, the hash is transformed back into an array by extracting its keys. The array is now in the desired order.

Command-Line Processing

Perl is great at parsing the output of various programs. This is a task for which many people use tools such as `awk` and `sed`. Perl gives you a larger vocabulary for performing these tasks. The following example is very simple, but illustrates how you might use Perl to chop up some output and do something with it. In the example, Perl is used to list only those files that are larger than 10KB:

```
$ ls -la ¦ perl -nae 'print "$F[8] is $F[4]\n" if $F[4] > 10000;'
```

The `-n` switch indicates that I want the Perl code run for each line of the output. The `-a` switch automatically splits the output into the `@F` array. The `-e` switch indicates that the Perl code is going to follow on the command line.

Related Ubuntu and Linux Commands

You will use these commands and tools when using Perl with Linux:

`a2p`—A filter used to translate `awk` scripts into Perl

`find2perl`—A utility used to create Perl code from command lines using the `find` command

`pcregrep`—A utility used to search data using Perl-compatible regular expressions

`perlcc`—A compiler for Perl programs

`perldoc`—A Perl utility used to read Perl documentation

`s2p`—A filter used to translate `sed` scripts into Perl

`vi`—The `vi` (actually `vim`) text editor

Reference

The first place to look is in the Perl documentation and Linux man pages.

Perl, all of its documentation, and millions of lines of Perl programs are all available free on the Internet. A number of Usenet newsgroups are also devoted to Perl, as are shelves of books and a quarterly journal.

Books

Although your local bookstore might have dozens of titles on Perl, the following are some of the more highly recommended of these. You might also look at the *Camel Critiques* (Tom Christiansen; http://language.perl.com/critiques/index.html) for reviews of other available Perl books.

- *Advanced Perl Programming*, by Sriram Srinivasan, O'Reilly & Associates
- *Sams Teach Yourself Perl in 21 Days*, *Second Edition*, by Laura Lemay, Sams Publishing
- *Sams Teach Yourself Perl in 24 Hours, Second Edition,* by Clinton Pierce, Sams Publishing
- *Learning Perl*, *Third Edition*, by Randal L. Schwartz, Tom Phoenix, O'Reilly & Associates
- *Programming Perl*, *Third Edition*, by Larry Wall, Tom Christiansen, and Jon Orwant, O'Reilly & Associates
- *Effective Perl Programming: Writing Better Programs with Perl*, by Joseph Hall, Addison-Wesley Publishing Company
- *CGI Programming with Perl, Second Edition*, by Gunther Birznieks, Scott Guelich, Shishir Gundavaram, O'Reilly & Associates
- *Mastering Regular Expressions*, by Jeffrey Friedl, O'Reilly & Associates

Usenet

Check out the following on Usenet:

- comp.lang.perl.misc—Discusses various aspects of the Perl programming language. Make sure that your questions are Perl specific, not generic CGI programming questions. The regulars tend to flame folks who don't know the difference.
- comp.infosystems.www.authoring.cgi—Discusses authoring of CGI programs, so much of the discussion is Perl specific. Make sure that your questions are related to CGI programming, not just Perl. The regulars are very particular about staying on topic.

WWW

Check these sites on the World Wide Web:

- http://www.perl.com—Tom Christiansen maintains the Perl language home page. This is the place to find all sorts of information about Perl, from its history and culture to helpful tips. This is also the place to download the Perl interpreter for your system.

- http://www.perl.com/CPAN—This is part of the site just mentioned, but it merits its own mention. CPAN (Comprehensive Perl Archive Network) is the place for you to find modules and programs in Perl. If you write something in Perl that you think is particularly useful, you can make it available to the Perl community here.
- http://www.perl.com/pub/q/FAQs—Frequently Asked Questions index of common Perl queries; this site offers a handy way to quickly search for answers about Perl.
- http://learn.perl.org—One of the best places to start learning Perl online. If you master Perl, go to http://jobs.perl.org.
- http://www.hwg.org—The HTML Writers Guild is a not-for-profit organization dedicated to assisting web developers. One of its services is a plethora of mailing lists. The `hwg-servers` mailing list and the `hwg-languages` mailing list are great places for asking Perl-related questions.
- http://www.pm.org—The Perl Mongers are local Perl users groups. There might be one in your area. The Perl advocacy site is http://www.perl.org.
- http://www.tpj.com—*The Perl Journal* is "a reader-supported monthly e-zine" devoted to the Perl programming language. *TPJ* is always full of excellent, amusing, and informative articles, and is an invaluable resource to both new and experienced Perl programmers.
- http://www-106.ibm.com/developerworks/linux/library/l-p101—A short tutorial about one-line Perl scripts and code.

Other

Other valuable resources not falling into any of the preceding categories are Perl-related Internet Relay Chat channels. You can usually find several channels available on an IRC server. However, the questions, discussions, and focus of similarly named channels will vary from server to server.

For example, a `#perlhelp` channel is usually for general questions about installing, using, and learning about Perl. A `#perl` channel is generally for more advanced questions, but on some servers beginner questions might be welcomed. You might also find some helpful answers on `#cgi` and `#html` channels. Browse to http://perl-begin.berlios.de/irc for a list of servers.

CHAPTER 24

Working with Python

As PHP has come to dominate the world of web scripting, Python is increasingly dominating the domain of command-line scripting. Python's precise and clean syntax makes it one of the easiest languages to learn, and it allows programmers to code more quickly and spend less time maintaining their code. Although PHP is fundamentally similar to Java and Perl, Python is closer to C and Modula-3, and so it might look unfamiliar at first.

Most of the other languages have a group of developers at their cores, but Python has Guido van Rossum—creator, father, and *Benevolent Dictator For Life (BDFL)*. Although Guido spends less time working on Python now, he still essentially has the right of veto changes to the language, which has enabled it to remain consistent over the many years of its development. The end result is that, in Guido's own words, "Even if you are in fact clueless about language design, you can tell that Python is a very simple language."

The following pages constitute a "quick-start" tutorial to Python designed to give you all the information you need to put together basic scripts and to point you toward resources that can take you further.

Python on Linux

Ubuntu comes with Python installed by default, as do many other versions of Linux and UNIX—even Mac OS X comes with Python preinstalled. Part of this is due to convenience because it is such a popular scripting language and it saves having to install it later if the user wants to run a script.

The Python binary is installed into `/usr/bin/python`; if you run that, you enter the Python interactive interpreter where you can type commands and have them executed immediately. Although PHP also has an interactive mode (use `php -a` to activate it), it is neither as powerful nor as flexible as Python's.

As with Perl, PHP, and other scripting languages, you can also execute Python scripts by adding a shebang line (#!) to the start of your scripts that point to `/usr/bin/python` and then setting the file to be executable.

The third and final way to run Python scripts is through `mod_python`, which you can install through `synaptic`.

For the purposes of this introduction, we use the interactive Python interpreter because it provides immediate feedback on commands as you type them.

Getting Interactive

We will be using the interactive interpreter for this chapter, so it is essential that you are comfortable using it. To get started, open a terminal and run the command `python`. You should see this:

```
[paul@caitlin ~]$ python
Python 2.4.3 (#2, Apr 27 2006, 14:43:58)
[GCC 4.0.3 (Ubuntu 4.0.3-1ubuntu5)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The `>>>` is where you type your input, and you can set and get a variable like this:

```
>>> python = 'great'
>>> python
'great'
>>>
```

On line one, the variable `python` is set to the text `great`, and on line two that value is read back from the variable simply by typing the name of the variable you want to read. Line three shows Python printing the variable; on line four you are back at the prompt to type more commands. Python remembers all the variables you use while in the interactive interpreter, which means you can set a variable to be the value of another variable.

When you are done, press Ctrl+D to exit. At this point, all your variables and commands are forgotten by the interpreter, which is why complex Python programs are always saved in scripts!

The Basics of Python

Python is a language wholly unlike most others, and yet it is so logical that most people can pick it up quickly. You have already seen how easily you can assign strings, but in Python nearly everything is that easy—as long as you remember the syntax!

Numbers

The way Python handles numbers is more precise than some other languages. It has all the normal operators—such as + for addition, - for subtraction, / for division, and * for multiplication—but it adds % for modulus (division remainder), ** for raise to the power, and // for floor division. It is also specific about which type of number is being used, as this example shows:

```
>>> a = 5
>>> b = 10
>>> a * b
50
>>> a / b
0
>>> b = 10.0
>>> a / b
0.5
>>> a // b
0.0
```

The first division returns 0 because both a and b are integers (whole numbers), so Python calculates the division as an integer, giving 0. By converting b to 10.0, Python considers it to be a floating-point number, and so the division is now calculated as a floating-point value, giving 0.5. Even with b being floating point, using //—floor division—rounds it down.

Using **, you can easily see how Python works with integers:

```
>>> 2 ** 30
1073741824
>>> 2 ** 31
2147483648L
```

The first statement raises 2 to the power of 30 (that is, 2 × 2 × 2 × 2 × 2 × …), and the second raises 2 to the power of 31. Notice how the second number has a capital *L* on the end of it—this is Python telling you that it is a long integer. The difference between long integers and normal integers is slight but important: Normal integers can be calculated using simple instructions on the CPU, whereas long integers—because they can be as big as you need them to be—need to be calculated in software and therefore are slower.

When specifying big numbers, you need not put the *L* at the end—Python figures it out for you. Furthermore, if a number starts off as a normal number and then exceeds its boundaries, Python automatically converts it to a long integer. The same is not true the other way around: If you have a long integer and then divide it by another number so that it could be stored as a normal integer, it remains a long integer:

```
>>> num = 999999999999999999999999999999999L
>>> num = num / 1000000000000000000000000000000
>>> num
999L
```

You can convert between number types using typecasting, like this:

```
>>> num = 10
>>> int(num)
10
>>> float(num)
10.0
>>> long(num)
10L
>>> floatnum = 10.0
>>> int(floatnum)
10
>>> float(floatnum)
10.0
>>> long(floatnum)
10L
```

You need not worry whether you are using integers or long integers; Python handles it all for you, so you can concentrate on getting the code right.

More on Strings

Python stores strings as an immutable sequence of characters—a jargon-filled way of saying that "it is a collection of characters that, once set, cannot be changed without creating a new string." Sequences are important in Python. There are three primary types, of which strings are one, and they share some properties. Mutability makes much sense when you learn about lists in the next section.

As you saw in the previous example, you can assign a value to strings in Python with just an equal sign, like this:

```
>>> mystring = 'hello';
>>> myotherstring = "goodbye";
>>> mystring
'hello'
>>> myotherstring;
'goodbye'
>>> test = "Are you really Bill O'Reilly?"
>>> test
"Are you really Bill O'Reilly?"
```

The first example encapsulates the string in single quotation marks, and the second and third in double quotation marks. However, printing the first and second strings show them both in single quotation marks because Python does not distinguish between the two. The third example is the exception—it uses double quotation marks because the string itself contains a single quotation mark. Here, Python prints the string with double quotation marks because it knows it contains the single quotation mark.

Because the characters in a string are stored in sequence, you can index into them by specifying the character you are interested in. Like most other languages, these indexes are zero based, which means you need to ask for character `0` to get the first letter in a string. For example:

```
>>> string = "This is a test string"
>>> string
'This is a test string'
>>> string[0]
'T'
>>> string [0], string[3], string [20]
('T', 's', 'g')
```

The last line shows how, with commas, you can ask for several indexes at the same time. You could print the entire first word using this:

```
>>> string[0], string[1], string[2], string[3]
('T', 'h', 'i', 's')
```

However, for that purpose you can use a different concept: slicing. A *slice* of a sequence draws a selection of indexes. For example, you can pull out the first word like this:

```
>>> string[0:4]
'This'
```

The syntax there means "take everything from position 0 (including 0) and end at position 4 (excluding it)." So, `[0:4]` copies the items at indexes 0, 1, 2, and 3. You can omit either side of the indexes, and it will copy either from the start or to the end:

```
>>> string [:4]
'This'
>>> string [5:]
'is a test string'
>>> string [11:]
'est string'
```

You can also omit both numbers, and it will give you the entire sequence:

```
>>> string [:]
'This is a test string'
```

Later you will learn precisely why you would want to do that, but for now there are a number of other string intrinsics that will make your life easier. For example, you can use the + and * operators to concatenate (join) and repeat strings, like this:

```
>>> mystring = "Python"
>>> mystring * 4
'PythonPythonPythonPython'
>>> mystring = mystring + " rocks! "
>>> mystring * 2
'Python rocks! Python rocks! '
```

In addition to working with operators, Python strings come with a selection of built-in methods. You can change the case of the letters with `capitalize()` (uppercases the first letter and lowercases the rest), `lower()` (lowercases them all), `title()` (uppercases the first letter in each word), and `upper()` (uppercases them all). You can also check whether strings match certain cases with `islower()`, `istitle()`, and `isupper()`; that also extends to `isalnum()` (returns `true` if the string is letters and numbers only) and `isdigit()` (returns `true` if the string is all numbers).

This example demonstrates some of these in action:

```
>>> string
'This is a test string'
>>> string.upper()
'THIS IS A TEST STRING'
>>> string.lower()
'this is a test string'
>>> string.isalnum()
False
>>> string = string.title()
>>> string
'This Is A Test String'
```

Why did `isalnum()` return `false`—our string contains only alphanumeric characters, doesn't it? Well, no. There are spaces in there, which is what is causing the problem. More important, we were calling `upper()` and `lower()` and they were not changing the contents of the string—they just returned the new value. So, to change our string from `This is a test string` to `This Is A Test String`, we actually have to assign it back to the string variable.

Lists

Python's built-in list data type is a sequence, like strings. However, they are mutable, which means they can be changed. Lists are like arrays in that they hold a selection of elements in a given order. You can cycle through them, index into them, and slice them:

```
>>> mylist = ["python", "perl", "php"]
>>> mylist
['python', 'perl', 'php']
>>> mylist + ["java"]
['python', 'perl', 'php', 'java']
>>> mylist * 2
['python', 'perl', 'php', 'python', 'perl', 'php']
>>> mylist[1]
'perl'
>>> mylist[1] = "c++"
>>> mylist[1]
'c++'
>>> mylist[1:3]
['c++', 'php']
```

The brackets notation is important: You cannot use parentheses, (`(` and `)`) or braces (`{` and `}`) for lists. Using + for lists is different from using + for numbers. Python detects you are working with a list and appends one list to another. This is known as *operator overloading*, and it is one of the reasons Python is so flexible.

Lists can be *nested*, which means you can put a list inside a list. However, this is where mutability starts to matter, and so this might sound complicated! If you recall, the definition of an immutable string sequence is that it is a collection of characters that, once set, cannot be changed without creating a new string. Lists are mutable as opposed to immutable, which means you can change your list without creating a new list.

This becomes important because Python, by default, copies only a reference to a variable rather than the full variable. For example:

```
>>> list1 = [1, 2, 3]
>>> list2 = [4, list1, 6]
>>> list1
[1, 2, 3]
>>> list2
[4, [1, 2, 3], 6]
```

Here we have a nested list. `list2` contains 4, then `list1`, then 6. When you print the value of `list2`, you can see it also contains `list1`. Now, proceeding on from that:

```
>>> list1[1] = "Flake"
>>> list2
[4, [1, 'Flake', 3], 6]
```

In line one, we set the second element in `list1` (remember, sequences are zero based!) to be `Flake` rather than 2; then we print the contents of `list2`. As you can see, when `list1` changed, `list2` was updated, too. The reason for this is that `list2` stores a reference to `list1` as opposed to a copy of `list1`; they share the same value.

We can show that this works both ways by indexing twice into `list2`, like this:

```
>>> list2[1][1] = "Caramello"
>>> list1
[1, 'Caramello', 3]
```

The first line says, "Get the second element in `list2` (`list1`) and the second element of that list, and set it to be `'Caramello'`." Then `list1`'s value is printed, and you can see it has changed. This is the essence of mutability: We are changing our list without creating a new list. On the other hand, editing a string creates a new string, leaving the old one unaltered. For example:

```
>>> mystring = "hello"
>>> list3 = [1, mystring, 3]
>>> list3
[1, 'hello', 3]
>>> mystring = "world"
>>> list3
[1, 'hello', 3]
```

Of course, this raises the question of how you copy without references when references are the default. The answer, for lists, is that you use the `[:]` slice, which we looked at earlier. This slices from the first element to the last, inclusive, essentially copying it without references. Here is how that looks:

```
>>> list4 = ["a", "b", "c"]
>>> list5 = list4[:]
>>> list4 = list4 + ["d"]
>>> list5
['a', 'b', 'c']
>>> list4
['a', 'b', 'c', 'd']
```

Lists have their own collections of built-in methods, such as `sort()`, `append()`, and `pop()`. The latter two add and remove single elements from the end of the list, with `pop()` also returning the removed element. For example:

```
>>> list5 = ["nick", "paul", "julian", "graham"]
>>> list5.sort()
>>> list5
['graham', 'julian', 'nick', 'paul']
>>> list5.pop()
'paul'
>>> list5
['graham', 'julian', 'nick']
>>> list5.append("Rebecca")
```

In addition, one interesting method of strings returns a list: `split()`. This takes a character to split by and then gives you a list in which each element is a chunk from the string. For example:

```
>>> string = "This is a test string";
>>> string.split(" ")
['This', 'is', 'a', 'test', 'string']
```

Lists are used extensively in Python, although this is slowly changing as the language matures.

Dictionaries

Unlike lists, *dictionaries* are collections with no fixed order. Instead, they have a *key* (the name of the element) and a *value* (the content of the element), and Python places them wherever it needs to for maximum performance. When defining dictionaries, you need to use braces (`{ }`) and colons (`:`). You start with an opening brace and then give each element a key and a value, separated by a colon, like this:

```
>>> mydict = { "perl" : "a language", "php" : "another language" }
>>> mydict
{'php': 'another language', 'perl': 'a language'}
```

This example has two elements, with keys `perl` and `php`. However, when the dictionary is printed, we find that `php` comes before `perl`—Python hasn't respected the order in which we entered them. We can index into a dictionary using the normal code:

```
>>> mydict["perl"]
'a language'
```

However, because a dictionary has no fixed sequence, we cannot take a slice, or index by position.

Like lists, dictionaries are mutable and can also be nested; however, unlike lists, you cannot merge two dictionaries by using `+`. This is because dictionary elements are located using the key. Therefore, having two elements with the same key would cause a clash. Instead, you should use the `update()` method, which merges two arrays by overwriting clashing keys.

You can also use the `keys()` method to return a list of all the keys in a dictionary.

Conditionals and Looping

So far, we have just been looking at data types, which should show you how powerful Python's data types are. However, you simply cannot write complex programs without conditional statements and loops.

Python has most of the standard conditional checks, such as, (greater than), <= (less than or equal to), and == (equal), but it also adds some new ones, such as `in`. For example, we can use `in` to check whether a string or a list contains a given character/element:

```
>>> mystring = "J Random Hacker"
>>> "r" in mystring
True
>>> "Hacker" in mystring
True
>>> "hacker" in mystring
False
```

The last example demonstrates how `in` is case sensitive. We can use the operator for lists, too:

```
>>> mylist = ["soldier", "sailor", "tinker", "spy"]
>>> "tailor" in mylist
False
```

Other comparisons on these complex data types are done item by item:

```
>>> list1 = ["alpha", "beta", "gamma"]
>>> list2 = ["alpha", "beta", "delta"]
>>> list1 > list2
True
```

`list1`'s first element (`alpha`) is compared against `list2`'s first element (`alpha`) and, because they are equal, the next element is checked. That is equal also, so the third element is checked, which is different. The *g* in `gamma` comes after the *d* in `delta` in the alphabet, so `gamma` is considered greater than `delta` and `list1` is considered greater than `list2`.

Loops come in two types, and both are equally flexible. For example, the `for` loop can iterate through letters in a string or elements in a list:

```
>>> string = "Hello, Python!"
>>> for s in string: print s,
...
H e l l o ,   P y t h o n !
```

The `for` loop takes each letter in `string` and assigns it to `s`. This then is printed to the screen using the `print` command, but note the comma at the end; this tells Python not to insert a line break after each letter. The "…" is there because Python allows you to enter more code in the loop; you need to press Enter again here to have the loop execute.

The exact same construct can be used for lists:

```
>>> mylist = ["andi", "rasmus", "zeev"]
>>> for p in mylist: print p
...
andi
rasmus
zeev
```

Without the comma after the `print` statement, each item is printed on its own line.

The other loop type is the `while` loop, and it looks similar:

```
>> while 1: print "This will loop forever!"
...
This will loop forever!
This will loop forever!
This will loop forever!
This will loop forever!
This will loop forever!
(etc)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
KeyboardInterrupt
>>>
```

That is an infinite loop (it will carry on printing that text forever), so you need to press Ctrl+C to interrupt it and regain control.

If you want to use multiline loops, you need to get ready to use your Tab key: Python handles loop blocks by recording the level of indent used. Some people find this odious; others admire it for forcing clean coding on users. Most of us, though, just get on with programming!

For example:

```
>>> i = 0
>>> while i < 3:
...     j = 0
...     while j < 3:
...             print "Pos: " + str(i) + "," + str(j) + ")"
...             j += 1
...     i += 1
...
Pos: (0,0)
Pos: (0,1)
Pos: (0,2)
Pos: (1,0)
```

```
Pos: (1,1)
Pos: (1,2)
Pos: (2,0)
Pos: (2,1)
Pos: (2,2)
```

You can control loops using the `break` and `continue` keywords. `break` exits the loop and continues processing immediately afterward, and `continue` jumps to the next loop iteration.

Functions

Other languages—such as PHP—read and process an entire file before executing it, which means you can call a function before it is defined because the compiler reads the definition of the function before it tries to call it. Python is different: If the function definition has not been reached by the time you try to call it, you get an error. The reason behind this behavior is that Python actually creates an object for your function, and that in turns means two things. First, you can define the same function several times in a script and have the script pick the right one at runtime. Second, you can assign the function to another name just by using =.

Function definition starts with `def`, then the function name, then parentheses and a list of parameters, and then a colon. The contents of a function need to be indented at least one level beyond the definition. So, using function assignment and dynamic declaration, you can write a script that prints the right greeting in a roundabout manner:

```
>>> def hello_english(Name):
...     print "Hello, " + Name + "!"
...
>>> def hello_hungarian(Name):
...     print "Szia, " + Name + "!"
...
>>> hello = hello_hungarian
>>> hello("Paul")
Szia, Paul!
>>> hello = hello_english
>>> hello("Paul")
```

Notice that function definitions include no type information. Functions are typeless, like we said. The upside of this is that you can write one function to do several things:

```
>>> def concat(First, Second):
...     return First + Second
...
>>> concat(["python"], ["perl"])
['python', 'perl']
```

```
>>> concat("Hello, ", "world!")
'Hello, world!'
```

That demonstrates how the `return` statement sends a value back to the caller, but also how a function can do one thing with lists and another thing with strings. The magic here is being accomplished by the objects. We write a function that tells two objects to add themselves together, and the objects intrinsically know how to do that. If they don't—if, perhaps, the user passes in a string and an integer—Python catches the error for us. However, it is this hands-off, "let the objects sort themselves out" attitude that makes functions so flexible. Our `concat()` function could conceivably concatenate strings, lists, or *zonks*—a data type someone created herself that allows adding. The point is that we do not limit what our function can do—tacky as it might sound, the only limit is your imagination!

Object Orientation

After having read this far, you should not be surprised to hear that Python's object orientation is flexible and likely to surprise you if you have been using C-like languages for several years.

The best way to learn Python *object-oriented programming (OOP)* is to just do it. So, here is a basic script that defines a class, creates an object of that class, and calls a function:

```
class dog(object):
    def bark(self):
        print "Woof!"

fluffy = dog()
fluffy.bark()
```

Defining a class starts, predictably, with the `class` keyword followed by the name of the class you are defining and a colon. The contents of that class need to be indented one level so that Python knows where it stops. Note that the `object` inside parentheses is there for object inheritance, which is discussed later. For now, the least you need to know is that if your new class is not based on an existing class, you should put `object` inside parentheses as shown in the previous code.

Functions inside classes work in much the same way as normal functions do (although they are usually called *methods*), with the main difference being that they should all take at least one parameter, usually called `self`. This parameter is filled with the name of the object the function was called on, and you need to use it explicitly.

Creating an instance of a class is done by assignment. You do not need any `new` keyword, as in some other languages—you just provide empty parentheses. Calling a function of that object is done using a period, the name of the class to call, with any parameters being passed inside parentheses.

Class and Object Variables

Each object has its own set of functions and variables, and you can manipulate those variables independently of objects of the same type. In addition, some class variables are set to a default value for all classes and can be manipulated globally.

This script demonstrates two objects of the `dog` class being created, each with its own name:

```
class dog(object):
    name = "Lassie"
    def bark(self):
        print self.name + " says 'Woof!'"
    def setName(self, name):
        self.name = name
fluffy = dog()
fluffy.bark()
poppy = dog()
poppy.setName("Poppy")
poppy.bark()
```

That outputs the following:

```
Lassie says 'Woof!'
Poppy says 'Woof!'
```

There, each dog starts with the name `Lassie`, but it gets customized. Keep in mind that Python assigns by reference by default, meaning each object has a reference to the class's `name` variable, and as we assign that with the `setName()` method, that reference is lost. What this means is that any references we do not change can be manipulated globally. Thus, if we change a class's variable, it also changes in all instances of that class that have not set their own value for that variable. For example:

```
class dog(object):
    name = "Lassie"
    color = "brown"
fluffy = dog()
poppy = dog()
print fluffy.color
dog.color = "black"
print poppy.color
fluffy.color = "yellow"
print fluffy.color
print poppy.color
```

So, the default color of dogs is brown—both the `fluffy` and `poppy` `dog` objects start off as brown. Then, using `dog.color`, we set the default color to be black, and because neither of the two objects has set its own color value, they are updated to be black. The third to last line uses poppy.color to set a custom color value for the poppy object—poppy becomes yellow, while fluffy and the dog class in general remain black.

Constructors and Destructors

To help you automate the creation and deletion of objects, you can easily override two default methods: __init__ and __del__. These are the methods called by Python when a class is being instantiated and freed, known as the *constructor* and *destructor*, respectively.

Having a custom constructor is great for when you need to accept a set of parameters for each object being created. For example, you might want each dog to have its own name on creation, and you could implement that with this code:

```
class dog(object):
    def __init__(self, name):
        self.name = name
fluffy = dog("Fluffy")
print fluffy.name
```

If you do not provide a name parameter when creating the dog object, Python reports an error and stops. You can, of course, ask for as many constructor parameters as you want, although it is usually better to ask only for the ones you need and have other functions to fill in the rest.

On the other side of things is the destructor method, which allows you to have more control over what happens when an object is destroyed. Using the two, we can show the life cycle of an object by printing messages when it is created and deleted:

```
class dog(object):
    def __init__(self, name):
        self.name = name
        print self.name + " is alive!"
    def __del__(self):
        print self.name + " is no more!"
fluffy = dog("Fluffy")
```

The destructor is there to give you the chance to free up resources allocated to the object or perhaps log something to a file.

Class Inheritance

Python allows you to reuse your code by inheriting one class from one or more others. For example, lions, tigers, and bears are all mammals and so share a number of similar properties. In that scenario, you would not want to have to copy and paste functions between them; it would be smarter (and easier!) to have a mammal class that defines all the shared functionality and then inherit each animal from that.

Consider the following code:

```
class car(object):
    color = "black"
    speed = 0
    def accelerateTo(self, speed):
        self.speed = speed
    def setColor(self, color):
        self.color = color
mycar = car()
print mycar.color
```

That creates a `car` class with a default color and also provides a `setColor()` function so people can change their own colors. Now, what do you drive to work? Is it a car? Sure it is, but chances are it is a Ford, or a Dodge, or a Jeep, or some other make—you don't get cars without a make. On the other hand, you do not want to have to define a class `Ford`, give it the methods `accelerateTo()`, `setColor()`, and however many other methods a basic car has and then do exactly the same thing for `Ferrari`, `Nissan`, and so on.

The solution is to use inheritance: Define a `car` class that contains all the shared functions and variables of all the makes and then inherit from that. In Python, you do this by putting the name of the class from which to inherit inside parentheses in the class declaration, like this:

```
class car(object):
    color = "black"
    speed = 0
    def accelerateTo(self, speed):
        self.speed = speed
    def setColor(self, color):
        self.color = color

class ford(car): pass
class nissan(car): pass
mycar = ford()
print mycar.color
```

The `pass` directive is an empty statement—it means our class contains nothing new. However, because the `ford` and `nissan` classes inherit from the `car` class, they get `color`, `speed`, `accelerateTo()`, and `setColor()` provided by their parent class. Note that we do not need `object` after the class names for `ford` and `nissan` because they are inherited from an existing class: `car`.

By default, Python gives you all the methods the parent class has, but you can override that by providing new implementations for the classes that need them. For example:

```
class modelt(car):
    def setColor(self, color):
        print "Sorry, Model Ts only come in black!"

myford = ford()
ford.setColor("green")
mycar = modelt()
mycar.setColor("green")
```

The first car is created as a Ford, so `setColor()` will work fine because it uses the method from the `car` class. However, the second car is created as a Model T, which has its own `setColor()` method, so the call will fail.

This provides an interesting scenario: What do you do if you have overridden a method and yet want to call the parent's method also? If, for example, changing the color of a Model T was allowed but just cost extra, we would want to print a message saying, “You owe $50 more,” but then change the color. To do this, we need to use the class object from which our current class is inherited—`car` in this example. Here's an example:

```
class modelt(car):
    def setColor(self, color):
        print "You owe $50 more"
        car.setColor(self, color)

mycar = modelt()
mycar.setColor("green")
print mycar.color
```

That prints the message and then changes the color of the car.

The Standard Library and the Vaults of Parnassus

A default Python install includes many modules (blocks of code) that enable you to interact with the operating system, open and manipulate files, parse command-line options, perform data hashing and encryption, and much more. This is one of the reasons most commonly cited when people are asked why they like Python so much—it comes stocked to the gills with functionality you can take advantage of immediately. In fact, the number of modules included in the Standard Library is so high that entire books have been written about them—try *Python Standard Library* (O'Reilly, ISBN: 0-596-00096-0) for a comprehensive, if slightly dated, list of them.

For unofficial scripts and add-ons for Python, the recommended starting place is called the Vaults of Parnassus: http://py.vaults.ca. There you can find about 20,000 public scripts and code examples for everything from mathematics to games.

Reference

The Python website (http://www.python.org) is packed with information and updated regularly. This should be your first stop, if only to read the latest Python news.

- http://www.zope.org—The home page of the Zope Content Management System (CMS), it's one of the most popular CMSes around and, more important, written entirely in Python.
- http://www.jython.org—Python also has an excellent Java-based interpreter to allow it to run on other platforms. If you prefer Microsoft .NET, try http://www.iron-python.com.
- http://www.pythonline.com—Guido van Rossum borrowed the name for his language from *Monty Python's Flying Circus*, and as a result many Python code examples use oblique *Monty Python* references. A visit to the official *Monty Python* site to hone your Python knowledge is highly recommended!

There are few truly great books about Python; however, you can find a list of what's on offer at http://www.python.org/moin/PythonBooks. If you are desperate to pick up a book immediately, you could do much worse than choose *Learning Python* (O'Reilly, ISBN: 0-596-00281-5).

CHAPTER 25
Writing PHP Scripts

IN THIS CHAPTER

This chapter introduces you to the world of PHP programming, from the point of view of using it as a web scripting language and as a command-line tool. PHP originally stood for *personal home page* because it was a collection of Perl scripts designed to ease the creation of guest books, message boards, and other interactive scripts commonly found on home pages. However, since those early days, it has received two major updates (PHP 3 and PHP 4), plus a substantial revision in PHP 5, which is the version bundled with Ubuntu.

Part of the success of PHP has been its powerful integration with databases—its earliest uses nearly always took advantage of a database back end. In PHP 5, however, two big new data storage mechanisms were introduced: SQLite, which is a powerful and local database system, and SimpleXML, which is an *application programming interface (API)* designed to make XML parsing and querying easy. As you will see over time, the PHP developers did a great job: Both SQLite and SimpleXML are easy to learn and use.

NOTE

Many packages for PHP are available through `synaptic`. The basic package is just called `php5`, but you might also want to add extensions such as `php5-ldap`, `php5-mysql`, or `php5-pgsql`. Choose only the extensions you plan to use; otherwise, it will waste system resources.

Introduction to PHP

In terms of the way it looks, PHP is a cross between Java and Perl, having taken the best aspects of both and merged them successfully into one language. The Java parts include a powerful object-orientation system, the capability to throw program exceptions, and the general style of writing that both languages borrowed from C. Borrowed from Perl is the "it should just work" mentality where ease of use is favored over strictness. As a result, you will find a lot of "there is more than one way to do it" in PHP.

Entering and Exiting PHP Mode

Unlike PHP's predecessors, you embed your PHP code inside your HTML as opposed to the other way around. Before PHP, many websites had standard HTML pages for most of their content, linking to Perl CGI pages to do back-end processing when needed. With PHP, all your pages are capable of processing and containing HTML.

Each `.php` file is processed by PHP that looks for code to execute. PHP considers all the text it finds to be HTML until it finds one of four things:

- `<?php`
- `<?`
- `<%`
- `<script language="php">`

The first option is the preferred method of entering PHP mode because it is guaranteed to work.

When in PHP mode, you can exit it by using `?>` (for `<?php` and `<?`); `%>` for `<%`) or `</script>` (for `<script language="php">`). This code example demonstrates entering and exiting PHP mode:

```
In HTML mode
<?php
  echo "In PHP mode";
?>
In HTML mode
In <?php echo "PHP"; ?> mode
```

Variables

All variables in PHP start with a dollar sign (`$`). Unlike many other languages, PHP does not have different types of variable for integers, floating-point numbers, arrays, or Booleans. They all start with a `$`, and all are interchangeable. As a result, PHP is a weakly typed language, which means you do not declare a variable as containing a specific type of data; you just use it however you want to.

Save the code in Listing 25.1 into the script `ubuntu1.php`.

LISTING 25.1 Testing Types in PHP

```
<?php
  $i = 10;
  $j = "10";
  $k = "Hello, world";
  echo $i + $j;
  echo $i + $k;
?>
```

To run that script, bring up a console and browse to where you saved it. Then type this command:

```
$ php ubuntu1.php
```

If PHP is installed correctly, you should see the output `2010`, which is really two things. The 20 is the result of 10 + 10 (`$i` plus `$j`), and the 10 is the result of adding 10 to the text string `Hello, world`. Neither of those operations are really straightforward. Whereas `$i` is set to the number 10, `$j` is actually set to be the text value `"10"`, which is not the same thing. Adding 10 to 10 gives 20, as you would imagine, but adding 10 to `"10"` (the string) forces PHP to convert `$j` to an integer on-the-fly before adding it.

Running `$i` + `$k` adds another string to a number, but this time the string is `Hello, world` and not just a number inside a string. PHP still tries to convert it, though, and converting any non-numeric string into a number converts it to 0. So, the second `echo` statement ends up saying `$i` + 0.

As you should have guessed by now, calling `echo` outputs values to the screen. Right now, that prints directly to your console, but internally PHP has a complex output mechanism that enables you to print to a console, send text through Apache to a web browser, send data over a network, and more.

Now that you have seen how PHP handles variables of different types, it is important that you understand the selection of types available to you.

Type	**Stores**
`integer`	Whole numbers; for example, 1, 9, or 324809873
`float`	Fractional numbers; for example, 1.1, 9.09, or 3.141592654
`string`	Characters; for example, “a”, “sfdgh”, or “Ubuntu Unleashed”
`boolean`	True or false
`array`	Several variables of any type
`object`	An instance of a class
`resource`	Any external data

The first four can be thought of as simple variables, and the last three as complex variables. Arrays are simply collections of variables. You might have an array of numbers (the ages of all the children in a class); an array of strings (the names of all Wimbledon tennis champions); or even an array of arrays, known as a *multidimensional array*. Arrays are covered in more depth in the next section because they are unique in the way in which they are defined.

Objects are used to define and manipulate a set of variables that belong to a unique entity. Each object has its own personal set of variables, as well as functions that operate on those variables. Objects are commonly used to model real-world things. You might define an object that represents a TV, with variables such as `$CurrentChannel` (probably an integer), `$SupportsHiDef` (a Boolean), and so on.

Of all the complex variables, the easiest to grasp are resources. PHP has many extensions available to it that allow you to connect to databases, manipulate graphics, or even make calls to Java programs. Because they are all external systems, they need to have types of data unique to them that PHP cannot represent using any of the six other data types. So, PHP stores their custom data types in resources—data types that are meaningless to PHP but can be used by the external libraries that created them.

Arrays

Arrays are one of our favorite parts of PHP because the syntax is smart and easy to read and yet manages to be as powerful as you could want. There are four pieces of jargon you need to know to understand arrays:

- An array is made up of many *elements*.
- Each element has a *key* that defines its place in the array. An array can have only one element with a given key.
- Each element also has a *value*, which is the data associated with the key.
- Each array has a *cursor*, which points to the current key.

The first three are used regularly; the last one less so. The array cursor is covered later in this chapter in the section "Basic Functions," but we will look at the other three now. With PHP, your keys can be virtually anything: integers, strings, objects, or other arrays. You can even mix and match the keys so that one key is an array, another is a string, and so on. The one exception to all this is floating-point numbers: You cannot use floating-point numbers as keys in your arrays.

There are two ways of adding values to an array: with the `[]` operator, which is unique to arrays; and with the `array()` pseudo-function. You should use `[]` when you want to add items to an existing array and use `array()` to create a new array.

To sum all this up in code, Listing 25.2 shows a script that creates an array without specifying keys, adds various items to it both without keys and with keys of varying types, does a bit of printing, and then clears the array.

LISTING 25.2 Manipulating Arrays

```
<?php
  $myarr = array(1, 2, 3, 4);

  $myarr[4] = "Hello";
  $myarr[] = "World!";
  $myarr["elephant"] = "Wombat";
  $myarr["foo"] = array(5, 6, 7, 8);

  echo $myarr[2];
  echo $myarr["elephant"];
  echo $myarr["foo"][1];

  $myarr = array();
?>
```

The initial array is created with four elements, to which we assign the values 1, 2, 3, and 4. Because no keys are specified, PHP automatically assigns keys for us starting at 0 and counting upward—giving keys 0, 1, 2, and 3. Then we add a new element with the `[]` operator, specifying 4 as the key and `"Hello"` as the value. Next, `[]` is used again to add an element with the value `"World!"` and no key and then again to add an element with the key “elephant” and the value `"wombat"`. The line after that demonstrates using a string key with an array value—an array inside an array (a multidimensional array).

The next three lines demonstrate reading back from an array, first using a numeric key, then using a string key, and then using a string key and a numeric key. Remember, the `"foo"` element is an array in itself, so that third reading line retrieves the array and then prints the second element (arrays start at 0, remember). The last line blanks the array by simply using `array()` with no parameters, which creates an array with elements and assigns it to `$myarr`.

The following is an alternative way of using `array()` that allows you to specify keys along with their values:

```
$myarr = array("key1" => "value1", "key2" => "value2",
7 => "foo", 15 => "bar");
```

Which method you choose really depends on whether you want specific keys or want PHP to pick them for you.

Constants

Constants are frequently used in functions that require specific values to be passed in. For example, a popular function is `extract()`, which takes all the values in an array and places them into variables in their own right. You can choose to change the name of the variables as they are extracted using the second parameter—send it a 0 and it overwrites variables with the same names as those being extracted, send it a 1 and it skips variables with the same names, send it 5 and it prefixes variables only if they exist already, and so on. Of course, no one wants to have to remember a lot of numbers for each function, so you can instead use `EXTR_OVERWRITE` for 0, `EXTR_SKIP` for 1, `EXTR_PREFIX_IF_EXISTS` for 5, and so on, which is much easier.

You can create constants of your own by using the `define()` function. Unlike variables, constants do not start with a dollar sign, which makes the code to define a constant look like this:

```
<?php
  define("NUM_SQUIRRELS", 10);
  define("PLAYER_NAME", "Jim");
  define("NUM_SQUIRRELS_2", NUM_SQUIRRELS);
  echo NUM_SQUIRRELS_2;
?>
```

That script demonstrates how you can set constants to numbers, strings, or even the value of other constants, although that doesn't really get used much!

References

Using the equal sign (=) copies the value from one variable to another so they both have their own copy of the value. Another option here is to use references, which is where a variable does not have a value of its own; instead, it points to another variable. This enables you to share values and have variables mutually update themselves.

To copy by reference, use the `&` symbol, as follows:

```
<?php
  $a = 10;
  $b = &$a;
  echo $a . "\n";
  echo $b . "\n";

  $a = 20;
  echo $a . "\n";
  echo $b . "\n";

  $b = 30;
  echo $a . "\n";
  echo $b . "\n";
?>
```

If you run that script, you will see that updating `$a` also updates `$b`, but also that updating `$b` updates `$a`, too.

Comments

Adding short comments to your code is recommended and usually a requirement in larger software houses. In PHP you have three options for commenting style: `//`, `/* */`, and `#`. The first option (two slashes) instructs PHP to ignore everything until the end of the line. The second (a slash and an asterisk) instructs PHP to ignore everything until it reaches `*/`. The last (a hash symbol) works like `//` and is included because it is common among shell scripting languages.

This code example demonstrates the difference between `//` and `/* */`:

```
<?php
  echo "This is printed!";
  // echo "This is not printed";
  echo "This is printed!";
  /* echo "This is not printed";
  echo "This is not printed either"; */
?>
```

It is generally preferred to use `//` because it is a known quantity. On the other hand, it is easy to introduce coding errors with `/* */` by losing track of where a comment starts and ends.

NOTE

Contrary to popular belief, having comments in your PHP script has almost no effect on the speed at which the script executes. What little speed difference exists is wholly removed if you use a code cache.

Escape Sequences

Some characters cannot be typed, and yet you will almost certainly want to use some of them from time to time. For example, you might want to use an ASCII character for new line, but you can't type it. Instead, you need to use an escape sequence: `\n`. Similarly, you can print a carriage return character with `\r`. It's important to know both of these because, on the Windows platform, you need to use `\r\n` to get a new line. If you do not plan to run your scripts anywhere else, you need not worry about this!

Going back to the first script we wrote, recall that it printed 2010 because we added 10 + 10 and then 10 + 0. We can rewrite that using escape sequences, like this:

```
<?php
  $i = 10;
  $j = "10";
  $k = "Hello, world";
  echo $i + $j;
  echo "\n";
  echo $i + $k;
  echo "\n";
?>
```

This time PHP prints a new line after each of the numbers, making it obvious that the output is 20 and 10 rather than 2010. Note that the escape sequences must be used in double quotation marks because they will not work in single quotation marks.

Three common escape sequences are `\\`, which means "ignore the backslash"; `\"`, which means "ignore the double quote"; and `\'`, which means "ignore the single quote." This is important when strings include quotation marks inside them. If we had a string such as `"are you really Bill O'Reilly?"`, which has a single quotation mark in, this code would not work:

```
<?php
  echo 'Are you really Bill O'Reilly?';
?>
```

PHP would see the opening quotation mark, read all the way up to the *O* in O'Reilly, and then see the quotation mark following the *O* as being the end of the string. The `Reilly?` part would appear to be a fragment of text and would cause an error. You have two options here: You can either surround the string in double quotation marks or escape the single quotation mark with `\'`.

If you choose the escaping route, it will look like this:

```
echo 'Are you really Bill O\'Reilly?';
```

Although they are a clean solution for small text strings, be careful with overusing escape sequences. HTML is particularly full of quotation marks, and escaping them can get messy:

```
$mystring = "<img src=\"foo.png\" alt=\"My picture\"
width=\"100\" height=\"200\" />";
```

In that situation, you are better off using single quotation marks to surround the text simply because it is a great deal easier on the eye!

Variable Substitution

PHP allows you to define strings using three methods: single quotation marks, double quotation marks, or heredoc notation. Heredoc isn't discussed here because it's fairly rare compared to the other two methods, but single quotation marks and double quotation marks work identically, with one minor exception—variable substitution.

Consider the following code:

```
<?php
  $age = 25
  echo "You are ";
  echo $age;
?>
```

That is a particularly clumsy way to print a variable as part of a string. Fortunately, if you put a variable inside a string, PHP performs *variable substitution*, replacing the variable with its value. That means we can rewrite the code like so:

```
<?php
  $age = 25
  echo "You are $age";
?>
```

The output is the same. The difference between single quotation marks and double quotation marks is that single-quoted strings do not have their variables substituted. Here's an example:

```
<?php
  $age = 25
  echo "You are $age";
  echo 'You are $age';
?>
```

The first `echo` prints `"You are 25"`, but the second one prints `"You are $age"`.

Operators

Now that we have data values to work with, we need some operators to use, too. We have already used + to add variables together, but many others in PHP handle arithmetic, comparison, assignment, and other operators. *Operator* is just a fancy word for something that performs an operation, such as addition or subtraction. However, *operand* might be new to you. Consider this operation:

```
$a = $b + c;
```

In this operation, `=` and `+` are operators, and `$a`, `$b`, and `$c` are operands. Along with `+`, you will also already know – (subtract), `*` (multiply), and `/` (divide), but here are some more.

Operator	What It Does
`=`	Assigns the right operand to the left operand.
`==`	Returns `true` if the left operand is equal to the right operand.
`!=`	Returns `true` if the left operand is not equal to the right operand.
`===`	Returns `true` if the left operand is identical to the right operand. This is not the same as `==`.
`!==`	Returns `true` if the left operand is not identical to the right operand. This is not the same as `!=`.

Operator	What It Does
`<`	Returns `true` if the left operand is smaller than the right operand.
`>`	Returns `true` if the left operand is greater than the right operand.
`<=`	Returns `true` if the left operand is equal to or smaller than the right operand.
`&&`	Returns `true` if both the left operand and the right operand are true.
`¦¦`	Returns `true` if either the left operand or the right operand is true.
`++`	Increments the operand by one.
`--`	Decrements the operand by one.
`+=`	Increments the left operand by the right operand.
`-=`	Decrements the left operand by the right operand.
`.`	Concatenates the left operand and the right operand (joins them together).
`%`	Divides the left operand by the right operand and returns the remainder.
`¦`	Performs a bitwise `OR` operation. It returns a number with bits that are set in either the left operand or the right operand.
`&`	Performs a bitwise `AND` operation. It returns a number with bits that are set both in the left operand and the right operand.

At least 10 other operators are not listed; to be fair, however, you're unlikely to use them. Even some of the ones in this list are used infrequently—bitwise `AND`, for example. Having said that, the bitwise `OR` operator is used regularly because it allows you to combine values.

Here is a code example demonstrating some of the operators:

```
<?php
  $i = 100;
  $i++; // $i is now 101
  $i--; // $i is now 100 again
  $i += 10; // $i is 110
  $i = $i / 2; // $i is 55
  $j = $i; // both $j and $i are 55
  $i = $j % 11; // $i is 0
?>
```

The last line uses modulus, which takes some people a little bit of effort to understand. The result of `$i % 11` is `0` because `$i` is set to `55` and modulus works by dividing the left operand (`55`) by the right operand (`11`) and returning the remainder. 55 divides by 11 exactly 5 times, and so has the remainder 0.

The concatenation operator, a period, sounds scarier than it is: It just joins strings together. For example:

```
<?php
  echo "Hello, " . "world!";
  echo "Hello, world!" . "\n";
?>
```

There are two “special” operators in PHP that are not covered here and yet are used frequently. Before we look at them, though, it’s important that you see how the comparison operators (such as <, <=, and !=) are used inside conditional statements.

Conditional Statements

In a *conditional statement* you instruct PHP to take different actions depending on the outcome of a test. For example, you might want PHP to check whether a variable is greater than 10 and, if so, print a message. This is all done with the `if` statement, which looks like this:

```
if (your condition) {
  // action to take if condition is true
} else {
  // optional action to take otherwise
}
```

The *`your condition`* part can be filled with any number of conditions you want PHP to evaluate, and this is where the comparison operators come into their own. For example:

```
if ($i > 10) {
  echo "11 or higher";
} else {
  echo "10 or lower";
}
```

PHP looks at the condition and compares `$i` to 10. If it is greater than 10, it replaces the whole operation with `1`; otherwise, it replaces it with `0`. So, if `$i` is 20, the result looks like this:

```
if (1) {
  echo "11 or higher";
} else {
  echo "10 or lower";
}
```

In conditional statements, any number other than 0 is considered to be equivalent to the Boolean value `true`, so `if (1)` always evaluates to `true`. There is a similar case for strings—if your string has any characters in, it evaluates to `true`, with empty strings evaluating to `false`. This is important because you can then use that `1` in another condition through `&&` or `¦¦` operators. For example, if you want to check whether `$i` is greater than 10 but less than 40, you could write this:

```
if ($i > 10 && $i < 40) {
  echo "11 or higher";
} else {
  echo "10 or lower";
}
```

If we presume that `$i` is set to `50`, the first condition (`$i, 10`) is replaced with `1` and the second condition (`$i < 40`) is replaced with `0`. Those two numbers are then used by the `&&` operator, which requires both the left and right operands to be *true*. Whereas `1` is equivalent to `true`, `0` is not, so the `&&` operand is replaced with `0` and the condition fails.

=, ==, ===, and similar operators are easily confused and often the source of programming errors. The first, a single equal sign, assigns the value of the right operand to the left operand. However, all too often you see code like this:

```
if ($i = 10) {
  echo "The variable is equal to 10!";
} else {
  echo "The variable is not equal to 10";
}
```

That is incorrect. Rather than checking whether `$i` is equal to 10, it assigns `10` to `$i` and returns `true`. What is needed is ==, which compares two values for equality. In PHP, this is extended so that there is also === (three equal signs), which checks whether two values are identical—more than just equal.

The difference is slight but important: If you have a variable with the string value `"10"` and compare it against the number value of 10, they are equal. Thus, PHP converts the type and checks the numbers. However, they are not identical. To be considered identical, the two variables must be equal (that is, have the same value) and be of the same data type (that is, both are strings, both are integers, and so on).

NOTE

It is common practice to put function calls in conditional statements rather than direct comparisons. For example:

```
if (do_something()) {
```

If the `do_something()` function returns `true` (or something equivalent to `true`, such as a nonzero number), the conditional statement evaluates to `true`.

Special Operators

The ternary operator and the execution operator work differently from those we have seen so far. The ternary operator is rarely used in PHP, thankfully, because it is really just a condensed conditional statement. Presumably it arose through someone needing to make his code occupy as little space as possible because it certainly does not make PHP code any easier to read!

The ternary operator works like this:

```
$age_description = ($age < 18) ? "child" : "adult";
```

Without explanation, that code is essentially meaningless; however, it expands into the following five lines of code:

```
if ($age < 18) {
  $age_description = "child";
} else {
  $age_description = "adult";
}
```

The ternary operator is so named because it has three operands: a condition to check (`$age < 18` in the previous code), a result if the condition is true (`"child"`), and a result if the condition is false (`"adult"`). Although we hope you never have to use the ternary operator, it is at least important to know how it works in case you stumble across it.

The other special operator is the execution operator, which is the backtick symbol, `` ` ``. The position of the backtick key varies depending on your keyboard, but it is likely to be just to the left of the 1 key (above Tab). The execution operator executes the program inside the back ticks, returning any text the program outputs. For example:

```
<?php
  $i = `ls -l`;
  echo $i;
?>
```

That executes the `ls` program, passing in `-l` (a lowercase *L*) to get the long format, and stores all its output in `$i`. You can make the command as long or as complex as you like, including piping to other programs. You can also use PHP variables inside the command.

Switching

Having multiple `if` statements in one place is ugly, slow, and prone to errors. Consider the code in Listing 25.3.

LISTING 25.3 How Multiple Conditional Statements Lead to Ugly Code

```
<?php
  $cat_age = 3;

  if ($cat_age == 1) {
    echo "Cat age is 1";
  } else {
    if ($cat_age == 2) {
      echo "Cat age is 2";
    } else {
      if ($cat_age == 3) {
        echo "Cat age is 3";
      } else {
        if ($cat_age == 4) {
```

LISTING 25.3 Continued

```
          echo "Cat age is 4";
        } else {
          echo "Cat age is unknown";
        }
      }
    }
  }
?>
```

Even though it certainly works, it is a poor solution to the problem. Much better is a `switch/case` block, which transforms the previous code into what's shown in Listing 25.4.

LISTING 25.4 Using a `switch/case` Block

```
<?php
  $cat_age = 3;

  switch ($cat_age) {
    case 1:
      echo "Cat age is 1";
      break;
    case 2:
      echo "Cat age is 2";
      break;
    case 3:
      echo "Cat age is 3";
      break;
    case 4:
      echo "Cat age is 4";
      break;
    default:
      echo "Cat age is unknown";
  }
?>
```

Although it is only slightly shorter, it is a great deal more readable and much easier to maintain. A `switch/case` group is made up of a `switch()` statement in which you provide the variable you want to check, followed by numerous `case` statements. Notice the `break` statement at the end of each `case`. Without that, PHP would execute each `case` statement beneath the one it matches. Calling `break` causes PHP to exit the `switch/case`. Notice also that there is a `default` `case` at the end that catches everything that has no matching case.

It is important that you do not use `case default:` but merely `default:`. Also, it is the last `case` label, so it has no need for a `break` statement because PHP exits the `switch/case` block there anyway.

Loops

PHP has four ways you can execute a block of code multiple times: `while`, `for`, `foreach`, and `do...while`. Of the four, only `do...while` sees little use; the others are popular and you will certainly encounter them in other people's scripts.

The most basic loop is the `while` loop, which executes a block of code for as long as a given condition is true. So, we can write an infinite loop—a block of code that continues forever—with this PHP:

```
<?php
  $i = 10;
  while ($i >= 10) {
    $i += 1;
    echo $i;
  }
?>
```

The loop block checks whether `$i` is greater or equal to 10 and, if that condition is true, adds 1 to `$i` and prints it. Then it goes back to the loop condition again. Because `$i` starts at 10 and we only ever add numbers to it, that loop continues forever. With two small changes, we can make the loop count down from 10 to 0:

```
<?php
  $i = 10;
  while ($i >= 0) {
    $i -= 1;
    echo $i;
  }
?>
```

So, this time we check whether `$i` is greater than or equal to 0 and subtract 1 from it with each loop iteration. `while` loops are typically used when you are unsure of how many times the code needs to loop because `while` keeps looping until an external factor stops it.

With a `for` loop, you specify precise limits on its operation by giving it a declaration, a condition, and an action. That is, you specify one or more variables that should be set when the loop first runs (the *declaration*), you set the circumstances that will cause the loop to terminate (the *condition*), and you tell PHP what it should change with each loop iteration (the *action*). That last part is what really sets a `for` loop apart from a `while` loop: You usually tell PHP to change the condition variable with each iteration.

We can rewrite the script that counts down from 10 to 0 using a `for` loop:

```
<?php
  for($i = 10; $i >= 0; $i -= 1) {
    echo $i;
  }
?>
```

This time we do not need to specify the initial value for `$i` outside the loop, and neither do we need to change `$i` inside the loop—it is all part of the `for` statement. The actual amount of code is really the same, but for this purpose the `for` loop is arguably tidier and therefore easier to read. With the `while` loop, the `$i` variable was declared outside the loop and so was not explicitly attached to the loop.

The third loop type is `foreach`, which is specifically for arrays and objects, although it is rarely used for anything other than arrays. A `foreach` loop iterates through each element in an array (or each variable in an object), optionally providing both the key name and the value.

In its simplest form, a `foreach` loop looks like this:

```
<?php
  foreach($myarr as $value) {
    echo $value;
  }
?>
```

This loops through the `$myarr` array we created earlier, placing each value in the `$value` variable. We can modify that so we get the keys as well as the values from the array, like this:

```
<?php
  foreach($myarr as $key => $value) {
    echo "$key is set to $value\n";
  }
?>
```

As you can guess, this time the array keys go in `$key` and the array values go in `$value`. One important characteristic of the `foreach` loop is that it goes from the start of the array to the end and then stops—and by *start* we mean the first item to be added rather than the lowest index number. This script shows this behavior:

```
<?php
  $array = array(6 => "Hello", 4 => "World",
                 2 => "Wom", 0 => "Bat");
  foreach($array as $key => $value) {
    echo "$key is set to $value\n";
  }
?>
```

If you try this script, you will see that foreach prints the array in the original order of 6, 4, 2, 0 rather than the numeric order of 0, 2, 4, 6.

The do...while loop works like the while loop, with the exception that the condition appears at the end of the code block. This small syntactical difference means a lot, though, because a do...while loop is always executed at least once. Consider this script:

```
<?php
  $i = 10;
  do {
    $i -= 1;
    echo $i;
  } while ($i < 10);
?>
```

Without running the script, what do you think it will do? One possibility is that it will do nothing; $i is set to 10, and the condition states that the code must loop only while $i is less than 10. However, a do...while loop always executes once, so what happens is that $i is set to 10 and PHP enters the loop, decrements $i, prints it, and then checks the condition for the first time. At this point, $i is indeed less than 10, so the code loops, $i is decremented again, the condition is rechecked, $i is decremented again, and so on. This is in fact an infinite loop, and so should be avoided!

If you ever want to exit a loop before it has finished, you can use the same break statement, which we used earlier to exit a switch/case block. This becomes more interesting if you find yourself with *nested* loops—loops inside of loops. This is a common situation to be in. For example, you might want to loop through all the rows in a chessboard and, for each row, loop through each column. Calling break exits only one loop or switch/case, but you can use break 2 to exit two loops or switch/cases, or break 3 to exit three, and so on.

Including Other Files

Unless you are restricting yourself to the simplest programming ventures, you will want to share code among your scripts at some point. The most basic need for this is to have a standard header and footer for your website, with only the body content changing. However, you might also find yourself with a small set of custom functions you use frequently, and it would be an incredibly bad move to simply copy and paste the functions into each of the scripts that use them.

The most common way to include other files is with the include keyword. Save this script as include1.php:

```
<?php
  for($i = 10; $i >= 0; $i -= 1) {
    include "echo_i.php";
  }
?>
```

Then save this script as `echo_i.php`:

```
<?php
  echo $i;
?>
```

If you run `include1.php`, PHP loops from 10 to 0 and includes `echo_i.php` each time. For its part, `echo_i.php` just prints the value of `$i`, which is a crazy way of performing an otherwise simple operation, but it does demonstrate how included files share data. Note that the `include` keyword in `include1.php` is inside a PHP block, but we reopen PHP inside `echo_i.php`. This is important because PHP exits PHP mode for each new file, so you always have a consistent entry point.

Basic Functions

PHP has a vast number of built-in functions that enable you to manipulate strings, connect to databases, and more. There is not room here to cover even 10% of the functions; for more detailed coverage of functions, check the "Reference" section at the end of this chapter.

Strings

Several important functions are used for working with strings, and there are many more less frequently used ones for which there is not enough space here. We are going to look at the most important here, ordered by difficulty—easiest first!

The easiest function is `strlen()`, which takes a string as its parameter and returns the number of characters in there, like this:

```
<?php
  $ourstring = "  The Quick Brown Box Jumped Over The Lazy Dog  ";
  echo strlen($ourstring);
?>
```

We will be using that same string in subsequent examples to save space. If you execute that script, it outputs 48 because 48 characters are in the string. Note the 2 spaces on either side of the text, which pad the 44-character phrase up to 48 characters.

We can fix that padding with the `trim()` function, which takes a string to trim and returns it with all the whitespace removed from either side. This is a commonly used function because all too often you encounter strings that have an extra new line at the end or a space at the beginning. This cleans it up perfectly.

Using `trim()`, we can turn our 48-character string into a 44-character string (the same thing, without the extra spaces), like this:

```
echo trim($ourstring);
```

Keep in mind that `trim()` returns the trimmed string, so that outputs `"The Quick Brown Box Jumped Over The Lazy Dog"`. We can modify that so `trim()` passes its return value to `strlen()` so that the code trims it and then outputs its trimmed length:

```
echo strlen(trim($ourstring));
```

PHP always executes the innermost functions first, so the previous code takes `$ourstring`, passes it through `trim()`, uses the return value of `trim()` as the parameter for `strlen()`, and prints it.

Of course, everyone knows that boxes do not jump over dogs—the usual phrase is "the quick brown fox." Fortunately, there is a function to fix that problem: `str_replace()`. Note that it has an underscore in it; PHP is inconsistent on this matter, so you really need to memorize the function name.

The `str_replace()` function takes three parameters: the text to search for, the text to replace it with, and the string you want to work with. When working with search functions, people often talk about *needles* and *haystacks*—in this situation, the first parameter is the needle (the thing to find), and the third parameter is the haystack (what you are searching through).

So, we can fix our error and correct *box* to *fox* with this code:

```
echo str_replace("Box", "Fox", $ourstring);
```

There are two little addendums to make here. First, note that we have specified `"Box"` as opposed to `"box"` because that is how it appears in the text. The `str_replace()` function is a *case-sensitive* function, which means it does not consider `"Box"` to be the same as `"box"`. If you want to do a non-case-sensitive search and replace, you can use the `stri_replace()` function, which works in the same way.

The second addendum is that, because we are actually changing only one character (*B* to *F*), we need not use a function at all. PHP enables you to read (and change) individual characters of a string by specifying the character position inside braces (`{` and `}`). As with arrays, strings are zero based, which means in the `$ourstring` variable `$ourstring{0}` is *T*, `$ourstring{1}` is *h*, `$ourstring{2}` is *e*, and so on. We could use this instead of `str_replace()`, like this:

```
<?php
  $ourstring = "  The Quick Brown Box Jumped Over The Lazy Dog  ";
  $ourstring{18} = "F";
  echo $ourstring;
?>
```

You can extract part of a string using the `substr()` function, which takes a string as its first parameter, a start position as its second parameter, and an optional length as its third parameter. Optional parameters are common in PHP. If you do not provide them, PHP assumes a default value. In this case, if you specify only the first two parameters, PHP copies from the start position to the end of the string. If you specify the third parameter, PHP copies that many characters from the start.

We can write a simple script to print `"Lazy Dog   "` by setting the start position to 38, which, remembering that PHP starts counting string positions from 0, copies from the 39th character to the end of the string:

```
echo substr($ourstring, 38);
```

If we just want to print the word `"Lazy,"` we need to use the optional third parameter to specify the length as 4, like this:

```
echo substr($ourstring, 38, 4);
```

The `substr()` function can also be used with negative second and third parameters. If you specify just parameter one and two and provide a negative number for parameter two, `substr()` counts backward from the end of the string. So, rather than specifying 38 for the second parameter, we can use –10, so it takes the last 10 characters from the string. Using a negative second parameter and positive third parameter counts backward from the end string and then uses a forward length. We can print `"Lazy"` by counting 10 characters back from the end and then taking the next four characters forward:

```
echo substr($ourstring, -10, 4);
```

Finally, we can use a negative third parameter, too, which also counts back from the end of the string. For example, using `"-4"` as the third parameter means to take everything except the last four characters. Confused yet? This code example should make it clear:

```
echo substr($ourstring, -19, -11);
```

That counts 19 characters backward from the end of the string (which places it at the *O* in Over) and then copies everything from there until 11 characters before the end of the string. That prints `"Over The"`. The same thing could be written using –19 and 8, or even 29 and 8—there is more than one way to do it!

Moving on, the `strpos()` function returns the position of a particular substring inside a string; however, it is most commonly used to answer the question, "Does this string contain a specific substring?" You need to pass it two parameters: a haystack and a needle (yes, that's a different order from `str_replace()`!).

In its most basic use, `strpos()` can find the first instance of `"Box"` in our phrase, like this:

```
echo strpos($ourstring, "Box");
```

This outputs 18 because that is where the *B* in Box starts. If `strpos()` cannot find the substring in the parent string, it returns `false` rather than the position. Much more helpful, though, is the ability to check whether a string contains a substring; a first attempt to check whether our string contains the word *The* might look like this:

```
<?php
  $ourstring = "The Quick Brown Box Jumped Over The Lazy Dog";
  if (strpos($ourstring, "The")) {
    echo "Found 'The'!\n";
  } else {
    echo "'The' not found!\n";
  }
?>
```

Note that we have temporarily taken out the leading and trailing whitespace from `$ourstring` and we are using the return value of `strpos()` for our conditional statement. This reads, "If the string is found, then print a message; if not, print another message." Or does it?

Run the script, and you will see it print the `"not found"` message. The reason for this is that `strpos()` returns `false` if the substring is not found and otherwise returns the position where it starts. If you recall, any nonzero number equates to `true` in PHP, which means that `0` equates to `false`. With that in mind, what is the string index of the first *The* in our phrase? Because PHP's strings are zero based and we no longer have the spaces on either side of the string, the *The* is at position 0, which our conditional statement evaluates to `false`—hence, the problem.

The solution here is to check for identicality. We know that `0` and `false` are equal, but they are not identical because `0` is an integer, whereas `false` is a Boolean. So, we need to rewrite the conditional statement to see whether the return value from `strpos()` is identical to `false`. If it is, the substring was not found:

```
<?php
  $ourstring = "The Quick Brown Box Jumped Over The Lazy Dog";
  if (strpos($ourstring, "The") !== false) {
    echo "Found 'The'!\n";
  } else {
    echo "'The' not found!\n";
  }
?>
```

Arrays

Working with arrays is no easy task, but PHP makes it easier by providing a selection of functions that can sort, shuffle, intersect, and filter them. As with other functions, there is only space here to choose a selection; this is by no means a definitive reference to PHP's array functions.

The easiest function to use is `array_unique()`, which takes an array as its only parameter and returns the same array with all duplicate values removed. Also in the realm of "so easy you do not need a code example" is the `shuffle()` function, which takes an array as its parameter and randomizes the order of its elements. Note that `shuffle()` does not return the randomized array; it uses your parameter as a reference and scrambles it directly. The last too-easy-to-demonstrate function is `in_array()`, which takes a value as its first parameter and an array as its second and returns `true` if the value is in the array.

With those out of the way, we can focus on the more interesting functions, two of which are `array_keys()` and `array_values()`. They both take an array as their only parameter and return a new array made up of the keys in the array or the values of the array, respectively. The `array_values()` function is an easy way to create a new array of the same data, just without the keys. This is often used if you have numbered your array keys, deleted several elements, and want to reorder it.

The `array_keys()` function creates a new array where the values are the keys from the old array, like this:

```
<?php
  $myarr = array("foo" => "red", "bar" => "blue", "baz" => "green");
  $mykeys = array_keys($myarr);
  foreach($mykeys as $key => $value) {
    echo "$key = $value\n";
  }
?>
```

That prints `"0 = foo"`, `"1 = bar"`, and `"2 = baz"`.

Several functions are used specifically for array sorting, but only two get much use: `asort()` and `ksort()`, the first of which sorts the array by its values and the second of which sorts the array by its keys. Given the array `$myarr` from the previous example, sorting by the values would produce an array with elements in the order bar/blue, baz/green, and foo/red. Sorting by key would give the elements in the order bar/blue, baz/green, and foo/red. As with the `shuffle()` function, both `asort()` and `ksort()` do their work *in place*, meaning they return no value, directly altering the parameter you pass in. For interest's sake, you can also use `arsort()` and `krsort()` for reverse value sorting and reverse key sorting, respectively.

This code example reverse sorts the array by value and then prints it as before:

```
<?php
  $myarr = array("foo" => "red", "bar" => "blue", "baz" => "green");
  arsort($myarr);
  foreach($myarr as $key => $value) {
    echo "$key = $value\n";
  }
?>
```

Previously when discussing constants, we mentioned the `extract()` function that converts an array into individual variables; now it is time to start using it for real. You need to provide three variables: the array you want to extract, how you want the variables prefixed, and the prefix you want used. Technically, the last two parameters are optional, but practically you should always use them to properly namespace your variables and keep them organized.

The second parameter must be one of the following:

- EXTR_OVERWRITE—If the variable exists already, overwrites it.
- EXTR_SKIP—If the variable exists already, skips it and moves on to the next variable.
- EXTR_PREFIX_SAME—If the variable exists already, uses the prefix specified in the third parameter.
- EXTR_PREFIX_ALL—Prefixes all variables with the prefix in the third parameter, regardless of whether it exists already.
- EXTR_PREFIX_INVALID—Uses a prefix only if the variable name would be invalid (for example, starting with a number).
- EXTR_IF_EXISTS—Extracts only variables that already exist. We have never seen this used.

You can also, optionally, use the bitwise OR operator, ¦, to add in EXTR_REFS to have extract() use references for the extracted variables. In general use, EXTR_PREFIX_ALL is preferred because it guarantees namespacing. EXTR_REFS is required only if you need to be able to change the variables and have those changes reflected in the array.

This next script uses extract() to convert $myarr into individual variables, $arr_foo, $arr_bar, and $arr_baz:

```
<?php
  $myarr = array("foo" => "red", "bar" => "blue", "baz" => "green");
  extract($myarr, EXTR_PREFIX_ALL, 'arr');
?>
```

Note that the array keys are "foo", "bar", and "baz" and that the prefix is "arr", but that the final variables will be $arr_foo, $arr_bar, and $arr_baz. PHP inserts an underscore between the prefix and array key.

Files

As you will have learned from elsewhere in the book, the UNIX philosophy is that everything is a file. In PHP, this is also the case: A selection of basic file functions is suitable for opening and manipulating files, but those same functions can also be used for opening and manipulating network sockets. We cover both here.

Two basic read and write functions for files make performing these basic operations easy. They are file_get_contents(), which takes a filename as its only parameter and returns the file’s contents as a string, and file_put_contents(), which takes a filename as its first parameter and the data to write as its second parameter.

Using these two, we can write a script that reads all the text from one file, `filea.txt`, and writes it to another, `fileb.txt`:

```
<?php
  $text = file_get_contents("filea.txt");
  file_put_contents("fileb.txt", $text);
?>
```

Because PHP enables us to treat network sockets like files, we can also use `file_get_contents()` to read text from a website, like this:

```
<?php
  $text = file_get_contents("http://www.slashdot.org");
  file_put_contents("fileb.txt", $text);
?>
```

The problem with using `file_get_contents()` is that it loads the whole file into memory at once; that's not practical if you have large files or even smaller files being accessed by many users. An alternative is to load the file piece by piece, which can be accomplished through the following five functions: `fopen()`, `fclose()`, `fread()`, `fwrite()`, and `feof()`. The *f* in those function names stands for *file*, so they open, close, read from, and write to files and sockets. The last function, `feof()`, returns `true` if the end of the file has been reached.

The `fopen()` function takes a bit of learning to use properly, but on the surface it looks straightforward. Its first parameter is the filename you want to open, which is easy enough. However, the second parameter is where you specify how you want to work with the file, and you should specify one of the following:

- `r`—Read only; it overwrites the file
- `r+`—Reading and writing; it overwrites the file
- `w`—Write only; it erases the existing contents and overwrites the file
- `w+`—Reading and writing; it erases the existing content and overwrites the file
- `a`—Write only; it appends to the file
- `a+`—Reading and writing; it appends to the file
- `x`—Write only, but only if the file does not exist
- `a+`—Reading and writing, but only if the file does not exist

Optionally, you can also add `b` (for example, `a+b` or `rb`) to switch to binary mode. This is recommended if you want your scripts and the files they write to work smoothly on other platforms.

When you call `fopen()`, you should store the return value. It is a resource known as a *file handle*, which the other file functions all need to do their jobs. The `fread()` function, for example, takes the file handle as its first parameter and the number of bytes to read as its second, returning the content in its return value. The `fclose()` function takes the file handle as its only parameter and frees up the file.

So, we can write a simple loop to open a file, read it piece by piece, print the pieces, and then close the handle:

```
<?php
  $file = fopen("filea.txt", "rb");
  while (!feof($file)) {
    $content = fread($file, 1024);
    echo $content;
  }
  fclose($file);
?>
```

That only leaves the `fwrite()` function, which takes the file handle as its first parameter and the string to write as its second. You can also provide an integer as the third parameter, specifying the number of bytes you want to write of the string, but if you exclude this, `fwrite()` writes the entire string.

If you recall, you can use `a` as the second parameter to `fopen()` to append data to a file. So, we can combine that with `fwrite()` to have a script that adds a line of text to a file each time it is executed:

```
<?php
  $file = fopen("filea.txt", "ab");
  fwrite($file, "Testing\n");
  fclose($file);
?>
```

To make that script a little more exciting, we can stir in a new function, `filesize()`, that takes a filename (not a file handle, but an actual filename string) as its only parameter and returns the file's size in bytes. Using that new function brings the script to this:

```
<?php
  $file = fopen("filea.txt", "ab");
  fwrite($file, "The filesize was" . filesize("filea.txt") . "\n");
  fclose($file);
?>
```

Although PHP automatically cleans up file handles for you, it is still best to use `fclose()` yourself so you are always in control.

Miscellaneous

Several functions do not fall under the other categories and so are covered here. The first one is `isset()`, which takes one or more variables as its parameters and returns `true` if they have been set. It is important to note that a variable with a value set to something that would be evaluated to `false`—such as `0` or an empty string—still returns `true` from `isset()` because it does not check the value of the variable. It merely checks that it is set; hence, the name.

The `unset()` function also takes one or more variables as its parameters, simply deleting the variable and freeing up the memory. With these two, we can write a script that checks for the existence of a variable and, if it exists, deletes it (see Listing 25.5).

LISTING 25.5 Setting and Unsetting Variables

```
<?php
  $name = "Ildiko";
  if (isset($name)) {
    echo "Name was set to $name\n";
    unset($name);
  } else {
    echo "Name was not set";
  }

  if (isset($name)) {
    echo "Name was set to $name\n";
    unset($name);
  } else {
    echo "Name was not set";
  }
?>
```

That script runs the same `isset()` check twice, but it `unset()`s the variable after the first check. As such, it prints `"Name was set to Ildiko"` and then `"Name was not set"`.

Perhaps the most frequently used function in PHP is `exit`, although purists will tell you that it is in fact a language construct rather than a function. `exit` terminates the processing of the script as soon as it is executed, meaning subsequent lines of code are not executed. That is really all there is to it; it barely deserves an example, but here is one just to make sure:

```
<?php
  exit;
  echo "Exit is a language construct!\n";
?>
```

That script prints nothing because the `exit` comes before the `echo`.

One function we can guarantee you will use a lot is `var_dump()`, which dumps out information about a variable, including its value, to the screen. This is invaluable for arrays because it prints every value and, if one or more of the elements is an array, it prints all the elements from those, and so on. To use this function, just pass it a variable as its only parameter:

```
<?php
  $drones = array("Graham", "Julian", "Nick", "Paul");
  var_dump($drones);
?>
```

The output from that script looks like this:

```
array(4) {
  [0]=>
  string(6) "Graham"
  [1]=>
  string(6) "Julian"
  [2]=>
  string(4) "Nick"
  [3]=>
  string(4) "Paul"
}
```

The `var_dump()` function sees a lot of use as a basic debugging technique because it is the easiest way to print variable data to the screen to verify it.

Finally, we briefly discuss regular expressions, with the emphasis on *briefly* because regular expression syntax is covered elsewhere in this book and the only unique thing relevant to PHP are the functions you will use to run the expressions. You have the choice of either *Perl-Compatible Regular Expressions (PCRE)* or POSIX Extended regular expressions, but there really is little to choose between them in terms of functionality offered. For this chapter, we use the PCRE expressions because, to the best of our knowledge, they see more use by other PHP programmers.

The main PCRE functions are `preg_match()`, `preg_match_all()`, `preg_replace()`, and `preg_split()`. We will start with `preg_match()` because it provides the most basic functionality by returning `true` if one string matches a regular expression. The first parameter to `preg_match()` is the regular expression you want to search for, and the second is the string to match. So, if we wanted to check whether a string had the word *Best*, *Test*, *rest*, *zest*, or any other word containing *est* preceded by any letter of either case, we could use this PHP code:

```
$result = preg_match("/[A-Za-z]est/", "This is a test");
```

Because the test string matches the expression, `$result` is set to `1` (true). If you change the string to a nonmatching result, you get `0` as the return value.

The next function is preg_match_all(), which gives you an array of all the matches it found. However, to be most useful, it takes the array to fill with matches as a by-reference parameter and saves its return value for the number of matches that were found.

We suggest you use preg_match_all() and var_dump() to get a feel for how the function works. This example is a good place to start:

```
<?php
  $string = "This is the best test in the west";
  $result = preg_match_all("/[A-Za-z]est/", $string, $matches);
  var_dump($matches);
?>
```

That outputs the following:

```
array(1) {
  [0]=>
  array(3) {
    [0]=>
    string(4) "best"
    [1]=>
    string(4) "test"
    [2]=>
    string(4) "west"
  }
}
```

If you notice, the $matches array is actually multidimensional in that it contains one element, which itself is an array containing all the matches to our regular expression. The reason for this is because our expression has no *subexpressions*, meaning no independent matches using parentheses. If we had subexpressions, each would have its own element in the $matches array containing its own array of matches.

Moving on, preg_replace() is used to change all substrings that match a regular expression into something else. The basic manner of using this is quite easy: You search for something with a regular expression and provide a replacement for it. However, a more useful variant is *backreferencing*, using the match as part of the replacement. For our example, we will imagine you have written a tutorial on PHP but want to process the text so each reference to a function is followed by a link to the PHP manual.

PHP manual page URLs take the form http://www.php.net/<somefunc>—for example, http://www.php.net/preg_replace. The string we need to match is a function name, which is a string of alphabetic characters, potentially also mixed with numbers and underscores and terminated with two parentheses, (). As a replacement, we will use the match we found, surrounded in HTML emphasis tags (<em></em>), and then with a link to the relevant PHP manual page. Here is how that looks in code:

```
<?php
  $regex = "/([A-Za-z0-9_]*)\(\)/";
  $replace = "<em>$1</em> (<a href=\"http://www.php.net/$1\">manual</A>)";
  $haystack = "File_get_contents()is easier than using fopen().";
  $result = preg_replace($regex, $replace, $haystack);
  echo $result;
?>
```

The `$1` is our backreference; it will be substituted with the results from the first subexpression. The way we have written the regular expression is very exact. The `[A-Za-z0-9_]*` part, which matches the function name, is marked as a subexpression. After that is `\(\)`, which means the exact symbols (and), not the regular expression meanings of them, which means that `$1` in the replacement will contain `fopen` rather than `fopen()`, which is how it should be. Of course, anything that is not backreferenced in the replacement is removed, so we have to put the `()` after the first `$1` (not in the hyperlink) to repair the function name.

After all that work, the output is perfect:

```
<em>File_get_contents()</em> (<a href="http://www.php.net/
file_get_contents">manual</A>) is easier than using <em>fopen()
</em> (<a href="http://www.php.net/fopen">manual</A>).
```

Handling HTML Forms

Given that PHP's primary role is handling web pages, you might wonder why this section has been left so late in the chapter. It is because handling HTML forms is so central to PHP that it is essentially automatic.

Consider this form:

```
<form method="POST" action="thispage.php">
User ID: <input type="text" name="UserID" /><br />
Password: <input type="password" name="Password" /><br />
<input type="submit" />
</form>
```

When a visitor clicks Submit, `thispage.php` is called again and this time PHP has the variables available to it inside the `$_REQUEST` array. Given that script, if the user enters `12345` and `frosties` as her user ID and password, PHP provides you with `$_REQUEST['UserID']` set to `12345` and `$_REQUEST['Password']` set to `frosties`. Note that it is important that you use HTTP post unless you specifically want `GET`. `POST` enables you to send a great deal more data and stops people from tampering with your URL to try to find holes in your script.

Is that it? Well, almost. That tells you how to retrieve user data, but you should be sure to sanitize it so users do not try to sneak HTML or JavaScript into your database as something you think is innocuous. PHP gives you the `strip_tags()` function for this purpose. It takes a string and returns the same string with all HTML tags removed.

Databases

The ease with which PHP can be used to create dynamic, database-driven websites is for many the key reason to use it. The stock build of PHP comes with support for MySQL, PostgreSQL, SQLite, Oracle, Microsoft SQL Server, ODBC, plus several other popular databases, so you are sure to find something to work with your data.

If you want to, you can learn all the individual functions for connecting to and manipulating each database PHP supports, but a much smarter idea is to use `PEAR::DB`. `PEAR::DB` is an abstraction layer over the databases that PHP supports, which means you write your code once, and—with the smallest of changes—it will work on every database server.

PEAR is the script repository for PHP and contains numerous tools and prewritten solutions for common problems. `PEAR::DB` is perhaps the most popular part of the PEAR project, but it is worth checking out the PEAR site to see whether anything else catches your eye.

Introduction to `PEAR::DB`

To get basic use out of `PEAR::DB`, you need to learn how to connect to a database, run a SQL query, and work with the results. This is not a SQL tutorial, so we have assumed you are already familiar with the language. For the sake of this tutorial, we have also assumed you are working with a database called `dentists` and a table called `patients` that contains the following fields:

- **ID**—The primary key, auto-incrementing integer for storing a number unique to each patient
- **Name**—A varchar(255) field for storing a patient name
- **Age**—Integer
- **Sex**—1 for male, 2 for female
- **Occupation**—A varchar(255) field for storing a patient occupation

Also for the sake of this tutorial, we will use a database server on IP address 10.0.0.1, running MySQL, with username `ubuntu` and password `alm65z`. You will need to replace these details with your own—use `localhost` for connecting to the local server.

The first step to using `PEAR::DB` is to include the standard `PEAR::DB` file, `DB.php`. Your PHP will be configured to look inside the PEAR directory for `include()` files, so you do not need to provide any directory information.

PEAR::DB is object oriented, and you specify your connection details at the same time as you create the initial DB object. This is done using a URL-like system that specifies the database server type, username, password, server, and database name all in one. After you have specified the database server here, everything else is abstracted, meaning you only need to change the connection line to port your code to another database server.

This first script connects to our server and prints a status message (see Listing 25.6).

LISTING 25.6 Connecting to a Database Through PEAR::DB

```
<?php
  include("DB.php");
  $dsn = "mysql://ubuntu:alm65z@10.0.0.1/dentists";
  $conn = DB::connect($dsn);

  if (DB::isError($conn)) {
    echo $conn->getMessage() . "\n";
  } else {
    echo "Connected successfully!\n";
  }
?>
```

You should be able to see how the connection string breaks down. It is server name first, then a username and password separated by a colon, then an @ symbol followed by the IP address to which to connect, and then a slash and the database name. Notice how the call to connect is DB::connect(), which calls PEAR::DB directly and returns a database connection object for storage in $conn. The variable name $dsn was used for the connection details because it is a common acronym standing for data source name.

If DB::connect() successfully connects to a server, it returns a database object we can use to run SQL queries. If not, we get an error returned that we can query using functions such as getMessage(). In the previous script, we print the error message if we fail to connect, but we also just print a message if we succeed. Next, we will change that so we run an SQL query if we have a connection.

Running SQL queries is done through the query() function of our database connection, passing in the SQL we want to execute. This then returns a query result that can be used to get the data. This query result can be thought of as a multidimensional array because it has many rows of data, each with many columns of attributes. This is extracted using the fetchInto() function, which loops through the query result converting one row of data into an array that it sends back as its return value. You need to pass in two parameters to fetchInto() specifying where the data should be stored and how you want it stored. Unless you have unusual needs, specifying DB_FETCHMODE_ASSOC for the second parameter is a smart move.

Listing 25.7 shows the new script.

LISTING 25.7 Running a Query Through `PEAR::DB`

```
<?php
  include("DB.php");
  $dsn = "mysql://ubuntu:alm65z@10.0.0.1/dentists";
  $conn = DB::connect($dsn);

  if (DB::isError($conn)) {
    echo $conn->getMessage() . "\n";
  } else {
    echo "Connected successfully!\n";

    $result = $conn->query("SELECT ID, Name FROM patients;");

    while ($result->fetchInto($row, DB_FETCHMODE_ASSOC)) {
      extract($row, EXTR_PREFIX_ALL, 'pat');
      echo "$pat_ID is $pat_Name\n";
    }
  }
?>
```

The first half is identical to the previous script, with all the new action happening if we get a successful connection.

Going along with the saying "never leave to PHP what you can clean up yourself," the current script has problems. We do not clean up the query result, and we do not close the database connection. If this code were being used in a longer script that ran for several minutes, this would be a huge waste of resources. Fortunately, we can free up the memory associated with these two by calling `$result->free()` and `$conn->disconnect()`. If we add those two function calls to the end of the script, it will be complete.

Reference

Being as popular as it is, PHP gets a lot of coverage on the Internet. The best place to look for information, though, is the PHP online manual, at http://www.php.net. It is comprehensive, well written, and updated regularly:

- http://www.phpbuilder.net—A large PHP scripts and tutorials site where you can learn new techniques and also chat with other PHP developers.
- http://www.zend.com—The home page of a company founded by two of the key developers of PHP. Zend develops and sells proprietary software, including a powerful IDE and a code cache, to aid PHP developers.

- http://pear.php.net/—The home of the PEAR project contains a large collection of software you can download and try, and it has thorough documentation for it all.
- http://www.phparch.com/—There are quite a few good PHP magazines around, but *PHP Architect* probably leads the way. It posts some of its articles online for free, and its forums are good, too.

Quality books on PHP abound, and you are certainly spoiled for choice. For beginning developers, the best available is *PHP and MySQL Web Development* (Sams Publishing), ISBN 0-672-32672-8. For a concise, to-the-point book covering all aspects of PHP, check out *PHP in a Nutshell* (O'Reilly). Finally, for advanced developers, you can consult *Advanced PHP Programming* (Sams Publishing), ISBN 0-672-32561-6.

This page intentionally left blank

CHAPTER 26

C/C++ Programming Tools for Ubuntu

IN THIS CHAPTER

If you're looking to learn C or C++ programming, this part of the book isn't the right place to start—unlike Perl, Python, PHP, or even C#, it takes more than a little dabbling to produce something productive with C, so this chapter is primarily focused on the tools Ubuntu offers you as a C or C++ programmer.

Whether you're looking to compile your own code or someone else's, the GNU Compiler Collection (`gcc`) is there to help—it understands C, C++, Fortran, Pascal, and dozens of other popular languages, which means you can try your hand at whatever interests you. Ubuntu also ships with hundreds of libraries you can link to, from the GUI toolkits behind GNOME and KDE to XML parsing and game coding. Some use C, others C++, and still others offer support for both, meaning you can choose what you're most comfortable with.

Programming in C with Linux

C is the programming language most frequently associated with Unix-like operating systems such as Linux or BSD. Since the 1970s, the bulk of the Unix operating system and its applications have been written in C. Because the C language doesn't directly rely on any specific hardware architecture, Unix was one of the first portable operating systems. In other words, the majority of the code that makes up Unix doesn't know and doesn't care which computer it is actually running on. Machine-specific features are isolated in a few modules within the Unix kernel, which makes it easy for you to modify them when you are porting to different hardware architectures.

C is a *compiled* language, which means that your C source code is first analyzed by the *preprocessor* and then translated into assembly language first and then into machine instructions that are appropriate to the target CPU. An assembler then creates a binary, or *object*, file from the machine instructions. Finally, the object file is linked to any required external software support by the *linker*. A C program is stored in a text file that ends with a `.c` extension and always contains at least one routine, or function, such as `main()`, unless the file is an *include* file (with a `.h` extension, also known as a *header* file) containing shared variable definitions or other data or declarations. *Functions* are the commands that perform each step of the task that the C program was written to accomplish.

NOTE

The Linux kernel is mostly written in C, which is why Linux works with so many different CPUs. To learn more about building the Linux kernel from source, see Chapter 32, "Kernel and Module Management."

C++ is an object-oriented extension to C. Because C++ is a superset of C, C++ compilers compile C programs correctly, and it is possible to write non–object-oriented code in C++. The reverse is not true: C compilers cannot compile C++ code.

C++ extends the capabilities of C by providing the necessary features for object-oriented design and code. C++ also provides some features, such as the capability to associate functions with data structures, that do not require the use of class-based object-oriented techniques. For these reasons, the C++ language enables existing Unix programs to migrate toward the adoption of object orientation over time.

Support for C++ programming is provided by `gcc`, which you run with the name `g++` when you are compiling C++ code.

Using the C Programming Project Management Tools Provided with Ubuntu

Ubuntu is replete with tools that make your life as a C/C++ programmer easier. There are tools to create programs (editors), compile programs (`gcc`), create libraries (`ar`), control the source (Subversion, but a massive number of projects use the older CVS system), automate builds (`make`), debug programs (`gdb` and `ddd`), and determine where inefficiencies lie (`gprof`).

The following sections introduce some of the programming and project management tools included with Ubuntu. The disc included with this book contains many of these tools, which you can use to help automate software development projects. If you have some previous Unix experience, you will be familiar with most of these programs because they are traditional complements to a programmer's suite of software.

Building Programs with `make`

You use the `make` command to automatically build and install a C program, and for that use it is an easy tool. If you want to create your own automated builds, however, you need to learn the special syntax that make uses; the following sections walk you through a basic make setup.

Using Makefiles

The make command automatically builds and updates applications by using a makefile. A *makefile* is a text file that contains instructions about which options to pass on to the compiler preprocessor, the compiler, the assembler, and the linker. The makefile also specifies, among other things, which source code files have to be compiled (and the compiler command line) for a particular code module and which code modules are needed to build the program—a mechanism called *dependency checking*.

The beauty of the make command is its flexibility. You can use make with a simple makefile, or you can write complex makefiles that contain numerous macros, rules, or commands that work in a single directory or traverse your file system recursively to build programs, update your system, and even function as document management systems. The make command works with nearly any program, including text processing systems such as TeX.

You could use make to compile, build, and install a software package, using a simple command like this:

```
# make install
```

You can use the default makefile (usually called Makefile, with a capital *M*), or you can use make's -f option to specify any makefile, such as MyMakeFile, like this:

```
# make -f MyMakeFile
```

Other options might be available, depending on the contents of your makefile.

Using Macros and Makefile Targets

Using make with macros can make a program portable. Macros allow users of other operating systems to easily configure a program build by specifying local values, such as the names and locations, or *pathnames*, of any required software tools. In the following example, macros define the name of the compiler (CC), the installer program (INS), where the program should be installed (INSDIR), where the linker should look for required libraries (LIBDIR), the names of required libraries (LIBS), a source code file (SRC), the intermediate object code file (OBS), and the name of the final program (PROG):

```
# a sample makefile for a skeleton program
CC= gcc
INS= install
INSDIR = /usr/local/bin
LIBDIR= -L/usr/X11R6/lib
LIBS= -lXm -lSM -lICE -lXt -lX11
SRC= skel.c
OBJS= skel.o
PROG= skel
```

```
skel:  ${OBJS}
        ${CC} -o ${PROG} ${SRC} ${LIBDIR} ${LIBS}

install: ${PROG}
        ${INS} -g root -o root ${PROG} ${INSDIR}
```

NOTE

The indented lines in the previous example are indented with tabs, not spaces. This is important to remember! It is difficult for a person to see the difference, but `make` can tell. If `make` reports confusing errors when you first start building programs under Linux, check your project's makefile for the use of tabs and other proper formatting.

Using the makefile from the preceding example, you can build a program like this:

```
# make
```

To build a specified component of a makefile, you can use a target definition on the command line. To build just the program, you use `make` with the `skel` target, like this:

```
# make skel
```

If you make any changes to any element of a target object, such as a source code file, `make` rebuilds the target automatically. This feature is part of the convenience of using `make` to manage a development project. To build and install a program in one step, you can specify the target of `install` like this:

```
# make install
```

Larger software projects might have a number of traditional targets in the makefile, such as the following:

- `test`—To run specific tests on the final software
- `man`—To process an include or a `troff` document with the `man` macros
- `clean`—To delete any remaining object files
- `archive`—To clean up, archive, and compress the entire source code tree
- `bugreport`—To automatically collect and then mail a copy of the build or error logs

Large applications can require hundreds of source code files. Compiling and linking these applications can be a complex and error-prone task. The `make` utility helps you organize the process of building the executable form of a complex application from many source files.

Using the `autoconf` Utility to Configure Code

The `make` command is only one of several programming automation utilities included with Ubuntu. There are others, such as `pmake` (which causes a parallel make), `imake` (which is a dependency-driven makefile generator that is used for building X11 clients), `automake`, and one of the newer tools, `autoconf`, which builds shell scripts that can be used to configure program source code packages.

Building many software packages for Linux that are distributed in source form requires the use of GNU's `autoconf` utility. This program builds an executable shell script named `configure` that, when executed, automatically examines and tailors a client's build from source according to software resources, or *dependencies* (such as programming tools, libraries, and associated utilities) that are installed on the target host (your Linux system).

Many Linux commands and graphical clients for X downloaded in source code form include `configure` scripts. To configure the source package, build the software, and then install the new program, the `root` user might use the script like this (after uncompressing the source and navigating into the resulting build directory):

```
$ ./configure ; make ; sudo make install
```

The `autoconf` program uses a file named `configure.in` that contains a basic *ruleset*, or set of macros. The `configure.in` file is created with the `autoscan` command. Building a properly executing `configure` script also requires a template for the makefile, named `Makefile.in`. Although creating the dependency-checking `configure` script can be done manually, you can easily overcome any complex dependencies by using a graphical project development tool such as KDE's KDevelop or GNOME's Glade. (See the "Graphical Development Tools" section, later in this chapter, for more information.)

Managing Software Projects with Subversion

Although `make` can be used to manage a software project, larger software projects require document management, source code controls, security, and revision tracking as the source code goes through a series of changes during its development. Subversion provides source code version control utilities for this kind of large software project management.

The Subversion system is used to track changes to multiple versions of files, and it can be used to backtrack or branch off versions of documents inside the scope of a project. It can also be used to prevent or resolve conflicting entries or changes made to source code files by multiple developers. Source code control with Subversion requires the use of at least the following five command options on the `svn` command line:

- `checkout`—Checks out revisions
- `update`—Updates your sources with changes made by other developers
- `add`—Adds new files to the repository
- `delete`—Eliminates files from the repository
- `commit`—Publishes changes to other repository developers

Note that some of these commands require you to use additional fields, such as the names of files. With the `commit` command, you should always try to pass the `-m` parameter (lets you provide a message describing the change) followed by some information about your changes. For example:

```
svn commit -m "This fixes bug 204982."
```

One of the most impressive features of Subversion is its ability to work offline—any local Subversion checkout automatically has a `.svn` directory hidden in there, which contains copies of all checked out files. Thanks to this, you can check your current files against the ones you checked out without having to contact the server—it all runs locally.

Debugging Tools

Debugging is both a science and an art. Sometimes, the simplest tool—the code listing—is the best debugging tool. At other times, however, you need to use other debugging tools. Three of these tools are `splint`, `gprof`, and `gdb`.

Using `splint` to Check Source Code

The `splint` command is similar to the traditional Unix `lint` command: It statically examines source code for possible problems, and it also has many additional features. Even if your C code meets the standards for C and compiles cleanly, it might still contain errors. `splint` performs many types of checks and can provide extensive error information. For example, this simple program might compile cleanly and even run:

```
$ gcc -o tux tux.c
$ ./tux
```

But the `splint` command might point out some serious problems with the source:

```
$ splint tux.c
Splint 3.1.1 --- 17 Feb 2004

tux.c: (in function main)
tux.c:2:19: Return value (type int) ignored: putchar(t[++j] -...
  Result returned by function call is not used. If this is intended, can cast
  result to (void) to eliminate message. (Use -retvalint to inhibit warning)
Finished checking --- 1 code warning
```

You can use the `splint` command's `-strict` option, like this, to get a more verbose report:

```
$ splint -strict tux.c
```

GCC also supports diagnostics through the use of extensive warnings (through the `-Wall` and `-pedantic` options):

```
$ gcc -Wall tux.c
tux.c:1: warning: return type defaults to `int'
tux.c: In function `main':
tux.c:2: warning: implicit declaration of function `putchar'
```

Using `gprof` to Track Function Time

You use the `gprof` (profile) command to study how a program is spending its time. If a program is compiled and linked with `-p` as a flag, a `mon.out` file is created when it executes, with data on how often each function is called and how much time is spent in each function. `gprof` parses and displays this data. An analysis of the output generated by `gprof` helps you determine where performance bottlenecks occur. Using an optimizing compiler can speed up a program, but taking the time to use `gprof`'s analysis and revising bottleneck functions significantly improves program performance.

Doing Symbolic Debugging with `gdb`

The `gdb` tool is a symbolic debugger. When a program is compiled with the `-g` flag, the symbol tables are retained and a symbolic debugger can be used to track program bugs. The basic technique is to invoke `gdb` after a *core dump* (a file containing a snapshot of the memory used by a program that has crashed) and get a stack trace. The stack trace indicates the source line where the core dump occurred and the functions that were called to reach that line. Often, this is enough to identify a problem. It isn't the limit of `gdb`, though.

`gdb` also provides an environment for debugging programs interactively. Invoking `gdb` with a program enables you to set breakpoints, examine the values of variables, and monitor variables. If you suspect a problem near a line of code, you can set a breakpoint at that line and run `gdb`. When the line is reached, execution is interrupted. You can check variable values, examine the stack trace, and observe the program's environment. You can single-step through the program to check values. You can resume execution at any point. By using breakpoints, you can discover many bugs in code.

A graphical X Window interface to `gdb` is called the Data Display Debugger, or `ddd`.

Using the GNU C Compiler

If you elected to install the development tools package when you installed Ubuntu (or perhaps later on, using `synaptic`), you should have the GNU C compiler (`gcc`). Many different options are available for the GNU C compiler, and many of them are similar to those of the C and C++ compilers that are available on other Unix systems. Look at the `man` page or information file for `gcc` for a full list of options and descriptions.

When you build a C program using `gcc`, the compilation process takes place in several steps:

1. First, the C preprocessor parses the file. To do so, it sequentially reads the lines, includes header files, and performs macro replacement.
2. The compiler parses the modified code to determine whether the correct syntax is used. In the process, it builds a symbol table and creates an intermediate object format. Most symbols have specific memory addresses assigned, although symbols defined in other modules, such as external variables, do not.
3. The last compilation stage, linking, ties together different files and libraries and then links the files by resolving the symbols that had not previously been resolved.

NOTE

Most C programs compile with a C++ compiler if you follow strict ANSI rules. For example, you can compile the standard `hello.c` program (everyone's first program) with the GNU C++ compiler. Typically, you name the file something like `hello.cc`, `hello.C`, `hello.c++`, or `hello.cxx`. The GNU C++ compiler accepts any of these names.

Graphical Development Tools

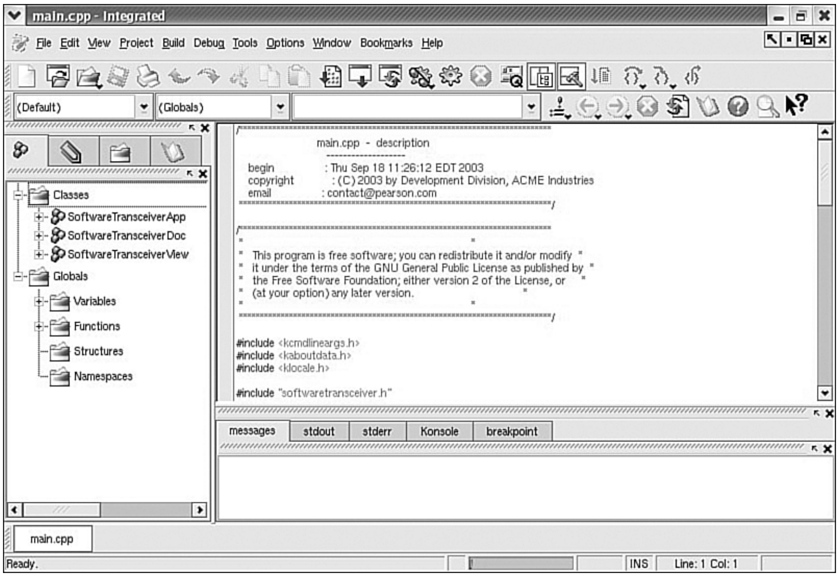
Ubuntu includes a number of graphical prototyping and development environments for use during X sessions. If you want to build client software for KDE or GNOME, you might find the KDevelop, Qt Designer, and Glade programs extremely helpful. You can use each of these programs to build graphical frameworks for interactive windowing clients, and you can use each of them to automatically generate the necessary skeleton of code needed to support a custom interface for your program.

Using the KDevelop Client

You can launch the KDevelop client (shown in Figure 26.1) from the application's menu, or from the command line of a terminal window, like this:

```
$ kdevelop &
```

After you press Enter, the KDevelop Setup Wizard runs, and you are taken through several short wizard dialogs that set up and ensure a stable build environment. You must then run `kdevelop` again (either from the command line or by clicking its menu item under the desktop panel's Programming menu). You will then see the main KDevelop window and can start your project by selecting KDevelop's Project menu and clicking the New menu item.



```
/***************************************************************************
                     main.cpp  -  description
                        -------------------
    begin            : Thu Sep 18 11:26:12 EDT 2003
    copyright          : (C) 2003 by Development Division, ACME Industries
    email            : contact@pearson.com
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#include <kcmdlineargs.h>
#include <kaboutdata.h>
#include <klocale.h>

#include "softwaretransceiver.h"
```

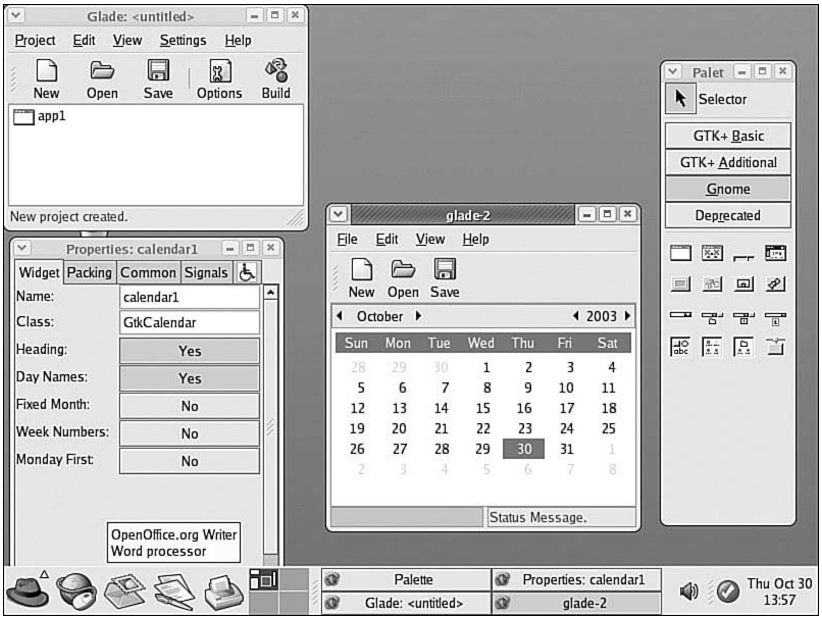
FIGURE 26.1 KDE's KDevelop is a rapid prototyping and client-building tool for use with Linux.

You can begin building your project by stepping through the wizard dialogs. When you click the Create button, KDevelop automatically generates all the files that are normally found in a KDE client source directory (including the `configure` script, which checks dependencies and builds the client's makefile). To test your client, you can either first click the Build menu's Make menu item (or press F8) or just click the Execute menu item (or press F9), and the client is built automatically. You can use KDevelop to create KDE clients, plug-ins for the `Konqueror` browser, KDE `kicker` panel applets, KDE desktop themes, Qt library–based clients, and even programs for GNOME.

The Glade Client for Developing in GNOME

If you prefer to use GNOME and its development tools, the Glade GTK+ GUI builder can help you save time and effort when building a basic skeleton for a program. You launch Glade from the desktop panel's Programming menu..

When you launch Glade, a directory named `Projects` is created in your home directory, and you see a main window, along with two floating Palette and Properties windows (see Figure 26.2, which shows a basic GNOME client with a calendar widget added to its main window). You can use Glade's File menu to save the blank project and then start building your client by clicking and adding user interface elements from the Palette window. For example, you can first click the Palette window's Gnome button and then click to create your new client's main window. A window with a menu and a toolbar appears—the basic framework for a new GNOME client!



Name: calendar1
Class: GtkCalendar
Heading: Yes
Day Names: Yes
Fixed Month: No
Week Numbers: No
Monday First: No

FIGURE 26.2 You can use the GNOME Glade client to build and preview a graphical interface for a custom GNOME program.

Related Ubuntu and Linux Commands

You will use many of these commands when programming in C and C++ for Linux:

- `ar`—The GNU archive development tool
- `as`—The GNU assembler
- `autoconf`—The GNU configuration script generator
- `cervisia`—A KDE client that provides a graphical interface to a CVS project
- `cvs`—A project revision control system
- `designer`—Trolltech's graphical prototyping tool for use with Qt libraries and X
- `gcc`—The GNU C/C++ compiler system
- `gdb`—The GNU interactive debugger
- `glade-2`—The GNOME graphical development environment for building GTK+ clients
- `gprof`—The GNU program profiler
- `kdevelop`—The KDE C/C++ graphical development environment for building KDE, GNOME, or terminal clients
- `make`—A GNU project management command
- `patch`—Larry Wall's source patching utility
- `splint`—The C source file checker
- `svn`—The Subversion revision control system

Reference

If you are interested in learning more about C and C++, you should look for the following books:

- *Sams Teach Yourself C in 21 Days*, by Peter Aitken and Bradley Jones, Sams Publishing
- *Sams Teach Yourself C++ for Linux in 21 Days*, by Jesse Liberty and David B. Horvath, Sams Publishing
- *C How to Program* and *C++ How to Program*, both by Harvey M. Deitel and Paul J. Deitel, Deitel Associates
- *The C Programming Language*, by Brian W. Kernighan and Dennis M. Ritchie, Prentice Hall
- *The Annotated C++ Reference Manual*, by Margaret A. Ellis and Bjarne Stroustrup, ANSI Base Document
- *Programming in ANSI C*, by Stephen G. Kochan, Sams Publishing

There are also a number of excellent websites providing more information:

- http://gcc.gnu.org/java/compile.html—More information about GCC's Java support.
- http://www.gnu.org/software/autoconf/autoconf.html—More information about the GNU Project's `autoconf` utility and how to build portable software projects.
- http://www.trolltech.com/products/qt/tools.html—Trolltech's page for Qt Designer and a number of programming automation tools (including translators) that you can use with Ubuntu.
- http://glade.gnome.org—Home page for the Glade GNOME developer's tool.
- http://www.kdevelop.org—Site that hosts the KDevelop Project's latest versions of the KDE graphical development environment, KDevelop.

This page intentionally left blank

CHAPTER 27
Mono

IN THIS CHAPTER

Although Microsoft intended it for Windows, Microsoft's .NET platform has grown to encompass many other operating systems. No, this isn't a rare sign of Microsoft letting customers choose which OS is best for them—instead, the spread of .NET is because of the Mono project, which is a free re-implementation of .NET available under the a GPL license.

Because of the potential for patent complications, it took most distros a long time to incorporate Mono, but it's here now and works just fine. What's more, Mono supports both C# and Visual Basic .NET, as well as the complete .NET 1.0 and 1.1 frameworks (and much of the 2.0 framework too), making it quick to learn and productive to use.

Why Use Mono?

Linux already has numerous programming languages available to it, so why bother with Mono and .NET? Here are my top five reasons:

- .NET is "compile once, run anywhere"; that means you can compile your code on Linux and run it on Windows, or the reverse.
- Mono supports C#, which is a C-like language with many improvements to help make it object-oriented and easier to use.
- .NET includes automatic garbage collection to remove the possibility of memory leaks.
- .NET uses comes with built-in security checks to ensure that buffer overflows and many types of exploits are a thing of the past.

- Mono uses a high-performance just-in-time compiler to optimize your code for the platform on which it's running. This lets you compile it on a 32-bit machine, then run it on a 64-bit machine and have the code dynamically re-compiled for maximum 64-bit performance.

At this point, Mono is probably starting to sound like Java, and indeed it shares several properties with it. However, Mono has the following improvements:

- The C# language corrects many of the irritations in Java, while keeping its garbage collection.
- .NET is designed to let you compile multiple languages down to the same bytecode, including C#, Visual Basic .NET, and many others. The Java VM is primarily restricted to the Java language.
- Mono even has a special project (known as "IKVM") that compiles Java source code down to .NET code that can be run on Mono.
- Mono is completely open source!

Whether you're looking to create command-line programs, graphical user interface apps, or even web pages, Mono has all the power and functionality you need.

Mono on the Command Line

Mono should already be installed on your system, however it is installed only for end users rather than for developers—you need to install a few more packages to make it usable for programming. Start up the Synaptic Package Manager, and make sure the following packages are selected:

- mono
- mono-debugger
- mono-devel
- monodevelop
- monodoc
- beagle
- beagle-dev
- mono-gmcs

That gives you the basics to do Mono development, plus a few extra bits and pieces if you want to branch out a bit.

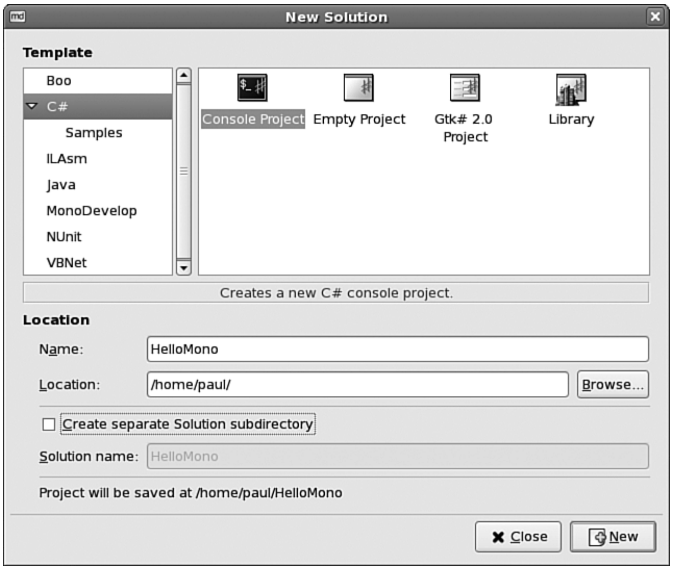
If you want to do some exciting things with Mono, lots of Mono-enabled libraries are available. Try going to the Search view and search for "sharp" to bring up the list of .NET-enabled libraries that you can use with Mono—the suffix is used because C# is the most popular .NET language. In this list you'll see things such as gnome-sharp2 and gtk-sharp2—we recommend you at least install the gtk-sharp2 libraries as these are used to create graphical user interfaces for Mono.

But for now, let's get you up and running with Mono on the command line. Mono is split into two distinct parts: the compiler and the interpreter. The compiler turns your source code into an executable, and is called gmcs. The interpreter actually runs your code as a working program, and is just called Mono. You should by now have installed MonoDevelop, so go the Applications menu, choose Programming, then MonoDevelop to start it up.

TIP

You don't have to use MonoDevelop to write your code, but it helps—syntax highlighting, code completion, and drag-and-drop GUI designers are just a few of its features.

When MonoDevelop has loaded, go to the File menu and choose New Project. From the left of the window that appears, choose C#, then Console Project. Give it a name and choose a location to save it—all being well you should see something similar to Figure 27.1. When you're ready, click New to have MonoDevelop generate your project for you.



Creates a new C# console project.

Location

Name: HelloMono

Location: /home/paul/ Browse...

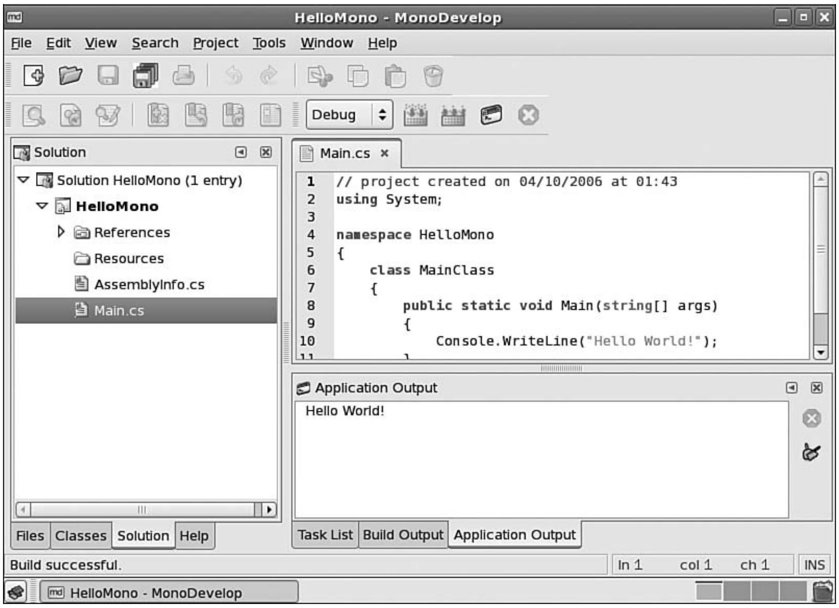
☐ Create separate Solution subdirectory

Solution name: HelloMono

Project will be saved at /home/paul/HelloMono

FIGURE 27.1 MonoDevelop ships with a number of templates to get you started, including one for a quick Console Project.

The default Console Project template creates a program that prints a simple message to the command line: the oh-so-traditional "Hello World!" Change it to something more insightful if you want, then press F5 to build and run the project. Just following the code view is a set of tabs where debug output is printed. One of those tabs, Application Output, becomes selected when the program runs, and you'll see "Hello World!" (or the message you chose) printed there—not bad given that you haven't written any code yet! You can see how this should look in Figure 27.2.


HelloMono - MonoDevelop
File Edit View Search Project Tools Window Help
Debug
Solution
Solution HelloMono (1 entry)
HelloMono
References
Resources
AssemblyInfo.cs
Main.cs
Files Classes Solution Help
Application Output
Task List Build Output Application Output
ln 1 col 1 ch 1 INS

```
// project created on 04/10/2006 at 01:43
using System;

namespace HelloMono
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
```

```
Hello World!
```

```
Build successful.
```

FIGURE 27.2 Your console template prints a message to the Application Output window in the bottom part of the window.

The Structure of a C# Program

As you can guess from its name, C# draws very heavily on C and C++ for its syntax, but it borrows several ideas from Java too. C# is object-oriented, which means your program is defined as a class, which is then instantiated through the `Main()` method call. To be able to draw on some of the .NET framework's many libraries, you need to add `using` statements at the top of your files—by default there's just `using System;` that enables you to get access to the console to write your message.

If you've come from C or C++, you'll notice that there are no header files in C#: your class definition and its implementation are all in the same file. You might also have noticed that the `Main()` method accepts the parameter `"string[] args"`, which is C#-speak for "an array of strings." C never had a native "string" data type, whereas C++ acquired it rather late in the game, and so both languages tend to use the antiquated char* data type to point to a string of characters. In C#, "string" is a data type all its own, and comes with built-in functionality such as the capability to replace substrings, the capability to trim off whitespace, and the capability to convert itself to upper- or lowercase if you want it to. Strings are also Unicode friendly out of the box in .NET, so that's one fewer thing for you to worry about.

The final thing you might have noticed—at least, if you had looked in the directory where MonoDevelop placed your compiled program (usually /path/to/your/project/bin/Debug)—is that Mono uses the Windows-like .exe file extension for its programs. because Mono aims to be 100% compatible with Microsoft .NET, which means you can take your Hello World program and run it unmodified on a Windows machine and have the same message printed out.

Printing Out the Parameters

We're going to expand your little program by having it print out all the parameters passed to it, one per line. In C# this is—rather sickeningly—just one line of code. Add this just after the existing Console.WriteLine() line in your program:

```
foreach (string arg in args) Console.WriteLine(arg);
```

The foreach keyword is a special kind of loop designed to iterate over an array of finite size. The for array exists in C#, and lets you loop a certain number of times; the while array exists too, and lets you loop continuously until you tell it to break. But the foreach loop is designed to loop over arrays that have a specific number of values, but you don't know how many values that will be. You get each value as a variable of any type you want—the preceding code says string arg in args, which means "for each array element in args, give it to me as the variable arg of type string. Of course, args is already an array of strings, so no datatype conversion will actually take place here—but it could if you wanted to convert classes or do anything of the like.

After you have each individual argument, call WriteLine() to print it out. This works whether there's one argument or one hundred arguments—or even if there are no arguments at all (in which case the loop doesn't execute at all).

Creating Your Own Variables

As you saw in the parameter list for Main() and the arg variable in your foreach loop, C# insists that each variable has a distinct data type. You can choose from quite a variety: Boolean for true/false values; string for text; int for numbers, float for floating-point numbers; and so on. If you want to be very specific, you can use int32 for a 32-bit integer (covering from –2147483648 to 2147483648) or int64 for a 64-bit integer (covering even larger numbers). But on the whole you can just use int and leave C# to work it out itself.

So now you can modify your program to accept two parameters, add them together as numbers, then print the result. This gives you a chance to see variable definitions, conversion, and mathematics, all in one. Edit the Main() method to look like this:

```
public static void Main (string[] args)
{
   int num1 = Convert.ToInt32(args[0]);
   int num2 = Convert.ToInt32(args[1]);
   Console.WriteLine("Sum of two parameters is: " + (num1 + num2)");
}
```

As you can see, each variable needs to be declared with a type (`int`) and a name (`num1` and `num2`), so that C# knows how to handle them. Your `args` array contains strings, so you need to explicitly convert the strings to integers with the `Convert.ToInt32()` method. Finally, the actual addition of the two strings is done at the end of the method, while they are being printed out. Note how C# is clever enough to have `integer + integer` be added together (in the case of num + num2), whereas `string + integer` attaches the integer to the end of the string (in the case of `"Sum of two parameters is:" + the result of num1 + num2`). This isn't by accident: C# tries to convert data types cleverly, and warns you only if it can't convert a data type without losing some data. For example, if you try to treat a 64-bit integer as a 32-bit integer, it warns you because you might be throwing a lot of data away.

Adding Some Error Checking

Right now your program crashes in a nasty way if users don't provide at least two parameters. The reason for this is that we use `arg[0]` and `arg[1]` (the first and second parameters passed to your program) without even checking whether *any* parameters were passed in. This is easily solved: `args` is an array, and arrays can reveal their size. If the size doesn't match what you expect, you can bail out.

Add this code at the start of the `Main()` method:

```
if (args.Length != 2) {
   Console.WriteLine("You must provide exactly two parameters!");
   return;
}
```

The new piece of code in there is `return`, which is a C# keyword that forces it to exit the current method. As `Main()` is the only method being called, this has the effect of terminating the program because the user didn't supply two parameters.

Using the `Length` property of `args`, it is now possible for you to write your own `Main()` method that does different things, depending on how many parameters are provided. To do this properly, you need to use the `else` statement and nest multiple `if` statements like this:

```
if (args.Length == 2) {
   /// whatever…
} else if (args.Length == 3) {
   /// something else
} else if (args.Length == 4) {
   /// even more
} else {
   /// only executed if none of the others are
}
```

Building on Mono's libraries

Ubuntu ships with several Mono-built programs, including Tomboy and Beagle. It also comes with a fair collection of .NET-enabled libraries, some of which you probably already installed earlier. The nice thing about Mono is that it lets you build on these libraries really easily: You just import them with a `using` statement, then get started.

To demonstrate how easy it is to build more complicated Mono applications, we're going to produce two: one using Beagle, the super-fast file indexer, and one using Gtk#, the GUI toolkit that's fast becoming the standard for Gnome development. Each has its own API that takes some time to master fully, but you can get started with them in minutes.

Searching with Beagle

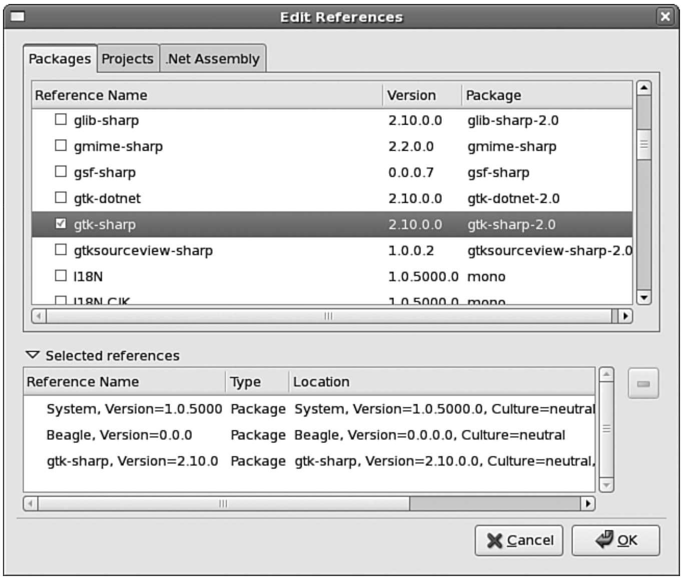
Beagle is the de facto Linux search tool for Gnome, and is also used by several KDE-based programs. It works by scanning your computer in the background, then monitoring for file system changes so that its data always stays up to date. However, the magic is that it indexes data cleverly—if you tag your images, it reads those tags. If you have album and artist data in your MP3s, it reads that data too. It also reads your emails, your instant messenger conversations, your web browser history, and much more—and provides all this data in one place, so if you search for "firefox" you'll find the application itself, all the times you've mentioned Firefox in your emails, and so on.

In MonoDevelop, go to File, New Project, select C#, then choose Console Project. Give it the name BeagleTest, and tell MonoDevelop not to create a separate directory for the solution, and also that you don't want Gtk# support or Packaging Integration. You'll be back at the default Hello World program, but you're going to change that. First, you need to tell Mono that you want to use Beagle and Gtk# (by hand, rather than using its project wizard). No, you're not going to create a GUI for your search, but you do want to take advantage of Gtk#'s idle loop system—we'll explain why soon.

To add references to these two libraries, right-click on the word References in the left pane (just above Resources) and select Edit References. A new window appears (shown in Figure 27.3), and from that you should make sure Beagle and gtk-sharp are selected. Now click OK, and the References group on the left should expand so that you can see you have Beagle, gtk-sharp, and System (the last one is the default reference for .NET programs).

Now it's time to write the code. At the top of the `Main.cs` file (your main code file), you need to edit the `"using"` statements to look like this:

```
using System;
using System.Collections;
using Beagle;
using Gtk;
```



Reference Name	Version	Package
☐ glib-sharp	2.10.0.0	glib-sharp-2.0
☐ gmime-sharp	2.2.0.0	gmime-sharp
☐ gsf-sharp	0.0.0.7	gsf-sharp
☐ gtk-dotnet	2.10.0.0	gtk-dotnet-2.0
☑ gtk-sharp	2.10.0.0	gtk-sharp-2.0
☐ gtksourceview-sharp	1.0.0.2	gtksourceview-sharp-2.0
☐ I18N	1.0.5000.0	mono
☐ I18N.CJK	1.0.5000.0	mono

Reference Name	Type	Location
System, Version=1.0.5000	Package	System, Version=1.0.5000.0, Culture=neutral
Beagle, Version=0.0.0	Package	Beagle, Version=0.0.0.0, Culture=neutral
gtk-sharp, Version=2.10.0	Package	gtk-sharp, Version=2.10.0.0, Culture=neutral,

FIGURE 27.3 You need to tell Mono exactly which resource libraries you want to import for your program.

The `BeagleTest` namespace and `MainClass` class aren't changing, but you do need to edit the `Main()` method so that you can run your Beagle query. Here's how it should look, with C# comments (//, as in C++) sprinkled throughout to help you understand what's going on:

```
public static void Main(string[] args)
{
    Application.Init();

    // "Query" is the main Beagle search type.
    //It does lots of magic for you - you just need to provide it with a search
term and tell it where to search
    Query q = new Query();

    // these two are callback functions.
    //What you're saying is, when a hit is returned
    // (that is, when Beagle finds something, it should
    // run your OnHitsAdded() method. That code isn't written
    // yet, but you'll get there soon.
    q.HitsAddedEvent += OnHitsAdded;
    q.FinishedEvent += OnFinished;

    // these two tell Beagle where to search
    q.AddDomain(QueryDomain.Neighborhood);
    q.AddDomain(QueryDomain.Global);
    // finally, you tell Beagle to search for the first word
    // provided to your command (args[0]), then ask it
    // to asynchronously run its search. That is, it runs
    // in the background and lets your program continue running
    q.AddText(args[0]);
    q.SendAsync();

    // tell Gtk# to run
    Application.Run();
}
```

The only thing I haven't explained in there is the Gtk# stuff, but you might already have guessed why it's needed. The problem is this: Beagle runs its search asynchronously, which means that it returns control to your program straight away. Without the `Application.Run()` call, the `SendAsync()` method is the last thing your program does, which meant that it terminates itself before Beagle actually has chance to return any data. So, Gtk# serves as an idle loop: when you call `Run()`, Gtk# makes sure your program carries on running until you tell it to quit, giving Beagle enough time to return its results.

Now, let's take a look at the `OnHitsAdded` and `OnFinished` methods, called whenever Beagle finds something to return and when it's finished searching, respectively:

```
static void OnHitsAdded (HitsAddedResponse response)
{
    // sometimes Beagle can return multiple hits (files)
    // in each response, so you need to go through each
    // one and print it out line by line
    foreach(Hit hit in response.Hits)
    {
        // the Uri of hits is its location, which might
        // be on the web, on your filesystem, or somewhere else
        Console.WriteLine("Hit: " + hit.Uri);
    }
}

static void OnFinished(FinishedResponse response)
{
    // the application is done, we can tell Gtk# to quit now
    Application.Quit();
}
```

When you're done, press F8 to compile your program. If you encounter any errors, you have typed something incorrectly, so check carefully against the preceding text. Now open a terminal, change to the directory where you created your project, then look inside there for the bin/Debug subdirectory. All being well, you should find the `BeagleTest.exe` file in there, which you can run like this:

```
mono BeagleTest.exe hello
```

If you get a long string of errors when you run your program, try running this command first: `export MONO_PATH=/usr/lib/beagle`. That tells Mono where to look for the Beagle library, which is probably your problem.

Creating a GUI with Gtk#

Gtk# was included with Gnome by default for the first time in Gnome 2.16, but it had been used for a couple of years prior to that and so was already mature. MonoDevelop comes with its own GUI designer called Stetic, which lets you drag and drop GUI elements onto your windows to design them.

To get started, go to File, New Project in MonoDevelop, choose C#, then Gtk# 2.0 project. Call it GtkTest, and deselect the box asking MonoDevelop to make a separate directory for the solution. You'll find that Main.cs contains a little more code this time because it needs to create and run the Gtk# application. However, the actual code to create your GUI lives in User Interface in the left pane. If you open that group, you'll see MainWindow, which, when double-clicked, brings up MonoDevelop's GUI designer.

There isn't space for me to devote much time to GUI creation, but it's very easy for you to drag and drop the different window widgets onto your form to see what properties they have. The widgets are all listed on the top right of the GUI designer, with widget properties on the bottom-right.

For now, drag a button widget onto your form. It automatically takes up all the space on your window. If you don't want this to happen, try placing one of the containers down first, then putting your button in there. For example, if you want a menu bar at the top, then a calendar, then a status bar, you ought to drop the VBox pane onto the window first, then drop each of those widgets into the separate parts of the VPane, as shown in Figure 27.4.

Your button will have the text "button1" by default, so click on it to select it, then look in the properties pane for Label. It might be hidden away in the Button Properties group, so you'll need to make sure that's open. Change the label to "Hello." Just at the top of the properties pane is a tab saying Properties (where you are right now), and another saying Signals. Signals are the events that happen to your widgets, such as the mouse moving over them, someone typing, or, of interest to us, when your button has been clicked. Look inside the Button Signals group for Clicked and double-click on it. MonoDevelop automatically switches you to the code view, with a pre-created method to handle button clicks.

Type this code into the method:

```
button1.Label = "World!";
```

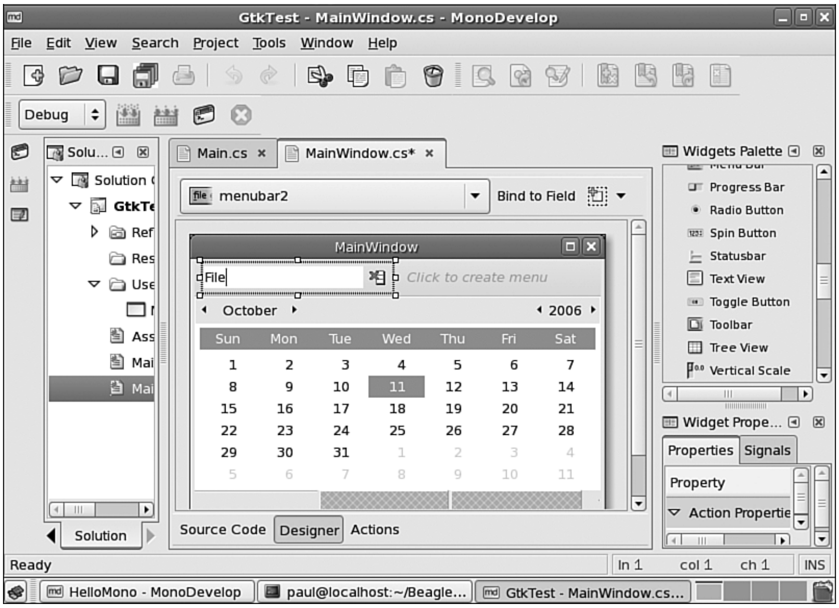


FIGURE 27.4 Using a VPane lets you space your widgets neatly on your form. Gtk# automatically handles window and widget resizing for you.

You need to make one last change before you try compiling. MonoDevelop doesn't automatically give you variables to work with each item in your window—you need to ask for them explicitly. Beneath the code window you will see a button saying Designer—click that to get back to your GUI designer window. Now click the button you created, then click the button marked Bind to Field. This edits your code to create the `button1` variable (if you click Source Code you see the variable near the top). Now press F5 to compile and run, and try clicking the button!

Reference

- http://www.mono-project.com/—The homepage of the Mono project is packed with information to help you get started. You can also download new Mono versions from here, if there's something you desperately need.
- http://www.monodevelop.com/—The MonoDevelop project has its own site, which is the best place to look for updates.
- http://www.icsharpcode.net/OpenSource/SD/—MonoDevelop started life as a port of SharpDevelop. If you happen to dual-boot on Windows, this might prove very useful to you.

- http://msdn.microsoft.com/vcsharp/—We don't print many Microsoft URLs in this book, but this one is important: It's the homepage of their C# project, which can be considered the spiritual home of C# itself.
- Jesse Liberty's *Programming C#* (O'Reilly, ISBN 0-596-00699-3) is compact, it's comprehensive, and it's competitively priced.
- If you're very short on time and want the maximum detail (admittedly, with rather limited readability), you should try *The C# Programming Language*, which was co-authored by the creator of C#, Anders Hejlsberg (Addison-Wesley, ISBN: 0-321-33443-4).
- For a more general book on the .NET framework and all the features it provides, you might find *.NET Framework Essentials* (O'Reilly, ISBN: 0-596-00505-9) useful.

PART VI
Ubuntu Housekeeping

IN THIS PART

This page intentionally left blank

CHAPTER 28

Securing Your Machines

IN THIS CHAPTER

No home computer with a connection to the Internet is 100% safe. If this information does not concern you, it should! Although there is no way to stop a serious cracker who is intent on getting into your computer or network, there are ways to make it harder for him and to warn you when he does.

In this chapter, we discuss all aspects of securing your Linux machines. You might have wondered why we did not spread this information around the book wherever it was appropriate, but the reason is simple: If you ever have a security problem with Linux, you know you can turn to this page and start reading without having to search or try to remember where you saw a tip. Everything you need is here in this one chapter, and we strongly advise you read it from start to finish.

Built-In Protection in the Kernel

A number of networking and low-level protective services are built in to the Linux kernel. These services can be enabled, disabled, or displayed using the `sysctl` command, or by echoing a value (usually a `1` or a `0` to turn a service on or off) to a kernel process file under the `/proc` directory.

Understanding Computer Attacks

There are many ways in which computer attacks can be divided, but perhaps the easiest is *internal*, which are computer attacks done by someone with access to a computer on the local network, and *external*, which are attacks by someone with access to a computer through the Internet. This might sound like a trivial separation to make, but it is actually important: Unless you routinely hire talented computer hackers or allow visitors to plug computers into your network, the worst internal attack you are likely encounter is from a disgruntled employee.

Hacker Versus Cracker

In earlier days, there was a distinction made between the words *hacker* and *cracker*. A hacker was someone who used technology to innovate in new or unusual ways, whereas a cracker was someone who used technology to attack another's computers and cause harm.

This distinction was lost on the general public, so the term *hacker* has now come to mean the same as *cracker*. This book follows general usage, so a *hacker* is a malicious person using his computer to cause problems for others.

Although you should never ignore the internal threat, you should arguably be more concerned with the outside world. The big bad Internet is a security vortex. Machines connected directly to the outside world can be attacked by people across the world, and invariably are, even only a few minutes after having been connected.

This situation is not a result of malicious users lying in wait for your IP address to do something interesting. Instead, canny virus writers have created worms that exploit a vulnerability, take control of a machine, and then spread it to other machines around them. As a result, most attacks today are the result of these autohacking tools; there are only a handful of true hackers around, and, to be frank, if one of these ever actually targets you seriously, it will take a mammoth effort to repel him regardless of which operating system you run.

Autohacking scripts also come in another flavor: prewritten code that exploits a vulnerability and gives its users special privileges on the hacked machine. These scripts are rarely used by their creators; instead, they are posted online and downloaded by wannabe hackers, who then use them to attack vulnerable machines.

So, the external category is itself made up of worms, serious day job hackers, and wannabe hackers (usually called *script kiddies*). Combined they will assault your Internet-facing servers, and it is your job to make sure your boxes stay up, happily ignoring the firefight around them.

On the internal front, things are somewhat more difficult. Users who sit inside your firewall are already past your primary source of defense and, worse, they might even have physical access to your machines.

Regardless of the source of the attack, there is a five-step checklist you can follow to secure your box:

1. Assess your vulnerability. Decide which machines can be attacked, which services they are running, and who has access to them.
2. Configure the server for maximum security. Only install what you need, only run what you must, and configure a local firewall.

3. Secure physical access to the server.
4. Create worst-case-scenario policies.
5. Keep up-to-date with security news.

Each of these is covered in the following sections, and each is as important as the others.

Assessing Your Vulnerability

It is a common mistake for people to assume that switching on a firewall makes them safe. This is not the case and, in fact, has never been the case. Each system has distinct security needs, and taking the time to customize its security layout will give you maximum security and the best performance.

The following list summarizes the most common mistakes:

- **Installing every package**—Do you plan to use the machine as a DNS server? If not, why have BIND installed? Go through Synaptic and ensure that you have only the software you need.
- **Enabling unused services**—Do you want to administer the machine remotely? Do you want people to upload files? If not, turn off SSH and FTP because they just add needless attack vectors. This goes for many other services.
- **Disabling the local firewall on the grounds that you already have a firewall at the perimeter**—In security, depth is crucial: The more layers someone has to hack through, the higher the likelihood she will give up or get caught.
- **Letting your machine give out more information than it needs to**—Many machines are configured to give out software names and version numbers by default, which is just giving hackers a helping hand.
- **Placing your server in an unlocked room**—If so, you might as well just turn it off now and save the worry. The exception to this is if all the employees at your company are happy and trustworthy. But why take the risk?
- **Plugging your machine into a wireless network**—Unless you need wireless, avoid it, particularly if your machine is a server. Never plug a server into a wireless network because it is just too fraught with security problems.

After you have ruled out these, you are onto the real problem: Which attack vectors are open on your server? In Internet terms, this comes down to which services are Internet-facing and which ports they are running on.

Two tools are often used to determine your vulnerabilities: Nmap and Nessus. Nessus scans your machine, queries the services running, checks their version numbers against its list of vulnerabilities, and reports problems.

Although Nessus sounds clever, it does not work well in many modern distributions (Ubuntu included) because of the way patches are made available to software. For example, if you're running Apache 2.0.52 and a bug is found that's fixed in 2.0.53, Ubuntu backports that patch to 2.0.52. This is done because the new release probably also includes new features that might break your code, so the Ubuntu team takes only what is necessary and copies it into your version. As a result, Nessus will see the version 2.0.52 and think it is vulnerable to a bug that has in fact been backported.

The better solution is to use Nmap, which scans your machine and reports on any open TCP/IP ports it finds. Any service you have installed that responds to Nmap's query is pointed out, which enables you to ensure that you have locked everything down as much as possible.

Nmap is available to install through Synaptic. Although you can use Nmap from a command line, it is easier to use with the front end—at least until you become proficient. To run the front end, select Actions, Run Application and run `nmapfe`. If you want to enable all Nmap's options, you need to switch to run `sudo nmapfe` from the console.

The best way to run Nmap is to use the SYN Stealth scan, with OS Detection and Version Probe turned on. You need to use sudo to enable the first two options (they are on by default when you use sudo), but it is well worth it. When you run Nmap (click the Scan button), it tests every port on your machine and checks whether it responds. If it does respond, Nmap queries it for version information and then prints its results onscreen.

The output lists the port numbers, service name (what usually occupies that port), and version number for every open port on your system. Hopefully, the information Nmap shows you will not be a surprise. If there is something open that you do not recognize, a hacker might have placed a backdoor on your system to allow herself easy access.

You should use the output from Nmap to help you find and eliminate unwanted services. The fewer services that are open to the outside world, the more secure you are.

Protecting Your Machine

After you have disabled all the unneeded services on your system, what remains is a core set of connections and programs that you want to keep. However, you are not finished yet: You need to clamp down your wireless network, lock your server physically, and put scanning procedures in place (such as Tripwire and promiscuous mode network monitors).

Securing a Wireless Network

Because wireless networking has some unique security issues, those issues deserve a separate discussion here.

Wireless networking, although convenient, can be very insecure by its very nature because transmitted data (even encrypted data) can be received by remote devices. Those devices could be in the same room; in the house, apartment, or building next door; or even several blocks away. Extra care must be used to protect the actual frequency used by your network. Great progress has been made in the past couple of years, but the possibility of a security breech is increased when the attacker is in the area and knows the frequency on which to listen. It should also be noted that the encryption method used by more wireless NICs is weaker than other forms of encryption (such as SSH) and should not be considered as part of your security plan.

TIP

Always use OpenSSH-related tools, such as `ssh` or `sftp`, to conduct business on your wireless LAN. Passwords are not transmitted as plain text, and your sessions are encrypted. See Chapter 19, "Remote Access with SSH and Telnet," to see how to connect to remote systems using `ssh`.

The better the physical security is around your network, the more secure it will be (this applies to wired networks as well). Keep wireless transmitters (routers, switches, and so on) as close to the center of your building as possible. Note or monitor the range of transmitted signals to determine whether your network is open to mobile network sniffing—now a geek sport known as *war driving*. (Linux software is available at http://sourceforge.net/project/showfiles.php?group_id=57253.) An occasional walk around your building not only gives you a break from work, but can also give you a chance to notice any people or equipment that should not be in the area.

Keep in mind that it takes only a single rogue wireless access point hooked up to a legitimate network hub to open access to your entire system. These access points can be smaller than a pack of cigarettes, so the only way to spot them is to scan for them with another wireless device.

Passwords and Physical Security

The next step toward better security is to use secure passwords on your network and ensure that users use them as well. For somewhat more physical security, you can force the use of a password with the LILO or GRUB bootloaders, remove bootable devices such as floppy and CD-ROM drives, or configure a network-booting server for Ubuntu. This approach is not well supported or documented at the time of this writing, but you can read about one way to do this in Brieuc Jeunhomme's Network Boot and Exotic Root HOWTO, available at http://www.tldp.org/HOWTO/Network-boot-HOWTO/. You can also read more about GRUB and LILO in Chapter 32, "Kernel and Module Management."

Also, keep in mind that some studies show that as much as 90% of network break-ins are by current or former employees. If a person no longer requires access to your network, lock out access or, even better, remove the account immediately. A good security policy also dictates that any data associated with the account first be backed up and retained for a set period of time to ensure against loss of important data. If you are able, remove the terminated employee from the system before he leaves the building.

Finally, be aware of physical security. If a potential attacker can get physical access to your system, getting full access becomes trivial. Keep all servers in a locked room, and ensure that only authorized personnel are given access to clients.

Configuring and Using Tripwire

Tripwire is a security tool that checks the integrity of normal system binaries and reports any changes to syslog or by email. Tripwire is a good tool for ensuring that your binaries have not been replaced by Trojan horse programs. *Trojan horses* are malicious programs inadvertently installed because of identical filenames to distributed (expected) programs, and they can wreak havoc on a breached system.

Ubuntu does not include the free version of Tripwire, but it can be used to monitor your system. To set up Tripwire for the first time, go to http://www.tripwire.org, and then download and install an open-source version of the software. After installation, run the `twinstall.sh` script (found under `/etc/tripwire`) as `root` like so:

```
$ sudo /etc/tripwire/twinstall.sh
----------------------------------------------
The Tripwire site and local passphrases are used to
sign a variety of files, such as the configuration,
policy, and database files.

Passphrases should be at least 8 characters in length
and contain both letters and numbers.

See the Tripwire manual for more information.

----------------------------------------------
Creating key files...

(When selecting a passphrase, keep in mind that good passphrases typically
have upper and lower case letters, digits and punctuation marks, and are
at least 8 characters in length.)

Enter the site keyfile passphrase:
```

You then need to enter a password of at least eight characters (perhaps best is a string of random madness, such as 5fwkc4ln) at least twice. The script generates keys for your site (host) and then asks you to enter a password (twice) for local use. You are then asked to enter the new site password. After following the prompts, the (rather extensive) default configuration and policy files (`tw.cfg` and `tw.pol`) are encrypted. You should then back up and delete the original plain-text files installed by Ubuntu.

To then initialize Tripwire, use its `--init` option like so:

```
$ sudo tripwire --init
Please enter your local passphrase:
```

```
Parsing policy file: /etc/tripwire/tw.pol
Generating the database...
*** Processing Unix File System ***
....
Wrote database file: /var/lib/tripwire/shuttle2.twd
The database was successfully generated.
```

Note that not all the output is shown here. After Tripwire has created its database (which is a snapshot of your file system), it uses this baseline along with the encrypted configuration and policy settings under the `/etc/tripwire` directory to monitor the status of your system. You should then start Tripwire in its integrity checking mode, using a desired option. (See the `tripwire` manual page for details.) For example, you can have Tripwire check your system and then generate a report at the command line, like so:

```
# tripwire -m c
```

No output is shown here, but a report is displayed in this example. The output could be redirected to a file, but a report is saved as `/var/lib/tripwire/report/hostname-YYYYMMDD-HHMMSS.twr` (in other words, using your host's name, the year, the month, the day, the hour, the minute, and the seconds). This report can be read using the `twprint` utility, like so:

```
# twprint --print-report -r \
/var/lib/tripwire/report/shuttle2-20020919-181049.twr ¦ less
```

Other options, such as emailing the report, are supported by Tripwire, which should be run as a scheduled task by your system's scheduling table, `/etc/crontab`, on off-hours. (It can be resource intensive on less powerful computers.) The Tripwire software package also includes a `twadmin` utility you can use to fine-tune or change settings or policies or to perform other administrative duties.

Devices

Do not ever advertise that you have set a NIC to promiscuous mode. Promiscuous mode (which can be set on an interface by using `ifconfig`'s `promisc` option) is good for monitoring traffic across the network and can often allow you to monitor the actions of someone who might have broken into your network. The `tcpdump` command also sets a designated interface to promiscuous mode while the program runs; unfortunately, the `ifconfig` command does not report this fact while `tcpdump` is running!

Do not forget to use the right tool for the right job. Although a network bridge can be used to connect your network to the Internet, it would not be a good option. Bridges have almost become obsolete because they forward any packet that comes their way, which is not good when a bridge is connected to the Internet. A router enables you to filter which packets are relayed.

Viruses

In the right hands, Linux is every bit as vulnerable to viruses as Windows is. That might come as a surprise to you, particularly if you made the switch to Linux on the basis of its security record. However, the difference between Windows and Linux is that it is much easier to secure against viruses on Linux. Indeed, as long as you are smart, you need never worry about them. Here is why:

- Linux never puts the current directory in your executable path, so typing `ls` runs `/bin/ls` rather than any `ls` in the current directory.
- A non-root user can infect only the files he has write access to, which is usually only the files in his home directory. This is one of the most important reasons for never using `sudo` when you don't need to!
- Linux forces you to mark files as executable, so you can't accidentally run a file called `myfile.txt.exe`, thinking it was just a text file.
- By having more than one common web browser and email client, Linux has strength through diversity: Virus writers cannot target one platform and hit 90% of the users.

Despite saying all that, Linux is susceptible to being a carrier for viruses. If you run a mail server, your Linux box can send virus-infected mails on to Windows boxes. The Linux-based server would be fine, but the Windows client would be taken down by the virus.

In this situation, you should consider a virus scanner for your machine. You have several to choose from, both free and commercial. The most popular free suite is Clam AV (http://www.clamav.net), but Central Command, BitDefender, F-Secure, Kaspersky, McAfee, and others all compete to provide commercial solutions—look around for the best deal before you commit.

Configuring Your Firewall

Always use a hardware-based or software-based firewall on computers connected to the Internet. Ubuntu has a graphical firewall configuration client named `gnome-lokkit`, along with a console-based firewall client named `lokkit`. Use these tools to implement selective or restrictive policies regarding access to your computer or LAN. Note: As always, make sure you enable Universe and Multiverse to get the full range of available software for your computer.

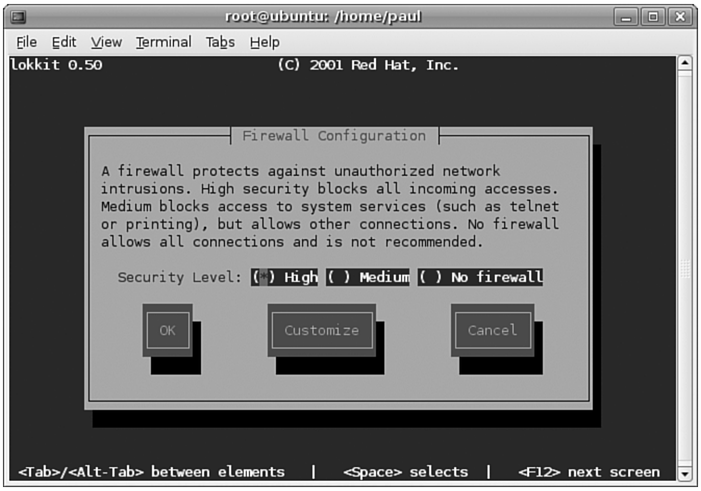
Start the `lokkit` command from a console or terminal window. You must run this command as root; otherwise, you will see an error message like this:

```
$ /usr/sbin/lokkit
ERROR - You must be root to run lokkit.
```

Use the `sudo` command to run `lokkit` like this:

```
$ sudo "/usr/sbin/lokkit"
```

After you press Enter, you see a dialog as shown in Figure 28.1. Press the Tab key to navigate to enable or disable firewalling. You can also customize your firewall settings to allow specific protocols access through a port and to designate an Ethernet interface for firewalling if multiple NICs are installed. Note that you can also use a graphical interface version of `lokkit` by running the `gnome-lokkit` client during an X session.



```
lokkit 0.50                    (C) 2001 Red Hat, Inc.

              Firewall Configuration

   A firewall protects against unauthorized network
   intrusions. High security blocks all incoming accesses.
   Medium blocks access to system services (such as telnet
   or printing), but allows other connections. No firewall
   allows all connections and is not recommended.

     Security Level: (*) High  ( ) Medium  ( ) No firewall

        OK            Customize            Cancel

<Tab>/<Alt-Tab> between elements   |   <Space> selects   |   <F12> next screen
```

FIGURE 28.1 Ubuntu's `lokkit` command quickly generates firewall rules in memory for Linux.

Forming a Disaster Recovery Plan

No one likes planning for the worst, which is why two thirds of people do not have wills. It is a scary thing to have your systems hacked: One or more criminals has broken through your carefully laid blocks and caused untold damage to the machine. Your boss, if you have one, will want a full report of what happened and why, and your users will want their email when they sit down at their desks in the morning. What to do?

If you ever do get hacked, nothing will take the stress away entirely. However, if you take the time to prepare a proper response in advance, you should at least avoid premature aging. Here are some tips to get you started:

- **Do not just pull the network cable out**—This alerts the hacker that he has been detected, which rules out any opportunities for security experts to monitor for that hacker returning and actually catch him.
- **Only inform the people who need to know**—Your boss and other IT people are at the top of the list; other employees are not. Keep in mind that it could be one of the employees behind the attack, and this tips them off.

- **If the machine is not required and you do not want to trace the attack, you can safely remove it from the network**—However, do not switch it off because some backdoors are only enabled when the system is rebooted.
- **Take a copy of all the log files on the system and store them somewhere else**—These might have been tampered with, but they might contain nuggets of information.
- **Check the `/etc/passwd` file and look for users you do not recognize**—Change all the passwords on the system, and remove bad users.
- **Check the output of `ps aux` for unusual programs running**—Also check to see whether any `cron` jobs are set to run.
- **Look in `/var/www` and see whether any web pages are there that should not be.**
- **Check the contents of the `.bash_history` files in the home directories of your users**—Are there any recent commands for your primary user?
- **If you have worked with external security companies previously, call them in for a fresh audit**—Hand over all the logs you have, and explain the situation. They will be able to extract all the information from the logs that is possible.
- **Start collating backup tapes from previous weeks and months**—Your system might have been hacked long before you noticed, so you might need to roll back the system more than once to find out when the attack actually succeeded.
- **Download and install Rootkit Hunter from http://www.rootkit.nl/projects/rootkit_hunter.html**—This searches for (and removes) the types of files that bad guys leave behind for their return.

Keep your disaster recovery plan somewhere safe; saving it as a file on the machine in question is a very bad move!

Keeping Up-to-Date on Linux Security Issues

A multitude of websites relate to security. One in particular hosts an excellent mailing list. The site is called Security Focus, and the mailing list is called BugTraq. BugTraq is well-known for its unbiased discussion of security flaws. Be warned: It receives a relatively large amount of traffic (20–100+ messages daily).

Often security holes are discussed on BugTraq before the software makers have even released the fix. The Security Focus site has other mailing lists and sections dedicated to Linux in general and is an excellent resource.

Related Ubuntu and Linux Commands

You will use these commands when managing security in your Ubuntu system:

`Ethereal`—GNOME graphical network scanner

`gnome-lokkit`—Ubuntu's basic graphical firewalling tool for X

`lokkit`—Ubuntu's basic graphical firewalling tool

`ssh`—The OpenSSH remote login client and preferred replacement for `telnet`

Reference

- http://www.insecure.org/nmap/—This site contains information on Nmap.
- http://www.securityfocus.com/—The Security Focus website.
- http://www.tripwire.org—Information and download links for the open-source version of Tripwire.
- http://www.ubuntu.com/usn—The official Ubuntu security notices list; well worth keeping an eye on!

CHAPTER 29
Performance Tuning

IN THIS CHAPTER

Squeezing extra performance out of your hardware might sound like a pointless task given how cheap commodity upgrades are today. To a certain degree that is true—for most of us, it is cheaper to buy a new computer than to spend hours fighting to get a 5% speed boost. But what if the speed boost were 20%? How about if it were 50%?

The amount of benefit you can get by optimizing your system varies depending on what kinds of tasks you are running, but there is something for everyone. Over the next few pages we will be looking at quick ways to optimize the Apache web server, both the KDE and Gnome desktop systems, both MySQL and PostgreSQL database servers, and more.

Before we start, you need to understand that *optimization* is not an absolute term: If we optimize a system, we have improved its performance, but it is still possible it could further be increased. We are not interested in getting 99.999% performance out of a system because optimization suffers from the law of diminishing returns—the basic changes make the biggest differences, but after that it takes increasing amounts of work to obtain decreasing speed improvements.

Hard Disk

Many Linux users love to tinker under the hood to increase the performance of their computers, and Linux gives you some great tools to do just that. Whereas Moms tell us, "Don't fix what's not broken," Dads often say, "Fix it until it breaks." In this section, you learn about many of the commands used to tune, or "tweak," your file system.

Before you undertake any "under the hood" work with Linux, however, keep a few points in mind. First, perform a benchmark on your system before you begin. Linux does not offer a well-developed benchmarking application, but availability changes rapidly. You can search online for the most up-to-date information for benchmarking applications for Linux. If you are a system administrator, you might choose to create your own benchmarking tests. Second, tweak only one thing at a time so you can tell what works, what does not work, and what breaks. Some of these tweaks might not work or might lock up your machine.

Always have a working boot disc handy and remember that you are personally assuming all risks for attempting any of these tweaks.

Using the BIOS and Kernel to Tune the Disk Drives

One method of tuning involves adjusting the settings in your BIOS. Because the BIOS is not Linux and every BIOS seems different, always read your motherboard manual for better possible settings and make certain that all the drives are detected correctly by the BIOS. Change only one setting at a time.

Linux does provide a limited means to interact with BIOS settings during the boot process (mostly overriding them). In this section, you will learn about those commands.

Other options are in the following list, and are more fully outlined in the BOOTPROMPT HOWTO and the kernel documentation. These commands can be used to force the IDE controllers and drives to be optimally configured. Of course, YMMV (Your Mileage May Vary) because these do not work for everyone.

`ide`*`x`*`=dma`—This will force DMA support to be turned on for the primary IDE bus, where x=0, or the secondary bus, where x=1.

`ide`*`x`*`=autotune`—This command will attempt to tune the interface for optimal performance.

`hd`*`x`*`=ide-scsi`—This command will enable SCSI emulation of an IDE drive. This is required for some CD-RW drives to work properly in write mode and it might provide some performance improvements for regular CD-R drives as well.

`idebus=`*`xx`*—This can be any number from 20 to 66; autodetection is attempted, but this can set it manually if `dmesg` says that it isn't autodetected correctly or if you have it set in the BIOS to a different value (overclocked). Most PCI controllers will be happy with `33`.

`pci=biosirq`—Some motherboards might cause Linux to generate an error message saying that you should use this. Look in `dmesg` for it; if you do not see it, you don't need to use it.

These options can be entered into `/etc/lilo.conf` or `/boot/grub/grub.conf` in the same way as other options are appended.

The hdparm Command

The hdparm utility can be used by root to set and tune the settings for IDE hard drives. You would do this to tune the drives for optimal performance.

Once a kernel patch and associated support programs, the hdparm program is now included with Ubuntu. You should only experiment with the drives mounted read-only because some settings can damage some file systems when used improperly. The hdparm command also works with CD-ROM drives and some SCSI drives.

The general format of the command is this:

```
# hdparm command device
```

This command runs a hard disk test:

```
hdparm -tT /dev/hda
```

You will need to replace /dev/hda with the location of your hard disk. hdparm will then run two tests—cached reads and buffered disk reads. A good IDE hard disk should be getting 400-500MB/sec for the first test, and 20-30MB/sec for the second. Note your scores, then try this command:

```
hdparm -m16 -d1 -u1 -c1 /dev/hda
```

That enables various performance-enhancing settings. Now try executing the original command again—if you see an increase, then you should run this command:

```
hdparm -m16 -d1 -u1 -c1 -k1 /dev/hda
```

The extra parameter tells hdparm to write the settings to disk so they will be used each time you boot up—ensuring optimal disk performance in the future.

The man entry for hdparm is extensive and contains useful detailed information, but since the kernel configuration selected by Ubuntu already attempts to optimize the drives, it might be that little can be gained through tweaking. Because not all hardware combinations can be anticipated by Ubuntu or by Linux and performance gains are always useful, you're encouraged to try.

TIP

You can use the hdparm command to produce a disk transfer speed result with

```
# hdparm -tT device
```

Be aware, however, that although the resulting numbers appear quantitative, they are subject to several technical qualifications beyond the scope of what is discussed and explained in this chapter. Simply put, do not accept values generated by hdparm as absolute numbers, but only as a relative measure of performance.

File System Tuning

Never content to leave things alone, Linux provides several tools to adjust and customize the file system settings. The belief is that hardware manufacturers and distribution creators tend to select conservation settings that will work well all the time, leaving some of the potential of your system leashed—that's why you have chosen Ubuntu Unleashed to help you.

The Linux file system designers have done an excellent job of selecting default values used for file system creation and the 2.6 version of the Linux kernel now contains new code for the IDE subsystem that significantly improves I/O (input/output) transfer speeds over older versions, obviating much of the need for special tweaking of the file system and drive parameters if you use IDE disks. Although these values work well for most users, some server applications of Linux benefit from file system tuning. As always, observe and benchmark your changes.

Synchronizing the File System with `sync`

Because Linux uses buffers when writing to devices, the write will not occur until the buffer is full, until the kernel tells it to, or if you tell it to by using the `sync` command. Traditionally, the command is given twice, as in the following:

```
# sync ; sync
```

It is really overkill to do it twice. Still, it can be helpful prior to the unmounting of certain types of media with slow write speeds (such as some USB hard drives or PCMCIA storage media), but only because it delays the user from attempting to remove the media too soon, not because two `sync`s are better than one.

The `tune2fs` Command

With `tune2fs`, you can adjust the tunable file system parameters on an `ext2` or `ext3` file system. A few performance-related items of note are as follows:

- To disable file system checking, the `-c 0` option sets the maximal mount count to zero.
- The interval between forced checks can be adjusted with the `-I` option.
- The `-m` option will set the reserved blocks percentage with a lower value, freeing more space at the expense of `fsck` having less space to write any recovered files.
- Decrease the number of superblocks to save space with the `-O sparse_super option`. (Modern file systems use this by default.) Always run `e2fsck` after you change this value.
- More space can be freed with the `-r` option that sets the number of reserved (for root) blocks.

Note that most of these uses of `tune2fs` free up space on the drive at the expense of the capability of `fsck` to recover data. Unless you really need the space and can deal with the consequences, just accept the defaults; large drives are now relatively inexpensive.

The `e2fsck` Command

This utility checks an `ext2/ext3` file system. Some useful arguments taken from `man e2fsck` are as follows:

`-c`—Checks for bad blocks and then marks them as bad.

`-f`—Forces checking on a clean file system.

`-v`—Verbose mode.

The `badblocks` Command

Although not a performance tuning program per se, the utility `badblocks` checks an (preferably) unmounted partition for bad blocks. It is not recommended that you run this command by itself, but rather allow it to be called by `fsck`. It should only be used directly if you specify the block size accurately—don't guess or assume anything.

The options available for `badblocks` are detailed in the `man` page. They allow for very low-level manipulation of the file system that is useful for data recovery by file system experts or for file system hacking, but are beyond the scope of this chapter and the average user.

Disabling File Access Time

Whenever Linux reads a file, it changes the last access time—known as the *atime*. This is also true for your web server: If you are getting hit by 50 requests a second, your hard disk will be updating the atime 50 times a second. Do you really need to know the last time a file was accessed? If not, you can disable atime setting for a directory by typing this:

```
chattr –R +A /path/to/directory
```

The `chattr` command changes file system attributes, of which "don't update atime" is one. To set that attribute, use `+A` and specify `–R` so that it is recursively set. `/path/to/directory` gets changed, and so do all the files and subdirectories it contains.

Kernel

As the Linux kernel developed over time, developers sought a way to fine-tune some of the kernel parameters. Before `sysctl`, those parameters had to be changed in the kernel configuration and then the kernel had to be recompiled.

The `sysctl` command can change some parameters of a running kernel. It does this through the `/proc` file system, which is a "virtual window" into the running kernel. Although it might appear that a group of directories and files exist under `/proc`, that is only a representation of parts of the kernel. When we're the root user (or using the sudo command), we can read values from and write values to those "files," referred to as *variables*. We can display a list of the variables as shown in the following. (An annotated list is presented because roughly 250 items [or more] exist in the full list.)

```
# sysctl -A
net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_rfc1337 = 0
net.ipv4.tcp_stdurg = 0
net.ipv4.tcp_abort_on_overflow = 0
net.ipv4.tcp_tw_recycle = 0
net.ipv4.tcp_syncookies = 0
net.ipv4.tcp_fin_timeout = 60
net.ipv4.tcp_retries2 = 15
net.ipv4.tcp_retries1 = 3
net.ipv4.tcp_keepalive_intvl = 75
net.ipv4.tcp_keepalive_probes = 9
net.ipv4.tcp_keepalive_time = 7200
net.ipv4.ipfrag_time = 30
```

The items shown are networking parameters, and actually tweaking these values is beyond the scope of this book. If we wanted to change a value, however, the `-w` parameter is used:

```
# sysctl -w net.ipv4.tcp_retries 2=20
```

This increases the value of that particular kernel parameter.

If you find that a particular setting is useful, you can enter it into the `/etc/sysctl.conf` file. The format is as follows, using the earlier example:

```
net.ipv4.tcp_retries 2=20
```

Of more interest to kernel hackers than regular users, `sysctl` is a potentially powerful tool that continues to be developed and documented.

TIP

The kernel does a good job of balancing performance for graphical systems, so there's not a great deal you can do to tweak your desktop to run faster.

Both GNOME and KDE are "heavyweight" desktop systems: They are all-inclusive, all-singing, and all-dancing environments that do far more than browse your file system. The drawback to this is that their size makes them run slow on older systems. On the flip side, Ubuntu also comes with the Xfce desktop, which is a great deal slimmer and faster than the other two. If you find GNOME and KDE are struggling just to open a file browser, Xfce is for you.

Apache

Despite being the most popular web server on the Internet, Apache is by no means the fastest. Part of the "problem" is that Apache has been written to follow every applicable standard to the letter, so much of its development work has been geared toward standards-compliancy rather than just serving web pages quickly. That said, with a little tweaking we can convert a $1,000 Dell server into something capable of surviving the Slashdot Effect.

NOTE

Slashdot.org is a popular geek news website that spawned the Slashdot Effect—the result of thousands of geeks descending on an unsuspecting website simultaneously. Our $1,000 Dell server had dual-2.8GHz Xeons with 1GB of RAM and SCSI hard disks—if you have more RAM, faster chips, and a high-end network card you can kick sand in Slashdot's face.

The first target for your tuning should be the `apache2.conf` file in `/etc/apache2`, as well as the other files in `/etc/apache2`. The more modules you have loaded, the more load Apache is placing on your server—take a look through the LoadModule list and comment out (start the line with a `#`) the ones you do not want. Some of these modules can be uninstalled entirely through the Add or Remove Packages dialog.

As a rough guide, you will almost certainly need `mod_mime` and `mod_dir`, and probably also `mod_log_config`. The default Apache configuration in Ubuntu is quite generic, so unless you are willing to sacrifice some functionality you may also need `mod_negotiation` (a speed killer if there ever was one), and `mod_access` (a notorious problem). Both of those last two modules can and should work with little or no performance decrease, but all too often they get abused and just slow things down.

Whatever you do, when you are disabling modules you should ensure you leave either `mod_deflate` or `mod_gzip` enabled, depending on your Apache version. Your bottleneck is almost certainly going to be your bandwidth rather than your processing power, and having one of these two compressing your content will usually turn 10Kb of HTML into 3Kb for supported browsers (most of them).

Next, ensure keepalives are turned off. Yes, you read that right: turn keepalives off. This adds some latency to people viewing your site, because they cannot download multiple files through the same connection. However, in turn it reduces the number of simultaneous open connections and so allows more people to connect.

If you are serving content that does not change, you can take the extreme step of enabling MMAP support. This allows Apache to serve pages directly from RAM without bothering to check whether they have changed, which works wonders for your performance. However, the downside is that when you do change your pages you need to restart Apache. Look for the `EnableMMAP` directive—it is probably commented out and set to off, so you will need to remove the comment and set it to on.

Finally, you should do all you can to ensure that your content is static: avoid PHP if you can, avoid databases if you can, and so on. If you know you are going to get hit by a rush of visitors, use plain HTML so that Apache is limited only by your bandwidth for how fast it can serve pages.

TIP

Some people, when questioned about optimizing Apache, will recommend you tweak the `HARD_SERVER_LIMIT` in the Apache source code and recompile. While we agree that compiling your own Apache source code is a great way to get a measurable speed boost if you know what you are doing, you should only ever need to change this directive if you are hosting a huge site.

The default value, 256, is enough to handle the Slashdot effect, and if you can handle that then you can handle most things.

MySQL

Tuning your MySQL server for increased performance is exceptionally easy to do, largely because you can see huge speed increases simply by getting your queries right. That said, there are various things you can tune in the server itself that help it cope with higher loads as long as your system has enough RAM.

The key is understanding its buffers—there are buffers and caches for all sorts of things, and finding out how full they are is crucial to maximizing performance. MySQL performs best when it is making full use of its buffers, which in turn places a heavy demand on system RAM. Unless you have 4GB RAM or more in your machine, you do not have enough capacity to set very high values for all your buffers—you need to pick and choose.

Measuring Key Buffer Usage

When you add indexes to your data, it enables MySQL to find data faster. However, ideally you want to have these indexes stored in RAM for maximum speed, and the variable `key_buffer_size` defines how much RAM MySQL can allocate for index key caching. If MySQL cannot store its indexes in RAM, you will experience serious performance problems. Fortunately, most databases have relatively small key buffer requirements, but you should measure your usage to see what work needs to be done.

To do this, log in to MySQL and type **`SHOW STATUS LIKE '%key_read%';`**. That returns all the status fields that describe the hit rate of your key buffer—you should get two rows back: `Key_reads` and `Key_read_requests`, which are the number of keys being read from disk and the number of keys being read from the key buffer. From these two numbers you can calculate the percentage of requests being filled from RAM and from disk, using this simple equation:

$$100 - ((\text{Key\_reads} / \text{Key\_read\_requests}) \times 100)$$

That is, you divide `Key_reads` by `Key_read_requests`, multiply the result by 100 and then subtract the result from 100. For example, if you have `Key_reads` of 1000 and `Key_read_requests` of 100000, you divide 1000 by 100000 to get 0.01; then you multiply that by 100 to get 1.0, and subtract that from 100 to get 99. That number is the percentage of key reads being served from RAM, which means 99% of your keys are served from RAM.

Most people should be looking to get more than 95% served from RAM, although the primary exception is if you update or delete rows very often—MySQL can't cache what keeps changing. If your site is largely read only, this should be around 98%. Lower figures mean you might need to bump up the size of your key buffer.

If you are seeing problems, the next step is to check how much of your current key buffer is being used. Use the `SHOW VARIABLES` command and look up the value of the `key_buffer_size` variable. It is probably something like 8388600, which is eight million bytes, or 8MB. Now, use the `SHOW STATUS` command and look up the value of `Key_blocks_used`.

You can now determine how much of your key buffer is being used by multiplying `Key_blocks_used` by 1024, dividing by `key_buffer_size`, and multiplying by 100. For example, if `Key_blocks_used` is 8000, you multiply that by 1024 to get 8192000; then you divide that by your `key_buffer_size` (8388600) to get 0.97656, and finally multiply that by 100 to get 97.656. Thus, almost 98% of your key buffer is being used.

Now, onto the important part: You have ascertained that you are reading lots of keys from disk, and you also now know that the reason for reading from disk is almost certainly because you do not have enough RAM allocated to the key buffer. A general rule of thumb is to allocate as much RAM to the key buffer as you can, up to a maximum of 25% of system RAM—128MB on a 512MB system is about the ideal for systems that read heavily from keys. Beyond that, you will actually see drastic performance decreases because the system has to use virtual memory for the key buffer.

Open `/etc/my.cnf` in your text editor, and look for the line that contains `key_buffer_size`. If you do not have one, you need to create a new one—it should be under the line `[mysqld]`. When you set the new value, do not just pick some arbitrarily high number. Try doubling what is there right now (or try 16MB if there's no line already); then see how it goes. To set 16MB as the key buffer size, you would need line like this:

```
[mysqld]
set-variable = key_buffer_size=16M
datadir=/var/lib/mysql
```

Restart your MySQL server with `service mysqld restart`, and then go back into MySQL and run `SHOW VARIABLES` again to see the `key_buffer_size`. It should be 16773120 if you have set it to 16M. Now, because MySQL just got reset, all its values for key hits and the like will also have been reset. You need to let it run for a while so you can assess how much has changed. If you have a test system you can run, this is the time to run it.

After your database has been accessed with normal usage for a short while (if you get frequent accesses, this might be only a few minutes), recalculate how much of the key buffer is being used. If you get another high score, double the size again, restart, and retest. You should keep repeating this until you see your key buffer usage is below 50% or you find you don't have enough RAM to increase the buffer further—remember that you should never allocate more than 25% of system RAM to the key buffer.

Using the Query Cache

Newer versions of MySQL allow you to cache the results of queries so that, if new queries come in that use exactly the same SQL, the result can be served from RAM. In some ways the query cache is quite intelligent: If, for example, part of the result changes due to another query, the cached results are thrown away and recalculated next time. However, in other ways it is very simple. For example, it uses cached results only if the new query is exactly the same as a cached query, even down to the capitalization of the SQL.

The query cache works well in most scenarios. If your site has an equal mix of reading and writing, the query cache will do its best but will not be optimal. If your site is mostly reading with few writes, more queries will be cached (and for longer), thus improving overall performance.

First, you need to find out whether you have the query cache enabled. To do this, use `SHOW VARIABLES` and look up the value of `have_query_cache`. All being well, you should get `YES` back, meaning that the query cache is enabled. Next, look for the value of `query_cache_size` and `query_cache_limit`—the first is how much RAM in bytes is allocated to the query cache, and the second is the maximum result size that should be cached. A good starting set of values for these two is 8388608 (8MB) and 1048576 (1MB).

Next, type **`SHOW STATUS LIKE 'Qcache%';`** to see all the status information about the query cache. You should get output like this:

```
mysql> SHOW STATUS LIKE 'qcache%';
+-------------------------+--------+
¦ Variable_name           ¦ Value  ¦
+-------------------------+--------+
¦ Qcache_free_blocks      ¦ 1      ¦
¦ Qcache_free_memory      ¦ 169544 ¦
¦ Qcache_hits             ¦ 698    ¦
¦ Qcache_inserts          ¦ 38     ¦
¦ Qcache_lowmem_prunes    ¦ 20     ¦
¦ Qcache_not_cached       ¦ 0      ¦
¦ Qcache_queries_in_cache ¦ 18     ¦
¦ Qcache_total_blocks     ¦ 57     ¦
+-------------------------+--------+
8 rows in set (0.00 sec)
```

From that, we can see that only 18 queries are in the cache (`Qcache_queries_in_cache`), we have 169544 bytes of memory free in the cache (`Qcache_free_memory`), 698 queries have been read from the cache (`Qcache_hits`), 38 queries have been inserted into the cache (`Qcache_inserts`), but 20 of them were removed due to lack of memory (`Qcache_lowmem_prunes`), giving the 18 from before. `Qcache_not_cached` is 0, which means 0 queries were not cached—MySQL is caching them all.

From that, we can calculate how many total queries came in—it is the sum of `Qcache_hits`, `Qcache_inserts`, and `Qcache_not_cached`, which is 736. We can also calculate how well the query cache is being used by dividing `Qcache_hits` by that number and multiplying by 100. In this case, 94.84% of all queries are being served from the query cache, which is a great number.

In our example, we can see that many queries have been trimmed because there is not enough memory in the query cache. This can be changed by editing your `/etc/my.cnf` file and adding a line like this one, somewhere in the `[mysqld]` section:

```
set-variable = query_cache_size=32M
```

An 8MB query cache should be enough for most people, but larger sites might want 16MB or even 32MB if you are storing a particularly large amount of data. Very few sites will have need to go beyond a 32MB query cache, but keep an eye on the `Qcache_lowmem_prunes` value to ensure you have enough RAM allocated.

Using the query cache does not incur much of a performance hit. When MySQL calculates the result of a query normally, it simply throws it away when the connection closes. With the query cache, it skips the throwing away, and so there is no extra work being done. If your site does have many updates and deletes, be sure to check whether you get any speed boost at all from the query cache.

Miscellaneous Tweaks

If you have tuned your key buffer and optimized your query cache and yet still find your site struggling, there are a handful more smaller changes you make that will add some more speed.

When reading from tables, MySQL has to open the file that stores the table data. How many files it keeps open at a time is defined by the `table_cache` setting, which is set to 64 by default. You can increase this setting if you have more than 64 tables, but you should be aware that Ubuntu does impose limits on MySQL about how many files it can have open at a time. Going beyond 256 is not recommended unless you have a particularly DB-heavy site and know exactly what you are doing.

The other thing you can tweak is the size of the read buffer, which is controlled by `read_buffer_size` and `read_buffer_rnd_size`. Both of these are allocated per connection, which means you should be very careful to have large numbers. Whatever you choose, `read_buffer_rnd_size` should be three to four times the size of `read_buffer_size`, so if `read_buffer_size` is 1MB (suitable for very large databases), `read_buffer_rnd_size` should be 4MB.

Query Optimization

The biggest speed-ups can be seen by reprogramming your SQL statements so they are more efficient. If you follow these tips, your server will thank you:

- Select as little data as possible. Rather than `SELECT *`, select only the fields you need.
- If you only need a few rows, use `LIMIT` to select the number you need.

- Declare fields as `NOT NULL` when creating tables to save space and increase speed.
- Provide default values for fields, and use them where you can.
- Be very careful with table joins because they are the easiest way to write inefficient queries.
- If you must use joins, be sure you join on fields that are indexed. They should also preferably be integer fields, as these are faster than strings for comparisons.
- Find and fix slow queries. Add `log-long-format` and `log-slow-queries = /var/log/slow-queries.log` to your `/etc/my.cnf` file, under `[mysqld]`, and MySQL will tell you the queries that took a long time to complete.
- Use `OPTIMIZE TABLE tablename` to defragment tables and refresh the indexes.

Reference

- http://www.coker.com.au/bonnie++/—The home page of bonnie, a disk benchmarking tool. It also contains a link to RAID benchmarking utilities and Postal, a benchmarking utility for SMTP servers.
- http://httpd.apache.org/docs-2.0/misc/perf-tuning.html—The official Apache guide to tuning your web server.
- http://dev.mysql.com/doc/refman/en/5.0-optimization.html—Learn how to optimize your MySQL server direct from the source, the MySQL manual.

There is one particular MySQL optimization book that will really help you get more from your system if you run a large site, and that is *High Performance MySQL* by Jeremy Zawodny and Derek Balling (O'Reilly), ISBN: 0-596-00306-4.

CHAPTER 30 Command Line Masterclass

IN THIS CHAPTER

In the earlier years of Linux, people made a big fuss of the graphical environments that were available—they rushed to tell new users that you really did not need to keep going back to the command line to type things. Now Linux is more mature, people accept that the command line is still a reality, and a welcome one. While the GUI does indeed make life easier for day-to-day tasks, your options are limited by what the developers wanted you to have—you cannot bring commands together in new ways, and neither can you use any of the GUI features if, for example, your GUI is broken. It does happen!

In his book *The Art of Unix Programming**, Eric Raymond wrote a short story that perfectly illustrates the power of the command line versus the GUI. It's reprinted here with permission, for your reading pleasure:

> One evening, Master Foo and Nubi attended a gathering of programmers who had met to learn from each other. One of the programmers asked Nubi to what school he and his master belonged. Upon being told they were followers of the Great Way of Unix, the programmer grew scornful.
>
> "The command-line tools of Unix are crude and backward," he scoffed. "Modern, properly designed operating systems do everything through a graphical user interface."
>
> Master Foo said nothing, but pointed at the moon. A nearby dog began to bark at the master's hand.

Raymond, Eric. The Art of Unix Programming. Addison-Wesley, 2004.

> "I don't understand you!" said the programmer.
>
> Master Foo remained silent, and pointed at an image of the Buddha. Then he pointed at a window.
>
> "What are you trying to tell me?" asked the programmer.
>
> Master Foo pointed at the programmer's head. Then he pointed at a rock.
>
> "Why can't you make yourself clear?" demanded the programmer.
>
> Master Foo frowned thoughtfully, tapped the programmer twice on the nose, and dropped him in a nearby trash can.
>
> As the programmer was attempting to extricate himself from the garbage, the dog wandered over and piddled on him.
>
> At that moment, the programmer achieved enlightenment.

Whimsical as the story is, it does illustrate that there are some things that the GUI just does not do well. Enter the command line: It is a powerful and flexible operating environment on Linux, and—if you practice—can actually be quite fun, too!

In this chapter, you will learn how to master the command line so that you are able to perform common tasks through it, and also link commands together to create new command groups. We will also be looking at the two most popular Linux text editors: Vim and Emacs. The command line is also known as the shell, the console, the command prompt, and the CLI (command-line interpreter). For the purposes of this chapter these are interchangeable, although there are fine-grained differences between them!

Why Use the Shell?

Moving from the GUI to the command line is a conscious choice for most people, although it is increasingly rare that it is an either/or choice. While in X, you can press Ctrl+Alt+F1 at any time to switch to a terminal, but most people tend to run an X terminal application that allows them to have the point-and-click applications and the command-line applications side by side.

Reasons for running the shell include:

- You want to chain two or more commands together.
- You want to use a command or parameter available only on the shell.
- You are working on a text-only system.
- You have used it for a long time and feel comfortable there.

Chaining two or more commands together is what gives the shell its real power. Hundreds of commands are available and, by combining them in different ways, you get hundreds of new commands. Some of the shell commands are available through the GUI, but these commands usually have only a small subset of their parameters available, which limits what you are able to do with them.

Working from a text-only system encapsulates both working locally with a broken GUI and also connecting to a remote, text-only system. If your Linux server is experiencing problems, the last thing you want to do is load it down with a GUI connection—working in text mode is faster and more efficient.

The last use is the most common: People use the shell just because it is familiar to them, with many people even using the shell to start GUI applications just because it saves them taking their hands off the keyboard for a moment! This is not a bad thing; it provides fluency and ease with the system and is a perfectly valid way of working.

Basic Commands

It is impossible to know how many commands the average shell citizen uses, but if we had to guess, we would place it at about 25:

- `cat`—Prints the contents of a file
- `cd`—Changes directories
- `chmod`—Changes file access permissions
- `cp`—Copies files
- `du`—Prints disk usage
- `emacs`—Text editor
- `find`—Finds files by searching
- `gcc`—The C/C++/Fortran compiler
- `grep`—Searches for a string in input
- `less`—Filter for paging through output
- `ln`—Creates links between files
- `locate`—Finds files from an index
- `ls`—Lists files in the current directory
- `make`—Compiles and installs programs
- `man`—The manual page reader
- `mkdir`—Makes directories
- `mv`—Moves files
- `ps`—Lists processes
- `rm`—Deletes files and directories

- `ssh`—Connects to other machines
- `tail`—Prints the last lines of a file
- `top`—Prints resource usage
- `vim`—A text editor
- `which`—Prints the location of a command
- `xargs`—Executes commands from its input

Of course, many other commands are available to you that get used fairly often—`diff`, `nmap`, `ping`, `su`, `uptime`, `who`, and so on—but if you are able to understand the 25 listed here, you will have sufficient skill to concoct your own command combinations.

Note that we say *understand* the commands—not know all their possible parameters and usages. This is because several of the commands, though commonly used, are used only in any complex manner by people with specific needs. `gcc` and `make` are good examples of this: Unless you plan to become a programmer, you need not worry about these beyond just typing `make` and `make install` now and then. If you want to learn more about these two, see Chapter 26, "C/C++ Programming Tools for Ubuntu."

Similarly, both Emacs and Vim are text editors that have a text-based interface all of their own, and SSH has already been covered in detail in Chapter 15, "Remote Access with SSH and Telnet."

What remains is 20 commands, each of which has many parameters to customize what it actually does. Again, we can eliminate much of this because many of the parameters are esoteric and rarely used, and, the few times in your Linux life that you need them, you can just read the manual page!

We will be going over these commands one by one, explaining the most common ways to use them. There is one exception to this: The `xargs` command requires you to understand how to join commands together.

Printing the Contents of a File with `cat`

Many of Ubuntu's shell commands manipulate text strings, so if you want to be able to feed them the contents of files, you need to be able to output those files as text. Enter the `cat` command, which prints the contents of any files you pass to it.

Its most basic use is like this:

```
cat myfile.txt
```

That prints the contents of `myfile.txt`. For this usage there are two extra parameters that are often used: `-n` numbers the lines in the output, and `-s` ("squeeze") prints a maximum of one blank line at a time. That is, if your file has 1 line of text, 10 blank lines, 1 line of text, 10 blank lines, and so on, `-s` shows the first line of text, a single blank line, the next line of text, a single blank line, and so forth. When you combine `-s` and `-n`, `cat` numbers only the lines that are printed—the 10 blank lines shown as one will count as 1 line for numbering.

This command prints information about your CPU, stripping out multiple blank lines and numbering the output:

```
cat -sn /proc/cpuinfo
```

You can also use `cat` to print the contents of several files at once, like this:

```
cat -s myfile.txt myotherfile.txt
```

In that command, `cat` merges `myfile.txt` and `myotherfile.txt` on the output, stripping out multiple blank lines. The important thing is that `cat` does not distinguish between the files in the output—there are no filenames printed, and no extra breaks between the 2. This allows you to treat the 2 as 1 or, by adding more files to the command line, to treat 20 files as 1.

Changing Directories with `cd`

Changing directories is surely something that has no options, right? Not so. `cd` is actually more flexible than most people realize. Unlike most of the other commands here, `cd` is not a command in itself—it is built in to `bash` (or whichever shell interpreter you are using), but it is still used like a command.

The most basic usage of `cd` is this:

```
cd somedir
```

That looks in the current directory for the `somedir` subdirectory, then moves you into it. You can also specify an exact location for a directory, like this:

```
cd /home/paul/stuff/somedir
```

The first part of `cd`'s magic lies in the characters (`-` and `~`, a dash and a tilde). The first means "switch to my previous directory," and the second means "switch to my home directory." This conversation with `cd` shows this in action:

```
[paul@caitlin ~]$ cd /usr/local
[paul@caitlin local]$ cd bin
[paul@caitlin bin]$ cd -
/usr/local
[paul@caitlin local]$ cd ~
[paul@caitlin ~]$
```

In the first line, we change to `/usr/local` and get no output from the command. In the second line, we change to `bin`, which is a subdirectory of `/usr/local`. Next, `cd -` is used to change back to the previous directory. This time Bash prints the name of the previous directory so we know where we are. Finally, `cd ~` is used to change back to our home directory, although if you want to save an extra few keystrokes, just typing `cd` by itself is equivalent to `cd ~`.

The second part of `cd`'s magic is its capability to look for directories in predefined locations. When you specify an absolute path to a directory (that is, one starting with a `/`), `cd` always switches to that exact location. However, if you specify a relative subdirectory—for example, `cd subdir`—you can tell `cd` where you would like that to be relative to. This is accomplished with the `CDPATH` environment variable. If this variable is not set, `cd` always uses the current directory as the base; however, you can set it to any number of other directories.

This next example shows a test of this. It starts in `/home/paul/empty`, an empty directory, and the lines are numbered for later reference:

```
1 [paul@caitlin empty]$ pwd
2 /home/paul/empty
3 [paul@caitlin empty]$ ls
4 [paul@caitlin empty]$ mkdir local
5 [paul@caitlin empty]$ ls
6 local
7 [paul@caitlin empty]$ cd local
8 [paul@caitlin local]$ cd ..
9 [paul@caitlin empty]$ export CDPATH=/usr
10 [paul@caitlin empty]$ cd local
11 /usr/local
12 [paul@caitlin empty]$ cd -
13 /home/paul/empty
14 [paul@caitlin empty]$ export CDPATH=.:/usr
15 [paul@caitlin empty]$ cd local
16 /home/paul/empty/local
17 [paul@caitlin local]$
```

Lines 1–3 show that we are in `/home/paul/empty` and that it is indeed empty—`ls` had no output. Lines 4–6 show the `local` subdirectory being made, so that `/home/paul/empty/local` exists. Lines 7 and 8 show you can `cd` into `/home/paul/empty/local` and back out again.

In line 9, `CDPATH` is set to `/usr`. This was chosen because Ubuntu has the directory `/usr/local`, which means our current directory (`/home/paul/empty`) and our `CDPATH` directory (`/usr`) both have a `local` subdirectory. In line 10, while in the `/home/paul/empty` directory, we use `cd local`. This time, Bash switches us to `/usr/local` and even prints the new directory to ensure we know what it has done.

Lines 12 and 13 move us back to the previous directory, /home/paul/empty. In line 14, CDPATH is set to be .:/usr. The : is the directory separator, so this means Bash should look first in the current directory, ., and then in the /usr directory. In line 15 cd local is issued again, this time moving to /home/paul/empty/local. Note that Bash has still printed the new directory—it does that whenever it looks up a directory in CDPATH.

Changing File Access Permissions with chmod

Your use of chmod can be greatly extended through one simple parameter: -c. This instructs chmod to print a list of all the changes it made as part of its operation, which means we can capture the output and use it for other purposes. For example:

```
[paul@caitlin tmp]$ chmod -c 600 *
mode of `1.txt' changed to 0600 (rw-------)
mode of `2.txt' changed to 0600 (rw-------)
mode of `3.txt' changed to 0600 (rw-------)
[paul@caitlin tmp]$ chmod -c 600 *
[paul@caitlin tmp]$
```

There the chmod command is issued with -c, and you can see it has output the result of the operation: Three files were changed to rw------- (read and write by user only). However, when the command is issued again, no output is returned. This is because -c prints only the changes that it made. Files that already match the permissions you are setting are left unchanged and therefore are not printed.

There are two other parameters of interest: --reference and -R. The first allows you to specify a file to use as a template for permissions rather than specifying permissions yourself. For example, if you want all files in the current directory to have the same permissions as the file /home/paul/myfile.txt, you would use this:

```
chmod --reference /home/paul/myfile.txt *
```

You can use -R to enable recursive operation, which means you can use it to chmod a directory and it will change the permissions of that directory as well as all files and subdirectories under that directory. You could use chmod -R 600 /home to change every file and directory under /home to become read/write to their owner.

Copying Files with cp

Like mv, which is covered later, cp is a command that is easily used and mastered. However, two marvelous parameters rarely see much use (which is a shame!) despite their power. These are --parents and -u, the first of which copies the full path of the file into the new directory. The second copies only if the source file is newer than the destination.

Using --parents requires a little explanation, so here is an example. You have a file, /home/paul/desktop/documents/work/notes.txt, and want to copy it to your /home/paul/backup folder. You could just do a normal cp, but that would give you

/home/paul/backup/notes.txt, so how would you know where that came from later? If you use --parents, the file is copied to /home/paul/backup/desktop/documents/work/notes.txt.

The -u parameter is perfect for synchronizing two directories because it allows you to run a command like cp -Ru myfiles myotherfiles and have cp recopy only files that have changed. The -R parameter means *recursive* and allows you to copy directory contents.

Printing Disk Usage with du

The du command prints the size of each file and directory that is inside the current directory. Its most basic usage is as easy as it gets:

```
du
```

That outputs a long list of directories and how much space their files take up. You can modify that with the -a parameter, which instructs du to print the size of individual files as well as directories. Another useful parameter is -h, which makes du use human-readable sizes like 18M (18MB) rather than 17532 (the same number in bytes, unrounded). The final useful basic option is -c, which prints the total size of files.

So, using du we can get a printout of the size of each file in our home directory, in human-readable format, and with a summary at the end, like this:

```
du -ahc /home/paul
```

Two advanced parameters deal with filenames you want excluded from your count. The first is --exclude, which allows you to specify a pattern that should be used to exclude files. This pattern is a standard shell file matching pattern as opposed to a regular expression, which means you can use ? to match a single character or * to match 0 or many characters. You can specify multiple --exclude parameters to exclude several patterns. For example:

```
du --exclude="*.xml" --exclude="*.xsl"
```

Of course, typing numerous --exclude parameters repeatedly is a waste of time, so you can use -X to specify a file that has the list of patterns you want excluded. The file should look like this:

```
*.xml
*.xsl
```

That is, each pattern you want excluded should be on a line by itself. If that file were called xml_exclude.txt, we could use it in place of the previous example like this:

```
du -X xml_exclude.txt
```

You can make your exclusion file as long as you need, or you can just specify multiple `-X` parameters.

TIP

Running `du` in a directory where several files are hard-linked to the same inode counts the size of the file only once. If you want to count each hard link separately for some reason, use the `-l` parameter (lowercase *L*).

Finding Files by Searching with `find`

The `find` command is one of the darkest and least understood areas of Linux, but it is also one of the most powerful. Admittedly, the `find` command does not help itself by using X-style parameters. The UNIX standard is `-c`, `-s`, and so on, whereas the GNU standard is `--dosomething`, `--mooby`, and so forth. X-style parameters merge the two by having words preceded by only one dash.

However, the biggest problem with `find` is that it has more options than most people can remember—it truly is capable of doing most things you could want. The most basic usage is this:

```
find -name "*.txt"
```

That searches the current directory and all subdirectories for files that end in `.txt`. The previous search finds files ending in `.txt` but not `.TXT`, `.Txt`, or other case variations. To search without case sensitivity, use `-iname` instead of `-name`. You can optionally specify where the search should start before the `-name` parameter, like this:

```
find /home -name "*.txt"
```

Another useful test is `-size`, which lets you specify how big the files should be to match. You can specify your size in kilobytes and optionally also use `+` or `-` to specify greater than or less than. For example:

```
find /home -name "*.txt" -size 100k
find /home -name "*.txt" -size +100k
find /home -name "*.txt" -size -100k
```

The first brings up files of exactly 100KB, the second only files larger than 100KB, and the last only files under 100KB.

Moving on, the `-user` option enables you to specify the user who owns the files you are looking for. So, to search for all files in `/home` that end with `.txt`, are under 100KB, and are owned by user `paul`, you would use this:

```
find /home -name "*.txt" -size -100k -user paul
```

You can flip any of the conditions by specifying -not before them. For example, you can add a -not before -user paul to find matching files owned by everyone *but* paul:

```
find /home -name "*.txt" -size -100k -not -user paul
```

You can add as many -not parameters as you need, even using -not -not to cancel each other out! (Yes, that is pointless.) Keep in mind, though, that -not -size -100k is essentially equivalent to -size +100k, with the exception that the former will match files of exactly 100KB whereas the latter will not.

You can use -perm to specify which permissions a file should have for it to be matched. This is tricky, so read carefully. The permissions are specified in the same way as with the chmod command: u for user, g for group, o for others, r for read, w for write, and x for execute. However, before you give the permissions, you need to specify either a plus, a minus, or a blank space. If you specify neither a plus nor a minus, the files must exactly match the mode you give. If you specify -, the files must match all the modes you specify. If you specify +, the files must match any the modes you specify. Confused yet?

The confusion can be cleared up with some examples. This next command finds all files that have permission o=r (readable for other users). Notice that if you remove the -name parameter, it is equivalent to * because all filenames are matched.

```
find /home -perm -o=r
```

Any files that have o=r set are returned from that query. Those files also might have u=rw and other permissions, but as long as they have o=r, they will match. This next query matches all files that have o=rw set:

```
find /home -perm -o=rw
```

However, that query does not match files that are o=r or o=w. To be matched, a file must be readable *and* writeable by other users. If you want to match readable *or* writeable (or both), you need to use +, like this:

```
find /home -perm +o=rw
```

Similarly, this next query matches files only that are readable by user, group, and others:

```
find /home -perm -ugo=r
```

Whereas this query matches files as long as they are readable by the user, or by the group, or by others, or by any combination of the three:

```
find /home -perm +ugo=r
```

If you use neither + or -, you are specifying the exact permissions to search for. For example, the next query searches for files that are readable by user, group, and others but not writeable or executable by anyone:

```
find /home -perm ugo=r
```

You can be as specific as you need to be with the permissions. For example, this query finds all files that are readable for the user, group, and others and writeable by the user:

```
find /home -perm ugo=r,u=w
```

To find files that are not readable by others, use the `-not` condition, like this:

```
find /home -not -perm +o=r
```

Now, on to the most advanced aspect of the find command: the `-exec` parameter. This enables you to execute an external program each time a match is made, passing in the name of the matched file wherever you want it. This has very specific syntax: Your command and its parameters should follow immediately after `-exec`, terminated by `\;`. You can insert the filename match at any point using `{}` (an opening and a closing brace side by side).

So, you can match all text files on the entire system (that is, searching recursively from `/` rather than from `/home` as in our previous examples) over 10KB, owned by `paul`, that are not readable by other users, and then use `chmod` to enable reading, like this:

```
find / -name "*.txt" -size +10k -user paul -not -perm +o=r -exec chmod o+r {} \;
```

When you type your own `-exec` parameters, be sure to include a space before `\;`. Otherwise, you might see an error such as `missing argument to `-exec'`.

Do you see now why some people think the `find` command is scary? Many people learn just enough about `find` to be able to use it in a very basic way, but hopefully you will see how much it can do if you give it chance.

Searches for a String in Input with `grep`

The `grep` command, like `find`, is an incredibly powerful search tool in the right hands. Unlike `find`, though, `grep` processes any text, whether in files, or just in standard input.

The basic usage of `grep` is this:

```
grep "some text" *
```

That searches all files in the current directory (but not subdirectories) for the string `some text` and prints matching lines along with the name of the file. To enable recursive searching in subdirectories, use the `-r` parameter, like this:

```
grep -r "some text" *
```

Each time a string is matched within a file, the filename and the match are printed. If a file contains multiple matches, each of the matches is printed. You can alter this behavior with the `-l` parameter (lowercase *L*), which forces `grep` to print the name of each file that contains at least one match, without printing the matching text. If a file contains more than one match, it is still printed only once. Alternatively, the `-c` parameter prints each filename that was searched and includes the number of matches at the end, even if there were no matches.

You have a lot of control when specifying the pattern to search for. You can, as we did previously, specify a simple string like `some text`, or you can invert that search by specifying the `-v` parameter. For example, this returns all the lines of the file `myfile.txt` that do not contain the word `hello`:

```
grep -v "hello" myfile.txt
```

You can also use regular expressions for your search term. For example, you can search `myfile.txt` for all references to `cat`, `sat`, or `mat` with this command:

```
grep "[cms]at" myfile.txt
```

Adding the `-i` parameter to that removes case sensitivity, matching `Cat`, `CAT`, `MaT`, and so on:

```
grep -i [cms]at myfile.txt
```

The output can also be controlled to some extent with the `-n` and `--color` parameters. The first tells `grep` to print the line number for each match, which is where it appears in the source file. The `--color` parameter tells `grep` to color the search terms in the output, which helps them stand out when among all the other text on the line. You choose which color you want using the `GREP_COLOR` environment variable: `export GREP_COLOR=36` gives you cyan, and `export GREP_COLOR=32` gives you lime green.

This next example uses these two parameters to number and color all matches to the previous command:

```
grep -in --color [cms]at myfile.txt
```

Later you will see how important `grep` is for piping with other commands.

Paging Through Output with `less`

The `less` command enables you to view large amounts of text in a more convenient way than the `cat` command. For example, your `/etc/passwd` file is probably more than a screen long, so if you run `cat /etc/passwd`, you are not able to see what the lines at the top were. Using `less /etc/passwd` enables you to use the cursor keys to scroll up and down the output freely. Type `q` to quit and return to the shell.

On the surface, `less` sounds like a very easy command; however, it has the infamy of being one of the few Linux commands that have a parameter for every letter of the alphabet. That is, `-a` does something, `-b` does something else, `-c`, `-d`, `-e` … `-x`, `-y`, `-z`—they all do things, with some letters even differentiating between upper- and lowercase.

Furthermore, it is only the parameter used to invoke less. After you are viewing your text, even more commands are available to you. Make no mistake—less is a complex beast to master.

Input to less can be divided into two categories: what you type before running less and what you type while running it. The former category is easy, so we shall start there.

We have already discussed how many parameters less is capable of taking, but they can be distilled down to three that are very useful: -M, -N, and +. Adding -M (this is different from -m!) enables verbose prompting in less. Rather than just printing a colon and a flashing cursor, less prints the filename, the line numbers being shown, the total number of lines, and the percentage of how far you are through the file. Adding -N (again, this is different from -n) enables line numbering.

The last option, +, allows you to pass a command to less for it to execute as it starts. To use this, you first need to know the commands available to you in less, which means we need to move onto the second category of less input: what you type while less is running.

The basic navigation keys are the up, down, left, and right cursors; Home and End (for navigating to the start and end of a file); and Page Up and Page Down. Beyond that, the most common command is /, which initiates a text search. You type what you want to search for and press Enter to have less find the first match and highlight all subsequent matches. Type / again and press Enter to have less jump to the next match. The inverse of that is ?, which searches backward for text. Type ?, enter a search string, and press Enter to go to the first previous match of that string, or just use ? and press Enter to go to the next match preceding the current position. You can use / and ? interchangeably by searching for something with / and then using ? to go backward in the same search.

Searching with / and ? is commonly used with the + command-line parameter from earlier, which passes less a command for execution after the file has loaded. For example, you can tell less to load a file and place the cursor at the first match for the search hello, like this:

```
less +/hello myfile.txt
```

Or, to place the cursor at the last match for the search hello:

```
less +?hello myfile.txt
```

Beyond the cursor keys, the controls primarily involve typing a number and then pressing a key. For example, to go to line 50 and type 50g, or to go to the 75% point of the file and type 75p. You can also place invisible mark points through the file by pressing m and then typing a single letter. Later, while in the same less session, you can press ' (a single quote) and then type the letter, and it will move you back to the same position. You can set up to 52 marks, named a–z and A–Z.

One clever feature of `less` is that you can, at any time, press `v` to have your file opened inside your text editor. This defaults to Vim, but you can change that by setting the `EDITOR` environment variable to something else.

If you have made it this far, you can already use `less` better than most users. You can, at this point, justifiably skip to the next section and consider yourself proficient with `less`. However, if you want to be a `less` guru, there are two more things to learn: how to view multiple files simultaneously and how to run shell commands.

Like most other file-based commands in Linux, `less` can take several files as its parameters. For example:

```
less -MN 1.txt 2.txt 3.txt
```

That loads all three files into `less`, starting at `1.txt`. When viewing several files, `less` usually tells you which file you are in, as well as numbering them: `1.txt (file 1 of 3)` should be at the bottom of the screen. That said, certain things will make that go away, so you should use `-M` anyway.

You can navigate between files by typing a colon and then pressing `n` to go to the next file or `p` to go to the previous file; these are referred to from now on as `:n` and `:p`. You can open another file for viewing by typing `:e` and providing a filename. This can be any file you have permission to read, including files outside the local directory. Use Tab to complete filenames. Files you open in this way are inserted one place after your current position, so if you are viewing file 1 of 4 and open a new file, the new file is numbered 2 of 5 and is opened for viewing straight away. To close a file and remove it from the list, use `:d`.

Viewing multiple files simultaneously has implications for searching. By default, `less` searches within only one file, but it is easy to search within all files. When you type `/` or `?` to search, follow it with a `*`—you should see `EOF-ignore` followed by a search prompt. You can now type a search and it will search in the current file; if nothing is found, it looks in subsequent files until it finds a match. Repeating searches is done by pressing Esc and then either `n` or `N`. The lowercase option repeats the search forward across files, and the uppercase repeats it backward.

The last thing you need to know is that you can get to a shell from `less` and execute commands. The simplest way to do this is just to type `!` and press Enter. This launches a shell and leaves you free to type all the commands you want, as per normal. Type `exit` to return to `less`. You can also type specific commands by entering them after the exclamation mark, using the special character `%` for the current filename. For example, `du -h %` prints the size of the current file. Finally, you can use `!!` to repeat the previous command.

Creating Links Between Files with `ln`

Linux allows you to create links between files that look and work like normal files for the most part. Moreover, it allows you to make two types of links, known as *hard* links and *symbolic* links (*symlinks*). The difference between the two is crucial, although it might not be obvious at first!

Each filename on your system points to what is known as an *inode*, which is the absolute location of a file. Linux allows you to point more than one filename to a given inode, and the result is a hard link—two filenames pointing to exactly the same file. Each of these files shares the same contents and attributes. So, if you edit one, the other changes because they are both the same file.

On the other hand, a symlink—sometimes called a *soft* link—is a redirect to the real file. When a program tries to read from a symlink, it automatically is redirected to what the symlink is pointing at. The fact that symlinks are really just dumb pointers has two advantages: You can link to something that does not exist (and create it later if you want), and you can link to directories.

Both types of links have their uses. Creating a hard link is a great way to back up a file on the same disk. For example, if you delete the file in one location, it still exists untouched in the other location. Symlinks are popular because they allow a file to appear to be in a different location; you could store your website in `/var/www/live` and an under-construction holding page in `/var/www/construction`. Then you could have Apache point to a symlink `/var/www/html` that is redirected to either the live or construction directory depending on what you need.

TIP

The `shred` command overwrites a file's contents with random data, allowing for safe deletion. Because this directly affects a file's contents, rather than just a filename, this means that all filenames hard linked to an inode are affected.

Both types of link are created using the `ln` command. By default, it creates hard links, but you can create symlinks by passing it the `-s` parameter. The syntax is `ln [-s] <something> <somewhere>`, for example:

```
ln -s myfile.txt mylink
```

That creates the symlink `mylink` that points to `myfile.txt`. Remove the `-s` to create a hard link. You can verify that your link has been created by running `ls -l`. Your symlink should look something like this:

```
lrwxrwxrwx  1 paul paul    5 Feb 19 12:39 mylink -> myfile.txt
```

Note how the file properties start with `l` (lowercase *L*) for link and how `ls -l` also prints where the link is going. Symlinks are always very small in size; the previous link is 5 bytes in size. If you created a hard link, it should look like this:

```
-rw-rw-r   2 paul paul   341 Feb 19 12:39 mylink
```

This time the file has normal attributes, but the second number is 2 rather than 1. That number is how many hard links point to this file, which is why it is 2 now. The file size is also the same as that of the previous filename because it *is* the file as opposed to just being a pointer.

Symlinks are used extensively in Linux. Programs that have been superseded, such as `sh`, now point to their replacements (in this case `bash`), and library versioning is accomplished through symlinks. For example, applications that link against `zlib` load `/usr/lib/libz.so`. Internally, however, that is just a symlink that points to the actual `zlib` library: `/usr/lib/libz.so.1.2.1.2`. This enables multiple versions of libraries to be installed without applications needing to worry.

Finding Files from an Index with `locate`

When you use the `find` command, it searches recursively through each directory each time you request a file. This is slow, as you can imagine. Fortunately, Ubuntu ships with a `cron` job that creates an index of all the files on your system every night. Searching this index is extremely fast, which means that, if the file you are looking for has been around since the last index, this is the preferable way of searching.

To look for a file in your index, use the command `locate` followed by the names of the files you want to find, like this:

```
locate myfile.txt
```

On a relatively modern computer (1.5GHz or higher), `locate` should be able to return all the matching files in under a second. The trade-off for this speed is lack of flexibility. You can search for matching filenames, but, unlike `find`, you cannot search for sizes, owners, access permissions, or other attributes. The one thing you can change is case sensitivity; use the `-i` parameter to do a case-insensitive search.

Although Ubuntu rebuilds the filename index nightly, you can force a rebuild whenever you want by running the command `updatedb` with sudo. This usually takes a few minutes, but when it's done the new database is immediately available.

Listing Files in the Current Directory with `ls`

The `ls` command, like `ln`, is one of those you expect to be very straightforward. It lists files, but how many options can it possibly have? In true Linux style, the answer is many, although again you need only know a few to wield great power!

The basic usage is simply `ls`, which out the files and directories in the current location. You can filter that using normal wildcards, so all these are valid:

```
ls *
ls *.txt
ls my*ls *.txt *.xml
```

Any directories that match these filters are recursed into one level. That is, if you run `ls my*` and you have the files `myfile1.txt` and `myfile2.txt` and a directory `mystuff`, the matching files are printed first. Then `ls` prints the contents of the `mystuff` directory.

The most popular parameters for customizing the output of `ls` are

- `-a`—Includes hidden files.
- `-h`—Uses human-readable sizes.
- `-l`—A lowercase *L*, it enables long listing.
- `-r`—Reverse order.
- `-R`—Recursively lists directories.
- `-s`—Shows sizes.
- `--sort`—Sorts the listing.

All files that start with a period are hidden in Linux, so that includes the `.gnome` directory in your home directory, as well as `.bash_history` and the `.` and `..` implicit directories that signify the current directory and the parent. By default, `ls` does not show these files, but if you run `ls -a`, they are shown. You can also use `ls -A` to show all the hidden files except `.` and `..`.

The `-h` parameter needs to be combined with the `-s` parameter, like this:

```
ls -sh *.txt
```

That outputs the size of each matching file in a human-readable format, such as 108KB or 4.5MB.

Using the `-l` parameter enables much more information about your files. Instead of just providing the names of the files, you get output like this:

```
drwxrwxr-x 24 paul paul  4096 Dec 24 21:33 arch
-rw-r--r--  1 paul paul 18691 Dec 24 21:34 COPYING
-rw-r--r--  1 paul paul 88167 Dec 24 21:35 CREDITS
drwxrwxr-x  2 paul paul  4096 Dec 24 21:35 crypto
```

That shows four matches and prints a lot of information about each of them. The first row shows the `arch` directory; you can tell it is a directory because its file attributes starts with a d. The `rwxrwxr-x` following that shows the access permissions, and this has special meanings because it is a directory. Read access for a directory allows users to see the directory contents, write access allows you to create files and subdirectories, and execute access allows you to `cd` into the directory. If a user has execute access but not read access, he will be able to `cd` into the directory but not list files.

Moving on, the next number on the line is 24, which also has a special meaning for directories: It is the number of subdirectories, including `.` and `..`. After that is `paul paul`, which is the name of the user owner and the group owner for the directory. Next is the size and modification time, and finally the directory name itself.

The next line shows the file `COPYING`, and most of the numbers have the same meaning, with the exception of the 1 immediately after the access permissions. For directories, this is the number of subdirectories, but for files this is the number of hard links to this file. A `1` in this column means this is the only filename pointing to this inode, so if you delete it, it is gone.

Ubuntu comes configured with a shortcut command for `ls -l`: `ll`.

The `--sort` parameter allows you to reorder the output from the default alphabetical sorting. You can sort by various things, although the most popular are extension (alphabetically), size (largest first), and time (newest first). To flip the sorting (making size sort by smallest first), use the `-r` parameter also. So, this command lists all `.ogg` files, sorted smallest to largest:

```
ls --sort size -r *.ogg
```

Finally, the `-R` parameter recurses through subdirectories. For example, typing `ls /etc` lists all the files and subdirectories in `/etc`, but `ls -R /etc` lists all the files in and subdirectories in `/etc`, all the files and subdirectories in `/etc/acpi`, all the files and subdirectories in `/etc/acpi/actions`, and so on until every subdirectory has been listed.

Reading Manual Pages with `man`

Time for a much-needed mental break: The `man` command is easy to use. Most people use only the topic argument, like this: `man gcc`. However, two other commands work closely with `man` that are very useful—`whatis` and `apropos`.

The `whatis` command returns a one-line description of another command, which is the same text you get at the top of that command's `man` page. For example:

```
[paul@caitlin ~]$ whatis ls
ls    (1) - list directory contents
```

The output there explains what `ls` does but also provides the `man` page section number for the command so you can query it further.

On the other hand, the apropos command takes a search string as its parameter and returns all `man` pages that match the search. For example, `apropos mixer` gives this list:

```
alsamixer (1) - soundcard mixer for ALSA soundcard driver
mixer (1) - command-line mixer for ALSA soundcard driver
aumix (1) - adjust audio mixer
```

So, use `apropos` to help you find commands and use `whatis` to tell you what a command does.

Making Directories with `mkdir`

Making directories is as easy as it sounds, although there is one parameter you should be aware of: `-p`. If you are in `/home/paul` and you want to create the directory `/home/paul/audio/sound`, you will get an error like this:

```
mkdir: cannot create directory `audio/sound`: No such file or directory
```

At first glance this seems wrong because `mkdir` cannot create the directory because it does not exist. What it actually means is that it cannot create the directory `sound` because the directory `audio` does not exist. This is where the `-p` parameter comes in: If you try to make a directory within another directory that does not exist, like the previous, `-p` creates the parent directories, too. So:

```
mkdir -p audio/sound
```

That first creates the audio directory and then creates the sound directory inside it.

Moving Files with `mv`

This command is one of the easiest around. There are two helpful parameters to `mv`: `-f`, which overwrites files without asking, and `-u`, which moves the source file only if it is newer than the destination file. That is it!

Listing Processes with `ps`

This is the third and last "command that should be simple, but isn't" that is discussed here. The `ps` command lists processes and gives you an extraordinary amount of control over its operation.

The first thing to know is that `ps` is typically used with what are known as BSD-style parameters. Back in the section "Finding Files by Searching with `find`," we discussed UNIX-STYLE, GNU-style, and X-style parameters (`-c`, `--dosomething`, and `-dosomething`, respectively), but BSD-style parameters are different because they use single letters without a dash.

So, the default use of `ps` lists all processes that you are running that are attached to the terminal. However, you can ask it to list all your processes attached to any terminal (or indeed no terminal) by adding the `x` parameter: `ps x`. You can ask it to list all processes for all users with the `a` parameter or combine that with `x` to list all processes for all users, attached to a terminal or otherwise: `ps ax`.

However, both of these are timid compared with the almighty `u` option, which enables user-oriented output. In practice, that makes a huge difference because you get important fields like the username of the owner, how much CPU time and RAM are being used, when the process was started, and more. This outputs a lot of information, so you might want to try adding the `f` parameter, which creates a process forest by using ASCII art to connect parent commands with their children. You can combine all the options so far with this command: `ps faux` (yes, with a little imagination you spell words with the parameters!).

You can control the order in which the data is returned by using the `--sort` parameter. This takes either a `+` or a `-` (although the `+` is default) followed by the field you want to sort by: `command`, `%cpu`, `pid`, and `user` are all popular options. If you use the minus sign, the results are reversed. This next command lists all processes, ordered by CPU usage descending:

```
ps aux --sort=-%cpu
```

There are many other parameters for `ps`, including a huge amount of options for compatibility with other UNIXes. If you have the time to read the man page, you should give it a try!

Deleting Files and Directories with `rm`

The `rm` command has only one parameter of interest: `--preserve-root`. By now, you should know that issuing `rm -rf /` with sudo will destroy your Linux installation because `-r` means *recursive* and `-f` means *force* (do not prompt for confirmation before deleting). It is possible for a clumsy person to issue this command by accident—not by typing the command on purpose, but by putting a space in the wrong place. For example:

```
rm -rf /home/paul
```

That command deletes the home directory of the user `paul`. This is not an uncommon command; after you have removed a user and backed up her data, you will probably want to issue something similar. However, if you add an accidental space between the `/` and the h in “home”, you get this:

```
rm -rf / home/paul
```

This time the command means “delete everything recursively from `/` and then delete `home/paul`”—quite a different result! You can stop this from happening by using the `--preserve-root` parameter, which stops you from catastrophe with this message:

```
rm: it is dangerous to operate recursively on `/'
rm: use --no-preserve-root to override this failsafe.
```

Of course, no one wants to keep typing `--preserve-root` each time they run `rm`, so you should add this line to the `.bashrc` file in your home directory:

```
alias rm='rm --preserve-root'
```

That alias automatically adds `--preserve-root` to all calls to `rm` in future Bash sessions.

Printing the Last Lines of a File with `tail`

If you want to watch a log file as it is written to, or want to monitor a user's actions as they are occurring, you need to be able to track log files as they change. In these situations you need the `tail` command, which prints the last few lines of a file and updates as new lines are added. This command tells tail to print the last few lines of `/var/log/httpd/access_log`, the Apache hit log:

```
tail /var/log/httpd/access_log
```

To get `tail` to remain running and update as the file changes, add the `-f` parameter (follow):

```
tail -f /var/log/httpd/access_log
```

You can tie the lifespan of a `tail` follow to the existence of a process by specifying the `--pid` parameter. When you do this, `tail` continues to follow the file you asked for until it sees that the process identified by PID is no longer running, at which point it stops tailing.

If you specify multiple files on the command line, `tail` follows both, printing file headers whenever the input source changes. Press Ctrl+C to terminate `tail` when in follow mode.

Printing Resource Usage with `top`

The `top` command is unusual in this list because the few parameters it takes are rarely, if ever, used. Instead, it has a number of commands you can use while it is running to customize the information it shows you. To get the most from these instructions, open two terminal windows. In the first, run the program `yes` and leave it running; in the second run `sudo top`.

The default sort order in `top` shows the most CPU-intensive tasks first. The first command there should be the `yes` process you just launched from the other terminal, but there should be many others also. First, we want to filter out all the other uses and focus on the user running `yes`. To do this, press `u` and enter the username you used when you ran `yes`. When you press Enter, top filters out processes not being run by that user.

The next step is to kill the process ID of the `yes` command, so you need to understand what each of the important fields means:

- `PID`—The process ID
- `User`—The owner of the process

- `PR`—Priority
- `NI`—Niceness
- `Virt`—Virtual image size in kilobytes
- `Res`—Resident size in kilobytes
- `Shr`—Shared memory size in kilobytes
- `S`—Status
- `%CPU`—CPU usage
- `%Mem`—Memory usage
- `Time+`—CPU time
- `Command`—The command being run

Several of them are unimportant unless you have a specific problem. The ones we are interested in are `PID`, `User`, `Niceness`, `%CPU`, `%MEM`, `Time+`, and `Command`. The `Niceness` of a process is how much time the CPU allocates to it compared to everything else on the system: 19 is the lowest, and –19 is the highest.

With the columns explained, you should be able to find the process ID of the errant `yes` command launched earlier; it is usually the first number below `PID`. Now type `k`, enter that process ID, and press Enter. You are prompted for a signal number (the manner in which you want the process killed), with 15 provided as the default. Signal 15 (also known as `SIGTERM`, for “terminate”) is a polite way of asking a process to shut down, and all processes that are not wildly out of control should respond to it. Give `top` a few seconds to update itself, and hopefully the `yes` command should be gone. If not, you need to be more forceful: type `k` again, enter the PID, and press Enter. When prompted for a signal to send, enter `9` and press Enter to send `SIGKILL`, or “terminate whether you like it or not.”

You can choose the fields to display by pressing `f`. A new screen appears that lists all possible fields, along with the letter you need to press to toggle their visibility. Selected fields are marked with an asterisk and have their letter, for example:

```
* A: PID        = Process Id
```

If you press the `a` key, the screen changes to this:

```
a: PID        = Process Id
```

When you have made your selections, press Enter to return to the normal `top` view with your normal column selection.

You can also press `F` to select the field you want to use for sorting. This works in the same way as the field selection screen, except that you can select only one field at a time.

Again, press Enter to get back to `top` after you have made your selection, and it will be updated with the new sorting.

If you press `B`, text bolding is enabled. By default, this bolds some of the header bar as well as any programs that are currently running (as opposed to sleeping), but if you press `x` you can also enable bolding of the sorted column. You can use `y` to toggle bolding of running processes.

The last command to try is `r`, which allows you to renice—or adjust the nice value—of a process. You need to enter the PID of the process, press Enter, and enter a new nice value. Keep in mind that 19 is the lowest and –20 is the highest; anything less than 0 is considered "high" and should be used sparingly.

Printing the Location of a Command with `which`

The purpose of `which` is to tell you the exact command that would be executed if you typed it. For example, `which mkdir` returns `/bin/mkdir`, telling you that running the command `mkdir` runs `/bin/mkdir`.

Combining Commands

So far, we have only been using commands individually, and for the large part that is what you will be doing in practice. However, some of the real power of these commands lies in the ability to join them together to get exactly what you want. There are some extra little commands that we have not looked at that are often used as glue because they do one very simple thing that enables a more complex process to work.

All the commands we have looked at have printed their information to the screen, but this is often flexible. There are two ways to control where output should go: piping and output redirection. A *pipe* is a connector between one command's output and another's input. Instead of sending its output to your terminal, a command sends that output directly to another command for input. Output redirection works in a similar way to pipes but is usually used for files. We will look at pipes first and then output redirection.

Two of the commands we have looked at so far are `ps` and `grep`: the process lister and the string matcher. We can combine the two to find out which users are playing Nethack right now:

```
ps aux ¦ grep nethack
```

That creates a list of all the processes running right now and sends that list to the `grep` command, which filters out all lines that do not contain the word `nethack`. Ubuntu allows you to pipe as many commands as you can sanely string together. For example, we could add in the `wc` command, which counts the numbers of lines, words, and characters in its input, to count precisely how many times Nethack is being run:

```
ps aux ¦ grep nethack ¦ wc -l
```

The `-l` parameter to wc (lowercase *L*) prints only the line count.

Using pipes in this way is often preferable to using the `-exec` parameter to find, simply because many people consider find to be a black art and so anything that uses it less frequently is better! This is where the `xargs` command comes in: It converts output from one command into arguments for another.

For a good example, consider this mammoth find command from earlier:

```
find / -name "*.txt" -size +10k -user paul -not -perm +o=r -exec chmod o+r {} \;
```

That searches every directory from `/` onward for files matching *.txt that are greater than 10KB, are owned by user `paul`, and do not have read permission for others. Then it executes `chmod` on each of the files. It is a complex command, and people who are not familiar with the workings of find might have problems understanding it. So, what we can do is break it up into two—a call to `find` and a call to `xargs`. The most conversion would look like this:

```
find / -name "*.txt" -size +10k -user paul -not -perm +o=r ¦ xargs chmod o+r
```

That has eliminated the confusing `{} \;` from the end of the command, but it does the same thing, and faster, too. The speed difference between the two is because using `-exec` with find causes it to execute `chmod` once for each file. However, `chmod` accepts many files at a time and, because we are using the same parameter each time, we should take advantage of that. The second command, using `xargs`, is called once with all the output from find, and so saves many command calls. The `xargs` command automatically places the input at the end of the line, so the previous command might look something like this:

```
xargs chmod o+r file1.txt file2.txt file3.txt
```

Not every command accepts multiple files, though, and if you specify the `-l` parameter, `xargs` executes its command once for each line in its input. If you want to check what it is doing, use the `-p` parameter to have `xargs` prompt you before executing each command.

For even more control, the `-i` parameter allows you to specify exactly where the matching lines should be placed in your command. This is important if you need the lines to appear before the end of the command or need it to appear more than once. Either way, using the `-i` parameter also enables the `-l` parameter so each line is sent to the command individually. This next command finds all files in `/home/paul` that are larger than 10,000KB in size (10MB) and copies them to `/home/paul/archive`:

```
find /home/paul -size +10000k ¦ xargs -i cp {} ./home/paul/archive
```

Using find with `xargs` is a unique case. All too often, people use pipes when parameters would do the job just as well. For example, these two commands are identical:

```
ps aux --sort=-%cpu ¦ grep -v `whoami`
ps -N ux --sort=-%cpu
```

The former prints all users and processes and then pipes that to `grep`, which in turn filters out all lines that contain the output from the program `whoami` (our username). So, line one prints all processes being run by other uses, sorted by CPU use. Line two does not specify the `a` parameter to `ps`, which makes it list only our parameters. It then uses the `-N` parameter to flip that, which means it is everyone but us, without the need for `grep`.

The reason people use the former is often just simplicity: Many people know only a handful of parameters to each command, so they can string together two commands simply rather than write one command properly. Unless the command is to be run regularly, this is not a problem. Indeed, the first line would be better because it does not drive people to the manual to find out what `ps -N` does!

Multiple Terminals

It is both curious and sad that many Linux veterans have not heard of the `screen` command. Curious because they needlessly go to extra effort to replicate what `screen` takes in its stride and sad because they are missing a powerful tool that would benefit them greatly.

Picture this scene: You connect to a server via SSH and are working at the remote shell. You need to open another shell window so you can have the two running side by side; perhaps you want the output from top in one window while typing in another. What do you do? Most people would open another SSH connection, but that is both wasteful and unnecessary. `screen` is a *terminal multiplexer*, which is a fancy term for a program that lets you run multiple terminals inside one terminal.

The best way to learn `screen` is to try it yourself, so open a console, type **`screen`**, and then press Enter. Your display will blank momentarily and then be replaced with a console; it will look like nothing has changed. Now, let's do something with that terminal. Run `top` and leave it running for the time being. Hold down the Ctrl key and press `a` (referred to as Ctrl+a from now on); then let go of them both and press `c`. Your prompt will clear again, leaving you able to type. Run the `uptime` command.

Pop quiz: What happened to the old terminal running `top`? It is still running, of course. You can type Ctrl+a and then press 0 to return to it. Type Ctrl+a and then press 1 to go back to your `uptime` terminal. While you are viewing other terminals, the commands in the other terminals carry on running as normal so you can multitask.

Many of `screen`'s commands are case sensitive, so the lettering used here is very specific: Ctrl+a means "press Ctrl and the a key," but Ctrl+A means "press Ctrl and Shift and the a key" so you get a capital A. Ctrl+a+A means "press Ctrl and the a key, let them go, and then press Shift and the a key."

You have seen how to create new displays and how to switch between them by number. However, you can also bring up a window list and select windows using your cursor with Ctrl+a+" (that is, press Ctrl and a together, let go, and press the double quotes key [usually Shift and the single quote key]). You will find that the screens you create have the name `bash` by default, which is not very descriptive. Select a window and press

Ctrl+a+A. You are prompted to enter a name for the current window, and your name is used in the window list.

Once you get past window 9, it becomes impossible to switch to windows using Ctrl+a and 0–9; as soon as you type the 1 of 10, `screen` switches to display 1. The solution is to use either the window list or the quick change option, in which you press Ctrl+a+' (single quote), enter either the screen number or the name you gave it, then press Enter. You can also change back to the previous window by pressing Ctrl+a+Ctrl+a. If you only work within a small set of windows, you can use Ctrl+a+n and Ctrl+a+p to move to the next and previous windows, respectively. Of course, if you are changing to and from windows only to see whether something has changed, you are wasting time because `screen` can monitor windows for you and report if anything changes. To enable (or disable) monitoring for a window, use Ctrl+a+M; when something happens, `screen` flashes a message. If you miss it (the messages disappear when you type something), use Ctrl+a+m to bring up the last message.

Windows close when you kill the main program inside. Using Ctrl+a+c, this window is Bash; type **`exit`** to quit. Alternatively, you can use Ctrl+a+K to kill a window. When all your windows are closed, `screen` terminates and prints a `screen is terminating` message so you know you are out.

However, there are two alternatives to quitting: locking and disconnecting. The first, activated with Ctrl+a+x, locks access to your screen data until you enter your system password. The second is the most powerful feature of `screen`: You can exit it and do other things for a while and then reconnect later and `screen` will pick up where you left off. For example, you could be typing at your desk, disconnect from `screen`, then go home, reconnect, and carry on as if nothing had changed. What's more, all the programs you ran from `screen` carry on running even while `screen` is disconnected. It even automatically disconnects for you if someone closes your terminal window while it is in a locked state (with Ctrl+a+x).

To disconnect from `screen`, press Ctrl+a+d. You are returned to the prompt from which you launched `screen` and can carry on working, close the terminal you had opened, or even log out completely. When you want to reconnect, run the command `screen -r`. You can, in the meantime, just run `screen` and start a new session without resuming the previous one, but that is not wise if you value your sanity! You can disconnect and reconnect the same `screen` session as many times you want, which potentially means you need never lose your session again.

Although this has been a mere taster of screen, hopefully you can see how useful it can be.

Reference

- http://www.gnu.org/—The website of the GNU project, it contains manuals and downloads for lots of command-line software.
- http://www.linuxdevcenter.com/linux/cmd/—A wide selection of Linux commands and explanations of what they do.
- http://www.tuxfiles.org/linuxhelp/cli.html—Several short command-line tutorials, courtesy of tuXfiles.
- http://www.linuxcommand.org/—Describes itself as "your one-stop command line shop!" It contains a wealth of useful information about the console.

Books

Understanding the way UNIX works at the nuts and bolts level can be both challenging and rewarding, and there are several good books that will help guide you on your way. Perhaps the best is *The Art of Unix Programming* by Eric Raymond (Addison-Wesley, ISBN: 0-13-142901-9), which focuses on the philosophy behind UNIX and manages to mix in much about the command line.

O'Reilly's *Unix CD Bookshelf* (ISBN: 0-596-00392-7) contains seven highly respected books in one, although it retails for more than $120 as a result! That said, it is incomparable in its depth and breadth of coverage.

This page intentionally left blank

CHAPTER 31
Managing Software

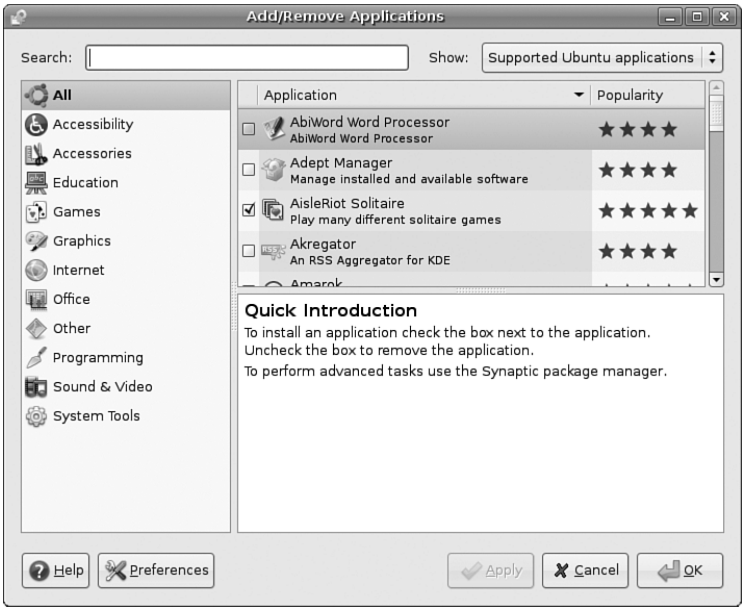
IN THIS CHAPTER

In this chapter, we look at the options you have to manage your software in Ubuntu. If you are used to a Windows environment where you are reliant on visiting several different vendor websites to download updates, you are in for a shock! Updating a full Ubuntu installation, including all the application software, is as easy as running the Update Manager program. You will discover just how easy it is to install and even remove various software packages.

Ubuntu provides a variety of tools for system resource management. The following sections introduce the graphical software management tools that you will use for most of your software management. (You will learn more about a collection of command-line tools in Chapter 30, "Command Line Masterclass.") This chapter also covers monitoring and managing memory and disk storage on your system.

Using Add/Remove Applications for Software Management

The Ubuntu graphical package-management tool identified in the Applications menu as Add/Remove is actually an application named `gnome-app-install`. Add/Remove Applications enables you to select packages arranged in categories and install or remove them. When you launch Add/Remove Applications, you are not prompted to enter your password; this is only required when making changes. You should see the Package Browsing screen, as shown in Figure 31.1.



Quick Introduction
To install an application check the box next to the application.
Uncheck the box to remove the application.
To perform advanced tasks use the Synaptic package manager.

FIGURE 31.1 The initial Add/Remove Applications screen allows you to browse through packages sorted by groups.

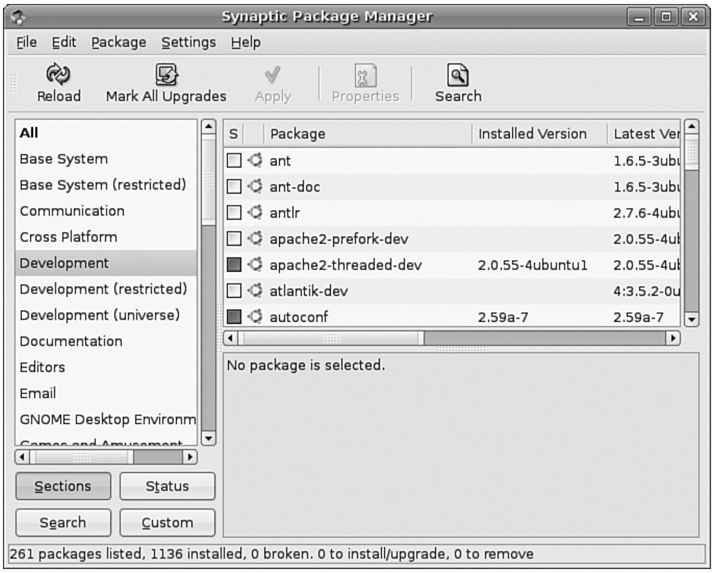
Along the left side of the screen, you will see the broad list of categories into which the applications have been divided. At the top, selected by default, is the All category, which lists every package that can be installed. The right side of the screen is split into two, with the upper portion providing the application list and the lower portion describing the currently selected application. Just above the application list are options for searching and filtering, with the default filter set to Supported Ubuntu Applications. You can change that if you want to, particularly if you want to try out all the software we discuss in this book.

Installing some new software is as simple as finding it in the package list and checking its box. After you have selected all the applications you want, click Apply. You will be prompted to enter your password so that Ubuntu can install the software. Currently installed applications are already checked, and you can remove them by unchecking them and clicking Apply.

To search for a specific application in the list, type something into the Search box at the top. Note that this searches within the current category; so if you are in the Games category and search for “office,” you will get no results. The best place to search is within the All category, to make sure you search all areas.

Using Synaptic for Software Management

The Add/Remove Applications dialog works just fine for adding applications, but if you need to install something very specific—such as a library—or if you want to reconfigure your installation system, you need to use Synaptic (see Figure 31.2). You can run Synaptic by selecting it from the System, Administration, Synaptic Package Manager menu option.



S	Package	Installed Version	Latest Ver
☐	ant		1.6.5-3ubu
☐	ant-doc		1.6.5-3ubu
☐	antlr		2.7.6-4ubu
☐	apache2-prefork-dev		2.0.55-4ul
■	apache2-threaded-dev	2.0.55-4ubuntu1	2.0.55-4ul
☐	atlantik-dev		4:3.5.2-0u
■	autoconf	2.59a-7	2.59a-7

No package is selected.

261 packages listed, 1136 installed, 0 broken. 0 to install/upgrade, 0 to remove

FIGURE 31.2 For more advanced software management, Synaptic is the preferred tool.

At first glance, Synaptic looks a little like the Add/Remove Applications window. Along the left are software categories (although this time there are more of them), along the top right are the package selections for that category, and on the bottom right is the Package Information window that shows information about the currently selected package. To install or remove software, click on the check box to the left of its name, and you'll see a menu appear offering the following options:

- **Unmark**—If you have marked this package for installation, upgrade, or one of the other options, this removes that mark.
- **Mark for Installation**—Add this packages to the list that will be installed.
- **Mark for Re-installation**—If you have some software already installed, but for some reason it's not working, this reinstalls it from scratch.
- **Mark for Upgrade**—If the software has updates available, this will download and install them.
- **Mark for Removal**—This deletes the selected package from your system, but leaves its configuration files intact so that if you ever reinstall it you do not have to reconfigure it.
- **Mark for Complete Removal**—This deletes the selected package from your system, but also removes any configuration files, purging everything from the system.

After you have made your changes, click the Apply button to have Synaptic download, install, upgrade, and uninstall as necessary. If you close the program without clicking Apply, your changes will be lost.

Beneath the categories on the left side of the screen, you will see four buttons: Sections, Status, Search, and Custom; with Sections selected. These customize the left list: Sections is the Categories view, Status lets you view packages that are installed or upgradable, Search stores results of your searches, and Custom has some esoteric groupings that are only really useful to advanced users.

You can press Ctrl+F at any time to search for a particular package. By default it is set to search by package name, but it is quite common to change the Look In box to be "Description and Name". As mentioned already, your search terms are saved under the Search view (the button on the bottom left), and you can just click from that list to re-search on that term.

As well as providing the method of installing and removing software, Synaptic provides the means to configure the servers you want to use for finding packages. In fact, this is where you can make one of the most important changes to your Ubuntu system: You can open it up to the Debian universe and multiverse.

Ubuntu is based on the Debian distribution, which has more than 15,000 software packages available for installation. Ubuntu uses only a small subset of that number, but makes it easy for you to enable the others. When you use Synaptic, you will see small orange Ubuntu logos next to every package; this identifies them as being officially supported by the Ubuntu developers. The other, non-Ubuntu packages you can enable will not have this logo, but they are still supported by the Debian developers.

To enable the Universe and Multiverse repositories, go to Settings, Repositories. This list shows all the servers you have configured for software installation and updates, and will include the Universe and Multiverse repositories in there. When you find them, check them and then click Close.

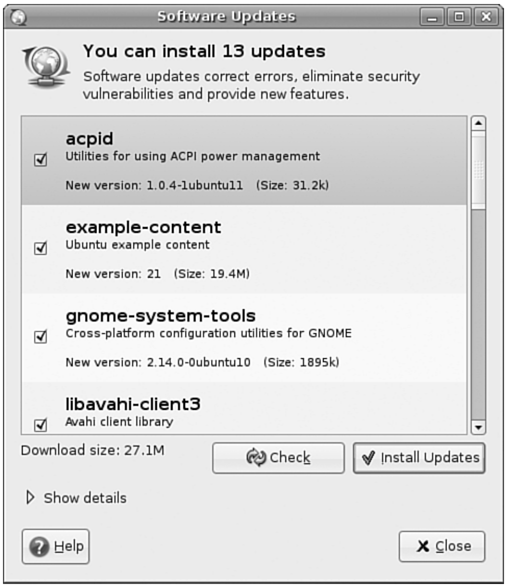
Synaptic shows a message box warning you that the repository listings have changed and that you need to click the Reload button (near the top left of the Synaptic window) to have it refresh the package lists. Go ahead and do that, and you should see a lot more software appear for your selection. However, notice that only a small number have the official Ubuntu "seal" attached, which means you need to be a bit more careful when installing software!

NOTE

Much of the software discussed in this book is only available through the Universe repository. As such, we highly recommend enabling it to get full use out of this book and your Ubuntu installation.

Staying Up-to-Date

Although you can manage your software updates through Synaptic, Ubuntu provides a dedicated tool in the form of Update Manager (launched through System, Administration, Update Manager, and shown in Figure 31.3). This tool is purposefully designed to be simple to use: when you run it, Update Manager automatically downloads the list of updates available for you and checks them all in the list it shows. If the update list was downloaded automatically not too long ago, you can force Ubuntu to refresh the list of available updates by clicking the Check button. Otherwise, all you need to do is click Install Updates and your system will be brought up-to-date. If you want a little more information about the updates, click Show Details at the bottom to see what has changed in the update.



You can install 13 updates

Software updates correct errors, eliminate security vulnerabilities and provide new features.

acpid
Utilities for using ACPI power management
New version: 1.0.4-1ubuntu11 (Size: 31.2k)

example-content
Ubuntu example content
New version: 21 (Size: 19.4M)

gnome-system-tools
Cross-platform configuration utilities for GNOME
New version: 2.14.0-0ubuntu10 (Size: 1895k)

libavahi-client3
Avahi client library

Download size: 27.1M

FIGURE 31.3 If you just need to update your software to apply bug fixes and security upgrades, use Update Manager.

Ubuntu automatically checks for updates periodically and notifies you when critical updates are available. That said, there's no harm running Update Manager yourself every so often, just to make sure; it's better to be safe than sorry.

Working on the Command Line

With so much software available for install, it is no surprise that Debian-based distros have many ways to manage software installation. At their root, however, they all use Debian's world-renowned Advanced Package Tool. One poster on Slashdot once said, "Welcome to Slashdot. If you can't think of anything original, just say how much APT rocks and you'll fit right in." You see, even though many other distros have tried to equal the power of APT, nothing else even comes close.

Why is APT so cool? Well, it was the first system to properly handle dependencies in software. Other distros, such as Red Hat, used RPM files that had dependencies. For example, an RPM for the Gimp would have a dependency on Gtk, the graphical toolkit on which the Gimp is based. As a result, if you tried to install your Gimp RPM without having the Gtk RPM, your install would fail. So you grab the Gtk RPM and try again. Aha: Gtk has a dependency on three other things that you need to download. And those three other things have dependencies on 20 other things. And so on, and so on, usually until you can't actually find a working RPM for one of the dependencies, and you give up.

APT, on the other hand, was designed to automatically find and download dependencies for your packages. So if you wanted to install the Gimp, it would download the Gimp's package as well as any other software it needed to work. No more hunting around by hand, no more worrying about finding the right version, and certainly no more need to compile things by hand. APT also handles installation resuming, which means that if you lose your Internet connection part-way through an upgrade (or your battery runs out, or you have to quit, or whatever), APT will just pick up where it left off next time you rerun it.

Day-to-Day Usage

To enable you to search for packages both quickly and thoroughly, APT uses a local cache of the available packages. Try running this command:

```
sudo apt-get update
```

The `apt-get update` command instructs APT to contact all the servers it is configured to use and download the latest list of file updates. If your lists are outdated, it takes a minute or two for APT to download the updates. Otherwise, this command executes it in a couple of seconds.

After the latest package information has been downloaded, you are returned to the command line. You can now ask APT to automatically download any software that has been updated, using this command:

```
sudo apt-get upgrade
```

If you have a lot of software installed on your machine, there is a greater chance of things being updated. APT scans your software and compares it to the latest package information from the servers and produces a report something like this:

```
paul@ubuntu:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree... Done
The following packages will be upgraded:
  example-content gnome-system-tools libavahi-client3 libavahi-common-data
  libavahi-common3 libavahi-glib1 ttf-opensymbol ubuntu-artwork ubuntu-minimal
  ubuntu-standard update-notifier
11 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 27.0MB of archives.
After unpacking 147kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

Each part of that report tells us something important. Starting at the top, the line "the following packages will be upgraded" gives us the exact list of packages for which updates are available. If you're installing new software or removing software, you'll see lists titled "The following packages will be installed" and "The following packages will be removed." A summary at the end shows that there's a total of 11 packages that APT will upgrade, with 0 new packages, 0 to remove, and 0 not upgraded. Because this is an upgrade rather than an installation of new software, all those new packages will only take up 147KB of additional space. Although they come to a 27MB download, they are overwriting existing files.

It's important to understand that a basic `apt-get upgrade` will never remove software or add new software. As a result, it is safe to use to keep your system fully patched, because it should never break things. However, occasionally you will see the "0 not upgraded" status change, which means some things cannot be upgraded. This happens when some software must be installed or removed to satisfy the dependencies of the updated package, which, as previously mentioned, `apt-get upgrade` will never do.

In this situation, you need to use `apt-get dist-upgrade`, so named because it's designed to allow users to upgrade from one version of Debian/Ubuntu to a newer version—an upgrade that inevitably involves changing just about everything on the system, removing obsolete software, and installing the latest features. This is one of the most-loved features of Debian because it allows you to move from version to version without having to download and install new CDs.

Whereas `apt-get upgrade` and `apt-get dist-upgrade` are there for upgrading packages, `apt-get install` is responsible for adding new software. For example, if you want to install the MySQL database server, you would run this:

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

Internally, APT would query "mysql-server" against its list of software, and find that it matches the mysql-server-5.0 package. It would then find which dependencies it needs that you don't already have installed and then give you a report like this one:

```
paul@ubuntu:~$ sudo apt-get install mysql-server
Reading package lists... Done
Building dependency tree... Done
The following extra packages will be installed:
  libdbd-mysql-perl libdbi-perl libnet-daemon-perl libplrpc-perl
  mysql-client-5.0 mysql-server-5.0
Suggested packages:
  dbishell libcompress-zlib-perl
Recommended packages:
  mailx
The following NEW packages will be installed
  libdbd-mysql-perl libdbi-perl libnet-daemon-perl libplrpc-perl
  mysql-client-5.0 mysql-server mysql-server-5.0
```

```
0 upgraded, 7 newly installed, 0 to remove and 11 not upgraded.
Need to get 28.5MB of archives.
After unpacking 65.8MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

This time you can see that APT has picked up and selected all the dependencies required to install MySQL Server 5.0, but it has also listed one recommended package and two suggested packages that it has not selected for installation. The "recommended" package is just that: The person who made the MySQL package (or its dependencies) thinks it would be a smart idea for you to also have the mailx package. If you want to add it, press N to terminate `apt-get` and rerun it like this:

```
apt-get install mysql-server mailx
```

The "suggested" packages are merely a lower form of recommendation. They don't add any crucial features to the software you selected for install, but it's possible that you might need them for certain, smaller things.

NOTE

APT maintains a package cache where it stores `.deb` files it has downloaded and installed. This usually lives in `/var/cache/apt/archives`, and can sometimes take up many hundreds of megabytes on your computer. You can have APT clean out the package cache by running `apt-get clean`, which deletes all the cached `.deb` files. Alternatively, you can run `apt-get autoclean`, which deletes cached `.deb` files that are beyond a certain age, thereby keeping newer packages.

If you try running `apt-get install` with packages you already have installed, APT will consider your command to be `apt-get update` and see whether new versions are available for download.

The last day-to-day package operation is, of course, removing things you no longer want. This is done through the `apt-get remove` command, as follows:

```
apt-get remove firefox
```

Removing packages can be dangerous, because APT also removes any software that relies on the packages you selected. For example, if you were to run `apt-get remove` libgtk2.0-0 (the main graphical toolkit for Ubuntu), you would probably find that APT insists on removing more than a hundred other things. The moral of the story is this: When you remove software, read the APT report carefully before pressing Y to continue with the uninstall.

A straight `apt-get remove` leaves behind the configuration files of your program so that if you ever reinstall it you do not also need to reconfigure it. If you want the configuration files removed as well as the program files, run this command instead:

```
apt-get remove --purge firefox
```

That performs a full uninstall.

NOTE

You can see a more extensive list of `apt-get` parameters by running `apt-get` without any parameters. The cryptic line at the bottom, “This APT has Super Cow Powers,” is made even more cryptic if you run the command `apt-get moo`.

Finding Software

With so many packages available, it can be hard to find the exact thing you need using command-line APT. The general search tool is called `apt-cache` and is used like this:

```
apt-cache search kde
```

Depending on which repositories you have enabled, that will return about a thousand packages. Many of those results will not even have KDE in the package name, but will be matched because the description contains the word *KDE*.

You can filter through this information in several ways. First, you can instruct `apt-cache` to search only in the package names, not in their descriptions. This is done with the `–n` parameter, like this:

```
apt-cache –n search kde
```

Now the search has gone down from 1,006 packages to 377 packages.

Another way to limit search results is to use some basic regular expressions, such as `^`, meaning “start,” and `$`, meaning “end.” For example, you might want to search for programs that are part of the main KDE suite and not libraries (usually named something like libkde), additional bits (such as xmms-kde), and things that are actually nothing to do with KDE yet still match our search (like tkdesk). This can be done by searching for packages that have a name starting with kde, as follows:

```
apt-cache –n search ^kde.
```

Perhaps the easiest way to find packages is to combine `apt-cache` with `grep`, to search within search results. For example, if you want to find all games-related packages for KDE, you could run this search:

```
apt-cache search games ¦ grep kde
```

When you’ve found the package you want to install, just run them through `apt-get install` as per usual. If you first want a little more information about that package, you can use `apt-cache showpkg`, like this:

```
apt-cache showpkg mysql-server-5.0
```

This shows information on “reverse depends” (which packages require, recommend, or suggest mysql-server-5.0), “dependencies” (which packages are required, recommended, or suggested to install mysql-server-5.0) and “provides” (which functions this package gives you). The "provides" list is quite powerful because it allows several different packages to provide a given resource. For example, a MySQL database-based program requires MySQL to be installed, but isn't fussy whether you install MySQL 4.1 or MySQL 5.0. In this situation, the Debian packages for MySQL 4.1 and MySQL 5.0 will both have "mysql-server-4.1" in the provides list, meaning that they offer the functionality provided by MySQL 4.1. As a result, you can install either version to satisfy the MySQL-based application.

Compiling Software from Source

Compiling applications from source is not that difficult. Most source code is available as compressed source *tarballs*—that is, `tar` files that have been compressed using `gzip` or `bzip`. The compressed files typically uncompress into a directory containing several files. It is always a good idea to compile source code as a regular user to limit any damage that broken or malicious code might inflict, so create a directory named `source` in your home directory.

From wherever you downloaded the source tarball, uncompress it into the `~/source` directory using the `-C` option to `tar`:

```
tar zxvf packagename.tgz -C  ~/source

tar zxvf packagename.tar.gz -C  ~/source

tar jxvf packagename.bz -C  ~/source

tar jxvf packagename.tar.bz2 -C  ~/source
```

If you are not certain what file compression method was used, use the `file` command to figure it out:

```
file packagename
```

Now, change directories to `~/source/`*`packagename`* and look for a file named `README`, `INSTALL`, or a similar name. Print out the file if necessary because it contains specific instructions on how to compile and install the software. Typically, the procedure to compile source code is as follows:

```
./configure
```

This runs a script to check whether all dependencies are met and the build environment is correct. If you are missing dependencies, the configure script normally tells you exactly which ones it needs. If you have the Universe and Multiverse repositories enabled in Synaptic, chances are you will find the missing software (usually libraries) in there.

When your configure script succeeds, run

```
make
```

to compile the software. And finally, run

```
sudo make install.
```

If the compile fails, check the error messages for the reason and run

```
make clean
```

before you start again. You can also run

```
sudo make uninstall
```

to remove the software if you do not like it.

Relevant Ubuntu and Linux Commands

You will use these commands while managing your Ubuntu system resources and software packages:

`apt-get`—The default command-line package-management tool (see Chapter 30 for more on APT)

`gnome-app-install`—The Ubuntu GUI Package Manager

`synaptic`—Advanced tool for software manipulation and repository editing

`update-manager`—The primary source of Ubuntu package updates

Reference

- http://www.debian.org/doc/manuals/project-history/ch-detailed.en.html—History of the Debian Linux package system.
- http://www.nongnu.org/synaptic/—Home of the Synaptic package manager.
- http://www.ubuntu.com/usn—The official list of Ubuntu security notices.

This page intentionally left blank

CHAPTER 32

Kernel and Module Management

IN THIS CHAPTER

A kernel is a complex piece of software that manages the processes and process interactions that take place within an operating system. As a user, you rarely, if ever, interact directly with it. Instead, you work with the applications that the kernel manages.

The Linux kernel is Linux. It is the result of years of cooperative (and sometimes contentious) work by numerous people around the world. There is only one common kernel source tree, but each major Linux distribution massages and patches its version slightly to add features, performance, or options. Each Linux distribution, including Ubuntu, comes with its own precompiled kernel as well as the kernel source code, providing you with absolute authority over the Linux operating system. In this chapter, we will examine the kernel and learn what it does for us and for the operating system.

In this chapter, you also learn how to obtain the kernel sources, as well as how and when to patch the kernel. The chapter leads you through an expert's tour of the kernel architecture and teaches you essential steps in kernel configuration, how to build and install modules, and how to compile drivers in Ubuntu. This chapter also teaches you important aspects of working with GRUB, the default Ubuntu boot loader. Finally, the chapter's troubleshooting information will help you understand what to do when something goes wrong with your Linux kernel installation or compilation process. As disconcerting as these problems can seem, this chapter shows you some easy fixes for many kernel problems.

Most users find that the precompiled Ubuntu kernel suits their needs. At some point, you might need to recompile the kernel to support a specific piece of hardware or add a new feature to the operating system. If you have heard horror stories about the difficulties of recompiling the Linux kernel, you can relax; this chapter gives you all the information you need to understand when recompiling is necessary and how to painlessly work through the process.

The Linux Kernel

The Linux kernel is the management part of the operating system that many people call "Linux." Although many think of the entire distribution as Linux, the only piece that can correctly be called Linux is the kernel. Ubuntu, like many Linux distributions, includes a kernel packaged with add-on software that interacts with the kernel so that the user can interface with the system in a meaningful manner.

The system utilities and user programs enable computers to become valuable tools to a user.

The First Linux Kernel

In 1991, Linus Torvalds released version .99 of the Linux kernel as the result of his desire for a powerful, UNIX-like operating system for his Intel 80386 personal computer. Linus wrote the initial code necessary to create what is now known as the Linux kernel and combined it with Richard Stallman's GNU tools. Indeed, because many of the Linux basic system tools come from the GNU Project, many people refer to the operating system as GNU/Linux. Since then, Linux has benefited as thousands of contributors add their talents and time to the Linux project. Linus still maintains the kernel, deciding on what will and will not make it into the kernel as official releases, known to many as the "vanilla" or "Linus" Linux kernel.

The Linux Source Tree

The source code for the Linux kernel is kept in a group of directories called the kernel source tree. The structure of the kernel source tree is important because the process of compiling (building) the kernel is automated; it is controlled by scripts interpreted by the `make` application. These scripts, known as makefiles, expect to find the pieces of the kernel code in specific places or they will not work. You will learn how to use `make` to compile a kernel later in this chapter.

It is not necessary for the Linux kernel source code to be installed on your system for the system to run or for you to accomplish typical tasks such as email, web browsing, or word processing. It is necessary that the kernel sources be installed, however, if you want to compile a new kernel. In the next section, we will show you how to install the kernel source files and how to set up the special symbolic link required. That link is `/usr/src/linux-2.6`, and it is how we will refer to the directory of the kernel source tree as we examine the contents of the kernel source tree.

The `/usr/src/linux-2.6` directory contains the `.config` and the `Makefile` files among others. The `.config` file is the configuration of your Linux kernel as it was compiled.

There is no `.config` file by default; you must select one from the `/configs` subdirectory. There, you will find configuration files for each flavor of the kernel Ubuntu provides; simply copy the one appropriate for your system to the default directory and rename it `.config`.

We have already discussed the contents of the `/configs` subdirectory, so let us examine the other directories found under `/usr/src/linux-2.6`. The most useful for us is the Documentation directory. In it and its subdirectories, you will find almost all the documentation concerning every part of the kernel. The file `00-INDEX` (each `Documentation` subdirectory also contains a `00-INDEX` file as well) contains a list of the files in the main directory and a brief explanation of what they are. Many files are written solely for kernel programmers and application writers, but a few are useful to the intermediate or advanced Linux user when attempting to learn about kernel and device driver issues. Some of the more interesting and useful documents are

`devices.txt`—A list of all possible Linux devices that are represented in the `/dev` directory, giving major and minor numbers and a short description. If you have ever gotten an error message that mentions `char-major-`*`xxx`*, this file is where that list is kept.

`ide.txt`—If your system uses IDE hard drives, this file discusses how the kernel interacts with them and lists the various kernel commands that can be used to solve IDE-related hardware problems, manually set data transfer modes, and otherwise manually manage your IDE drives. Most of this management is automatic, but if you want to understand how the kernel interacts with IDE devices, this file explains it.

`initrd.txt`—This file provides much more in-depth knowledge of initial ramdisks, giving details on the loopback file system used to create and mount them and explaining how the kernel interacts with them.

`kernel-parameters.txt`—This file is a list of most of the arguments that you can pass at boot time to configure kernel or hardware settings, but it does not appear too useful at first glance because it is just a list. However, knowing that a parameter exists and might relate to something you are looking for can assist you in tracking down more information because now you have terms to enter into an Internet search engine such as http://www.google.com/linux.

`sysrq.txt`—If you have ever wondered what that key on you keyboard marked "SysRq" was used for, this file has the answer. Briefly, it is a key combination hardwired into the kernel that can help you recover from a system lockup. Ubuntu disables this function by default for security reasons. You can re-enable it by entering the command `# echo "1" > /proc/sys/kernel/sysrq` and disable it by echoing a value of `0` instead of `1`.

In the other directories found in `Documentation`, you will find similar text files that deal with the kernel modules for CD-ROM drivers, file system drivers, gameport and joystick drivers, video drivers (not graphics card drivers—those belong to X11R6 and not to the kernel), network drivers, and all the other drivers and systems found in the Linux operating system. Again, these documents are usually written for programmers, but they can provide useful information to the intermediate and advanced Linux user as well.

The directory named `scripts` contains many of the scripts that `make` uses. It really does not contain anything of interest to anyone who is not a programmer or a kernel developer (also known as a kernel hacker).

After a kernel is built, all the compiled files will wind up in the `arch` directory and its subdirectories. Although you can manually move them to their final location, we will show you later in this chapter how the `make` scripts will do it for you. In the early days of Linux, this post-compilation file relocation was all done by hand; you should be grateful for `make`.

NOTE

The `make` utility is a very complex program. Complete documentation on the structure of `make` files, as well as the arguments that it can accept, can be found at http://www.gnu.org/software/make/manual/make.html.

The remainder of the directories in `/usr/src/linux-2.6` contain the source code for the kernel and the kernel drivers. When you install the kernel sources, these files are placed there automatically. When you patch kernel sources, these files are altered automatically. When you compile the kernel, these files are accessed automatically. Although you never need to touch the source code files, they can be useful. The kernel source files are nothing more than text files with special formatting, which means that we can look at them and read the programmers' comments. Sometimes, a programmer will write an application, but cannot (or often will not) write the documentation. The comments he puts in the source code are often the only documentation that exists for the code.

Small testing programs are even "hidden" in the comments of some of the code, along with comments and references to other information. Because the source code is written in a language that can be read as easily—almost—as English, a non-programmer might be able to get an idea of what the application or driver is actually doing (see Chapter 30, "C/C++ Programming Tools for Ubuntu"). This information might be of use to an intermediate to advanced Linux user when he is confronted by kernel- and driver-related problems.

NOTE

The interaction and control of hardware is handled by a small piece of the kernel called a *device driver*. The driver tells the computer how to interact with a modem, a SCSI card, a keyboard, a mouse, and so on in response to a user prompt. Without the device driver, the kernel does not know how to interact with the associated device.

Types of Kernels

In the early days of Linux, kernels were a single block of code containing all the instructions for the processor, the motherboard, and the other hardware. If you changed hardware, you were required to recompile the kernel code to include what you needed and discard what you did not. Including extra, unneeded code carried a penalty since the kernel became larger and occupied more memory. On older systems that had only 4MB–8MB of memory, wasting precious memory for unnecessary code was considered unacceptable. Kernel compiling was something of a "black art" as early Linux users attempted to wring the most performance from their computers. These kernels compiled as a single block of code are called *monolithic kernels*.

As the kernel code grew larger and the number of devices that could be added to a computer increased, the requirement to recompile became onerous. A new method of building the kernel was developed to make the task of compiling easier. The part of the kernel's source code that composed the code for the device drivers could be optionally compiled as a module that could be loaded and unloaded into the kernel as required. This is known as the *modular* approach to building the kernel. Now, all the kernel code could be compiled at once—with most of the code compiled into these modules. Only the required modules would be loaded; the kernel could be kept smaller, and adding hardware was much simpler.

The typical Ubuntu kernel has some drivers compiled as part of the kernel itself (called *in-line* drivers) and others compiled as modules. Only device drivers compiled in-line are available to the kernel during the boot process; modular drivers are only available after the system has been booted.

NOTE

As a common example, drivers for SCSI disk drives must be available to the kernel if you intend to boot from SCSI disks. If the kernel is not compiled with those drivers in-line, the system will not boot because it will not be able to access the disks.

A way around this problem for modular kernels is to use an initial Ram disk (`initrd`) discussed later in "Creating an Initial RAM Disk Image." The `initrd` loads a small kernel and the appropriate device driver, which then can access the device to load the actual kernel you want to run.

Some code can only be one or the other (for technical reasons unimportant to the average user), but most code can be compiled either as modular or in-line. Depending on the application, some system administrators prefer one way over the other, but with fast modern processors and abundant system memory, the performance differences are of little concern to all but the most ardent Linux hackers.

When compiling a kernel, the step in which you make the selection of modular or in-line is part of the `make config` step that we will detail later in this chapter. Unless you have a specific reason to do otherwise, we suggest that you select the modular option when given a choice. The process of managing modules is addressed in the next section because you will be managing them more frequently than you will be compiling a kernel.

Managing Modules

When using a modular kernel, special tools are required to manage the modules. Modules must be loaded and unloaded, and it would be nice if that were done as automatically as possible. We also need to be able to pass necessary parameters to modules when we load them—things such as memory addresses and interrupts. (That information varies from module to module, so you will need to look at the documentation for your modules to determine what, if any, information needs to be passed to it.) In this section, we will cover the tools provided to manage modules and then look at a few examples of using them.

Linux provides the following module management tools for our use. All these commands (and `modprobe.conf`) have `man` pages:

`lsmod`—This command lists the loaded modules. It is useful to pipe this through the `less` command because the listing is usually more than one page long.

`insmod`—This command loads the specified module into the running kernel. If a module name is given without a full path, the default location for the running kernel, `/lib/modules/*/`, will be searched. Several options are offered for this command—the most useful is `-f`, which forces the module to be loaded.

`rmmod`—This command unloads (removes) the specified module from the running kernel. More than one module at a time can be specified.

`modprobe`—A more sophisticated version of `insmod` and `rmmod`, it uses the dependency file created by `depmod` and automatically handles loading, or with the `-r` option, removing modules. There is no force option, however. A useful option to modprobe is `–t`, which causes `modprobe` to cycle through a set of drivers until it finds one that matches your system. If you were unsure of what module would work for your network card, you would use this command:

```
# modprobe -t net
```

The term `net` is used because that is the name of the directory (`/lib/modules/*/kernel/net`) where all the network drivers are kept. It will try each one in turn until it loads one successfully.

`modinfo`—This will query a module's object file and provide a list of the module name, author, license, and any other information that is there. It often is not very useful.

`depmod`—This program creates a dependency file for kernel modules. Some modules need to have other modules loaded first; that is, they "depend" on the other modules. (A lot of the kernel code is like this because it eliminates redundancy in the code base.) During the boot process, one of the startup files contains the command `depmod -a` and it is run every time you boot to re-create the file `/lib/modules/*/modules.dep`. If you make changes to the file `/etc/modprobe.conf`, run `depmod -a` manually. The `depmod` command, its list of dependencies, and the `/etc/modprobe.conf` file enable kernel modules to be automatically loaded as needed.

`/etc/modprobe.conf`—This is not a command, but a file that controls how `modprobe` and `depmod` behave; it contains kernel module variables. Although the command syntax can be quite complex, most actual needs are very simple. The most common use is to alias a module and then pass it some parameters. For example, in the following code, we alias a device name (from `devices.txt`) to a more descriptive word and then pass some information to an associated module. The `i2c-dev` device is used to read the CPU temperature and fan speed on our system. These lines for `/etc/modprobe.conf` were suggested for our use by the program's documentation. We added them with a text editor.

```
alias char-major-89 i2c-dev
options eeprom ignore=2,0x50,2,0x51,2,0x52
```

A partial listing of `lsmod` is shown here, piped through the `less` command, allowing us to view it a page at a time:

```
# lsmod ¦ less
Module                Size  Used by
parport_pc           19392  1
Module                Size  Used by
parport_pc           19392  1
lp                    8236  0
parport              29640  2 parport_pc,lp
autofs4              10624  0
sunrpc              101064  1
```

The list is actually much longer, but here we see that the input module is being used by the `joydev` (joystick device) module, but the joystick module is not being used. This computer has a joystick port that was autodetected, but no joystick is connected. A scanner module is also loaded, but because the USB scanner is unplugged, the module is not being used. You would use the `lsmod` command to determine whether a module was loaded and what other modules were using it. If you examine the full list, you would see modules for all the devices attached to your computer.

To remove a module, `joydev` in this example, use

```
# rmmod joydev
```

or

```
# modprobe -r joydev
```

A look at the output of `lsmod` would now show that it is no longer loaded. If we removed `input` as well, we could then use `modprobe` to load both `input` and `joydev` (one depends on the other, remember) with a simple

```
# modprobe joydev
```

If Ubuntu were to balk at loading a module (because it had been compiled using a different kernel version from what we are currently running; for example, the NVIDIA graphics card module), we could force it to load like this:

```
# insmod -f nvidia
```

We would ignore the complaints (error messages) in this case if the kernel generated any.

In Chapter 10, “Multimedia Applications,” we talked about loading a scanner module; in the example there, we manually loaded the scanner module and passed it the vendor ID. The scanner was not included in the lookup list because it is not supported by the GPL scanner programs; as a result, the scanner module was not automatically detected and loaded. However, the scanner will work with a closed-source application after the module is loaded. Automatic module management is nice when it works, but sometimes it is necessary to work with modules directly.

When to Recompile

Ubuntu systems use a modified version of the plain vanilla Linux kernel (a modified version is referred to as a *patched* kernel) with additional drivers and other special features compiled into it.

Ubuntu has quite an intensive testing period for all distribution kernels and regularly distributes updated versions. The supplied Ubuntu kernel is compiled with as many modules as possible to provide as much flexibility as possible. A running kernel can be further tuned with the `sysctl` program, which enables direct access to a running kernel and permits some kernel parameters to be changed. As a result of this extensive testing, configurability, and modularity, the precompiled Ubuntu kernel does everything most users need it to do. Most users only need to recompile the kernel to

- Accommodate an esoteric piece of new hardware.
- Conduct a system update when Ubuntu has not yet provided precompiled kernels.
- Experiment with the system capabilities.

Ubuntu supplies several precompiled versions of the kernel for Athlon and Pentium processors, for single- and multi-processor motherboards, and for Enterprise-class systems (higher security; uses 4GB of memory). These are all available through Synaptic, or just by grepping through the results of an `apt-cache` search.

Kernel Versions

The Linux kernel is in a constant state of development. As new features are added, bugs are fixed, and new technology is incorporated into the code base, it becomes necessary to provide stable releases of the kernel for use in a production environment. It is also important to have separate releases that contain the newest code for developers to test. To keep track of the kernels, version numbers are assigned to them. Programmers enjoy using sequential version numbers that have abstract meaning. Is version 8 twice as advanced as version 4 of the same application? Is version 1 of one application less developed than version 3 of another? The version numbers cannot be used for this kind of qualitative or quantitative comparison. It is entirely possible that higher version numbers can have fewer features and more bugs than older versions. The numbers exist solely to differentiate and organize sequential revisions of software.

For the latest development version of the kernel at the time of writing, for example, the kernel version number is `2.6.15-23`.

The kernel version can be broken down into four sections:

- `major version`—This is the major version number, now at `2`.
- `minor version`—This is the minor version number, now at `6`.
- `sublevel number`—This number indicates the current iteration of the kernel; here it is number `15`.
- `extraversion level`—This is the number representing a collection of patches and additions made to the kernel by the Ubuntu engineers to make the kernel work for them (and you). Each collection is numbered, and the number is indicated here in the kernel name. From our preceding example, it is `23`.

Typing `uname -r` at the command prompt displays your current kernel version.

```
# uname -r
2.6.15-23
```

Even-numbered minor versions are stable kernels, whereas odd-numbered minor versions are development releases. Version `2.6.x` is the stable production kernel, whereas version `2.5.x` is the development Linux kernel. When a new version of the development kernel is started, it will be labeled `2.7.x`.

For production machines, you should always use the kernels with even minor numbers. The odd minor numbers introduce new features, so you might want to use those on a test machine if you need features they provide.

Obtaining the Kernel Sources

The Linux kernel has always been freely available to anyone who wants it. If you just want to recompile the existing kernel, install the kernel sources package from the CD. To get the very latest “vanilla” version, open an FTP connection to ftp.kernel.org using your favorite FTP client and log in as anonymous. Because you are interested in the 2.6 kernel, change directories to `/pub/linux/kernel/v2.6`. The latest stable kernel as of this writing is `2.6.17`.

NOTE

ftp.kernel.org receives more than its share of requests for download. It is considered a courtesy to use a mirror site to reduce the traffic that ftp.kernel.org bears. http://www.kernel.org/mirrors/ has a list of all mirrors around the world. Find one close to your geographic location and substitute that address for ftp.kernel.org.

A number of different entries are on the FTP archive site for each kernel version, but because you are only interested in the full kernel it is only necessary to get the full package of source code. There are two of these packages:

```
linux-2.6.17.tar.gz
linux-2.6.17.bz2
```

Although these are the same kernel packages, they are built using different compression utilities: The `.gz` extension is the `gzip` package, found on almost every Linux system available. The `.bz2` extension is the newer `bzip2` utility, which has better compression than `gzip`. Both packages have the same content, so download the one compressed with the program you use.

Once downloaded, move the package to a directory other than `/usr/src` and unpack it. If you downloaded the `.gz` package, the unpacking command is `tar -xzvf linux-2.6.17.tar.gz`. Otherwise, the `bzip2` unpack command is `tar -xjvf linux-2.6.17.tar.bz2`. Once unpacked, the package will create a new directory, `linux-2.6.17`. Copy it to `/usr/src` or move it there. Then, create a symbolic link of `linux-2.6` to `linux-2.6.17` (otherwise, some scripts will not work). Here is how to create the symbolic link:

```
# rm /usr/src/linux-2.6
# ln -s /usr/src/linux-2.6.17 /usr/src/linux-2.6
```

By creating a symbolic link to `/usr/src/linux-2.6`, it is possible to allow multiple kernel versions to be compiled and tailored for different functions: You will just change the symbolic link to the kernel directory you want to work on.

CAUTION

The correct symbolic link is critical to the operation of `make`. Always have the symbolic link point to the version of the kernel sources you are working with.

Patching the Kernel

It is possible to patch a kernel to the newest Linux kernel version as opposed to downloading the entire source code. This choice can be beneficial for those who are not using a high-speed broadband connection. (A typical compressed kernel source file is nearly 30MB for a download time of about 10 minutes on a 512Kb DSL connection; adjust accordingly for your connection.) Whether you are patching existing sources or downloading the full source, the end results will be identical.

Patching the kernel is not a mindless task. It requires the user to retrieve all patches from her current version to the version she wants to upgrade to. For example, if you are currently running `2.6.1` (and have those sources) and want to upgrade to `2.6.8`, you must retrieve the `2.6.2` and `2.6.3` patch sets, and so on. Once downloaded, these patches must be applied in succession to upgrade to `2.6.8`. This is more tedious than downloading the entire source, but useful for those who keep up with kernel hacking and want to perform incremental upgrades to keep their Linux kernel as up-to-date as possible.

To patch up to several versions in a single operation, you can use the patch-kernel script located in the kernel source directory for the kernel version you currently use. This script applies all necessary version patches to bring your kernel up to the latest version.

The format for using the patch-kernel script looks like this:

```
patch-kernel source_dir patch_dir stopversion
```

The source directory defaults to `/usr/src/linux` if none is given, and the `patch_dir` defaults to the current working directory if one is not supplied.

For example, assume that you have a `2.6.6` kernel code tree that needs to be patched to the `2.6.8` version. The `2.6.7` and `2.6.8` patch files have been downloaded from `ftp.kernel.org` and are placed in the `/patch` directory in the source tree. You issue the following command in the `/usr/src/linux-2.6` directory:

```
# scripts/patch-kernel /usr/src/linux-2.6.6 /usr/src/linux-2.6.6/patch
```

Each successive patch file is applied, eventually creating a `2.6.8` code tree. If any errors occur during this operation, files named `xxx#` or `xxx.rej` are created, where `xxx` is the version of patch that failed. You will have to resolve these failed patches manually by examining the errors and looking at the source code and the patch. An inexperienced person will not have any success with this because you need to understand C programming and kernel programming to know what is broken and how to fix it. Because this was a stock `2.6.6` code tree, the patches were all successfully applied without errors. If you are attempting to apply a nonstandard third-party patch, the patch might likely fail.

When you have successfully patched the kernel, you are ready to begin compiling this code tree as if we started with a fresh, stock `2.6.8` kernel tree.

Using the `patch` Command

If you have a special, nonstandard patch to apply—such as a third-party patch for a commercial product, for example—you can use the `patch` command rather than the special patch-kernel script that is normally used for kernel source updates. Here are some quick steps and an alternative method of creating patched code and leaving the original code alone.

1. Create a directory in your home directory and name it something meaningful, like `mylinux`.
2. Copy the pristine Linux source code there with

   ```
   cp -ravd /usr/src/linux-2.6/* ~/mylinux
   ```
3. Copy the patch file to that same directory with

   ```
   cp patch_filename ~/mylinux
   ```
4. Change to the `~/mylinux` directory with

   ```
   cd ~/mylinux
   ```
5. Apply the patch with

   patch -p1 < *`patch_filename`* > mypatch.log 2>&1

   (This last bit of code saves the message output to a file so that you can look at it later.)
6. If the patch applies successfully, you are done and have not endangered any of the pristine source code. In case the newly patched code does not work, you will not have to reinstall the original, pristine source code.
7. Copy your new code to `/usr/src` and make that special symbolic link described elsewhere in the chapter.

Compiling the Kernel

If you want to update the kernel from new source code you have downloaded, or you have applied a patch to add new functionality or hardware support, you will need to compile and install a new kernel to actually use that new functionality. Compiling the kernel involves translating the kernel's contents from human-readable code to binary form. Installing the kernel involves putting all the compiled files where they belong in `/boot` and `/lib` and making changes to the bootloader.

The process of compiling the kernel is almost completely automated by the `make` utility as is the process of installing. By providing the necessary arguments and following the steps covered next, you can recompile and install a custom kernel for your use.

Here is a checklist of steps to compile and configure the kernel:

1. Verify a working bootdisk for the old kernel to be able to reboot your system in case something goes wrong with the new kernel.

CAUTION

Before making any changes to your current, working kernel, make sure that you have a backup copy on a floppy disk. This will allow you to boot into your system with a known working kernel in case something goes wrong during configuration. The command to do this is as follows:

```
# mkbootdisk --device /dev/fd0 `uname -r`
```

This assumes that your floppy drive is `/dev/fd0`. (Here is a good shell script tip: the ` character tells the shell to execute what is within ` first and then returns that output as part of the input of the mkbootdisk command.) On this machine, the result is

```
# mkbootdisk --device /dev/fd0 2.6.7-1
```

This command will not be echoed to your screen, but it is what the system will execute.

2. Apply all patches, if any, so that you have the features you desire. See the previous section for details.
3. Back up the `.config` file, if it exists, so that you can recover from the inevitable mistake. Use the following cp command:

```
# cp .config .config.bak
```

NOTE

If you are recompiling the Ubuntu default kernel, the `/usr/src/linux-2.6/configs` directory contains several versions of configuration files for different purposes.

Ubuntu provides a full set of `.config` files in the subdirectory configs, all named for the type of system they were compiled for. For example, `kernel-2.6.7-i686-smp.config` is a configuration file for a multiprocessor Pentium-class computer. If you want to use one of these default configurations as the basis for a custom kernel, simply copy the appropriate file to `/usr/src/linux-2.6` and rename it `.config`.

4. Run the `make mrproper` directive to prepare the kernel source tree, cleaning out any old files or binaries.
5. Restore the `.config` file that the command `make mrproper` deleted, and edit the Makefile to change the `EXTRAVERSION` number.

NOTE

If you want to keep any current version of the kernel that was compiled with the same code tree, manually edit the Makefile with your favorite text editor and add some unique string to the `EXTRAVERSION` variable.

You can use any description you prefer, however.

6. Modify the kernel configuration file using `make config`, `make menuconfig`, or `make xconfig`—we recommend the latter.
7. Run `make dep` to create the code dependencies used later in the compilation process.

TIP

If you have a multi-processor machine, you can use both processors to speed the make process by inserting `-j`*x* after the `make` command, where as a rule of thumb, *x* is one more than the number of processors you have. You might try a larger number and even try this on a single processor machine (we have used `-j8` successfully on an SMP machine); it will only load up your CPU. For example,

```
# make -j3 bzImage
```

All the make processes except `make dep` will work well with this method of parallel compiling.

8. Run `make clean` to prepare the sources for the actual compilation of the kernel.
9. Run `make bzImage` to create a binary image of the kernel.

NOTE

Several choices of directives exist, although the most common ones are as follows:

`zImage`—This directive compiles the kernel, creating an uncompressed file called `zImage`.

`bzImage`—This directive creates a compressed kernel image necessary for some systems that require the kernel image to be under a certain size for the BIOS to be able to parse them; otherwise, the new kernel will not boot. It is the most commonly used choice. However, the Ubuntu kernel compiled with `bzImage` is still too large to fit on a floppy, so a smaller version with some modules and features removed is used for the boot floppies. Ubuntu recommends that you boot from the rescue CD-ROM.

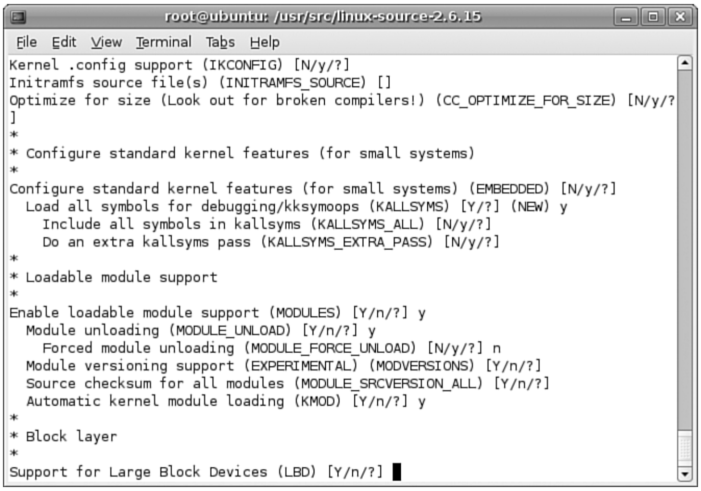
`bzDisk`—This directive does the same thing as `bzImage`, but it copies the new kernel image to a floppy disk for testing purposes. This is helpful for testing new kernels without writing kernel files to your hard drive. Make sure that you have a floppy disk in the drive because you will not be prompted for one.

10. Run `make modules` to compile any modules your new kernel needs.
11. Run `make modules_install` to install the modules in `/lib/modules` and create dependency files.
12. Run `make install` to automatically copy the kernel to `/boot`, create any other files it needs, and modify the bootloader to boot the new kernel by default.
13. Using your favorite text editor, verify the changes made to `/etc/lilo.conf` or `/boot/grub/grub.conf`; fix if necessary and rerun `/sbin/lilo` if needed.

14. Reboot and test the new kernel.
15. Repeat the process if necessary, choosing a Configuration Interface.

Over time, the process for configuring the Linux kernel has changed. Originally, you configured the kernel by responding to a series of prompts for each configuration parameter (this is the `make config` utility described shortly). Although you can still configure Linux this way, most users find this type of configuration confusing and inconvenient; moving back through the prompts to correct errors, for instance, is impossible.

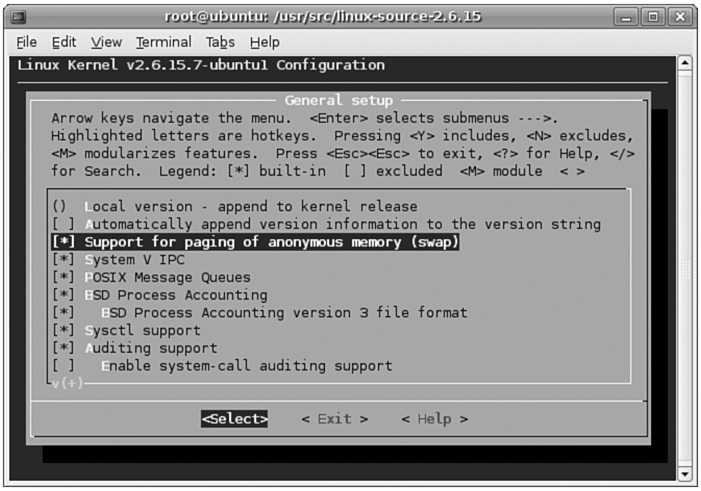
The `make config` utility is a command-line tool. The utility presents a question regarding kernel configuration options. The user responds with a `Y`, `N`, `M`, or `?`. (It is not case sensitive.) Choosing `M` configures the option to be compiled as a module. A response of `?` displays context help for that specific options, if available. (If you choose `?` and no help is available, you can turn to the vast Internet resources to find information.) We recommend that you avoid the `make config` utility, shown in Figure 32.1.


root@ubuntu: /usr/src/linux-source-2.6.15
File Edit View Terminal Tabs Help

```
Kernel .config support (IKCONFIG) [N/y/?]
Initramfs source file(s) (INITRAMFS_SOURCE) []
Optimize for size (Look out for broken compilers!) (CC_OPTIMIZE_FOR_SIZE) [N/y/?
]
*
* Configure standard kernel features (for small systems)
*
Configure standard kernel features (for small systems) (EMBEDDED) [N/y/?]
  Load all symbols for debugging/kksymoops (KALLSYMS) [Y/?] (NEW) y
    Include all symbols in kallsyms (KALLSYMS_ALL) [N/y/?]
    Do an extra kallsyms pass (KALLSYMS_EXTRA_PASS) [N/y/?]
*
* Loadable module support
*
Enable loadable module support (MODULES) [Y/n/?] y
  Module unloading (MODULE_UNLOAD) [Y/n/?] y
    Forced module unloading (MODULE_FORCE_UNLOAD) [N/y/?] n
  Module versioning support (EXPERIMENTAL) (MODVERSIONS) [Y/n/?]
  Source checksum for all modules (MODULE_SRCVERSION_ALL) [Y/n/?]
  Automatic kernel module loading (KMOD) [Y/n/?] y
*
* Block layer
*
Support for Large Block Devices (LBD) [Y/n/?] █
```

FIGURE 32.1 The `make config` utility in all its Spartan glory.

If you prefer to use a command-line interface, you can use `make menuconfig` to configure the Linux kernel. `menuconfig` provides a graphical wrapper around a text interface. Although it is not as raw as `make config`, `menuconfig` is not a fancy graphical interface either; you cannot use a mouse, but must navigate through it using keyboard commands. The same information presented in make config is presented by `make menuconfig`, but as you can see in Figure 32.2, it looks a little nicer. Now, at least, you can move back and forth in the selection process in case you change your mind or have made a mistake.


root@ubuntu: /usr/src/linux-source-2.6.15
File Edit View Terminal Tabs Help

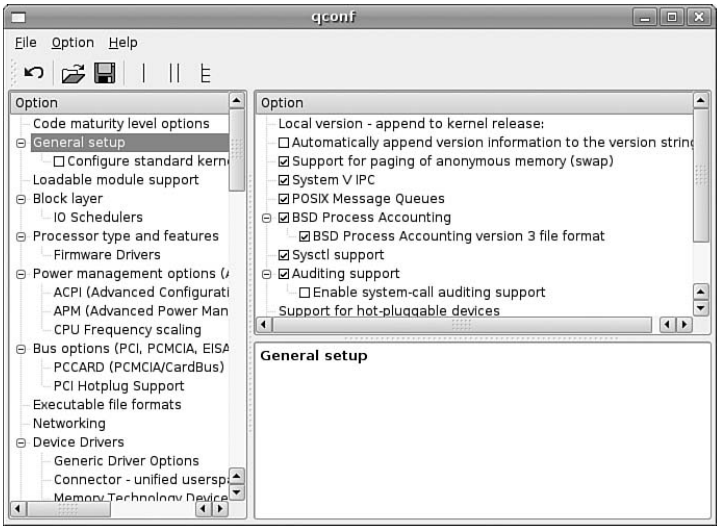
```
Linux Kernel v2.6.15.7-ubuntu1 Configuration

                        General setup
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

  ()  Local version - append to kernel release
  [ ] Automatically append version information to the version string
  [*] Support for paging of anonymous memory (swap)
  [*] System V IPC
  [*] POSIX Message Queues
  [*] BSD Process Accounting
  [*]   BSD Process Accounting version 3 file format
  [*] Sysctl support
  [*] Auditing support
  [ ]   Enable system-call auditing support
  v(+)

              <Select>    < Exit >    < Help >
```

FIGURE 32.2 The `make menuconfig` utility, a small improvement over `make config`.

In `make menuconfig`, you use the arrow keys to move the selector up and down and the spacebar to toggle a selection. The Tab key moves the focus at the bottom of the screen to either Select, Exit, or Help.

If a graphical desktop is not available, `menuconfig` is the best you can do. However, both `menuconfig` and `xconfig` (see below) offer an improvement over editing the `.config` file directly. If you want to configure the kernel through a true graphical interface—with mouse support and clickable buttons—`make xconfig` is the best configuration utility option. To use this utility, you must have the X Window System running. The application `xconfig` is really nothing but a Tcl/Tk graphics *widget set* providing borders, menus, dialog boxes, and the like. Its interface is used to wrap around data files that are parsed at execution time. Figure 32.3 shows the main menu of `xconfig` for the `2.6.7` kernel.

After loading this utility, you use it by clicking on each of the buttons that list the configuration options. Each button you click opens another window that has the detail configuration options for that subsection. Three buttons are at the bottom of each window: Main Menu, Next, and Prev(ious). Clicking the Main Menu button closes the current window and displays the main window. Clicking Next takes you to the next configuration section. When configuring a kernel from scratch, click the button labeled Code Maturity Level Options, and then continue to click the Next button in each subsection window to proceed through all the kernel configuration choices. When you have selected all options, the main menu is again displayed. The buttons on the lower right of the main menu are for saving and loading configurations. Their functions are self-explanatory. If you just want to have a look, go exploring! Nothing will be changed if you elect not to save it.



```
Code maturity level options
⊖ General setup
    ☐ Configure standard kern
  Loadable module support
⊖ Block layer
    IO Schedulers
⊖ Processor type and features
    Firmware Drivers
⊖ Power management options (
    ACPI (Advanced Configurati
    APM (Advanced Power Man
    CPU Frequency scaling
⊖ Bus options (PCI, PCMCIA, EISA
    PCCARD (PCMCIA/CardBus)
    PCI Hotplug Support
  Executable file formats
  Networking
⊖ Device Drivers
    Generic Driver Options
    Connector - unified usersp
    Memory Technology Device
```

```
  Local version - append to kernel release:
  ☐ Automatically append version information to the version strin
  ☑ Support for paging of anonymous memory (swap)
  ☑ System V IPC
  ☑ POSIX Message Queues
⊖ ☑ BSD Process Accounting
    ☑ BSD Process Accounting version 3 file format
  ☑ Sysctl support
⊖ ☑ Auditing support
    ☐ Enable system-call auditing support
  Support for hot-pluggable devices
```

FIGURE 32.3 The much nicer `make xconfig` GUI interface. We recommend that you use this interface if you are able.

If you are upgrading kernels from a previous release, it is not necessary to go though the entire configuration from scratch. Instead, you can use the directive `make oldconfig`; it uses the same text interface that `make config` uses, and it is noninteractive. It will just prompt for changes for any new code.

Using `xconfig` to Configure the Kernel

For simplicity's sake, during this brisk walk-through, we will assume that you are using `make xconfig`. Prior to this point, we also assume that you will have completed the first five steps in our kernel compilation checklist shown previously.

As you learned in the preceding section, you configure the kernel using `make xconfig` by making choices in several configuration subsection windows. Each subsection window contains specific kernel options. With hundreds of choices, the kernel is daunting to configure. We cannot really offer you detailed descriptions of which options to choose because your configuration will not match your own system and setup.

Table 32.1 provides a brief description of each subsection's options so that you can get an idea of what you might encounter. We recommend that you copy your kernel's `.config` file to `/usr/src/linux-2.6` and run `make xconfig` from there. Explore all the options. As long as you do not save the file, absolutely nothing will be changed on your system.

TABLE 32.1 Kernel Subsections for Configuration

Name	Description
Code maturity level options	Enables development code to be compiled into the kernel even if it has been marked as obsolete or as testing code only. This option should only be used by kernel developers or testers because of the possible unusable state of the code during development.
General setup	This section contains several different options covering how the kernel talks to the BIOS, whether it should support PCI or PCMCIA, whether it should use APM or ACPI, and what kind of Linux binary formats will be supported. Contains several options for supporting kernel structures necessary to run binaries compiled for other systems directly without recompiling the program.
Loadable module support	Determines whether the kernel enables drivers and other nonessential code to be compiled as loadable modules that can be loaded and unloaded at runtime. This option keeps the basic kernel small so that it can run and respond more quickly; in that regard, choosing this option is generally a good idea.
Processor type and features	Several options dealing with the architecture that will be running the kernel.
Power management options	Options dealing with ACPI and APM power management features.
Bus options	Configuration options for the PCMCIA bus found in laptops and PCI hotplug devices.
Memory Technology Devices (MTD)	Options for supporting flash memory devices, such as EEPROMS. Generally, these devices are used in embedded systems.
Parallel port support	Several options for configuring how the kernel will support parallel port communications.
Plug-and-play configuration	Options for supporting Plug and Play PCI, ISA, and plug-and-play BIOS support. Generally, it is a good idea to support plug-and-play for PCI and ISA devices.
Block devices	Section dealing with devices that communicate with the kernel in blocks of characters instead of streams. This includes IDE and ATAPI devices connected via parallel ports, as well as enabling network devices to communicate as block devices.
ATA/IDE/MFM/RLL support	Large collection of options to configure the kernel to communicate using different types of data communication protocols to talk to mass storage devices, such as hard drives. Note that this section does not cover SCSI.

TABLE 32.1 Continued

Name	Description
SCSI device support	Options for configuring the kernel to support Small Computer Systems Interface. This subsection covers drivers for specific cards, chipsets, and tunable parameters for the SCSI protocol.
Old CD-ROM drivers	Configuration options to support obscure, older CD-ROM devices that do not conform to the SCSI or IDE standards. These are typically older CD-ROM drivers that are usually a proprietary type of SCSI (not SCSI, not IDE).
Multi-device support	Options for enabling the kernel to support RAID devices in (RAID and LVM) software emulation and the different levels of RAID. Also contains options for support of a logical volume manager.
Fusion MPT device support	Configures support for LSI's Logic Fusion Message Passing Technology. This technology is for high performance SCSI and local area network interfaces.
IEEE1394 (firewire) support	Experimental support for Firewire devices.
I20 device support	Options for supporting the Intelligent Input/Output architecture. This architecture enables the hardware driver to be split from the operating system driver, thus enabling a multitude of hardware devices to be compatible with an operating system in one implementation.
Networking support	Several options for the configuration of networking in the kernel. The options are for the types of supported protocols and configurable options of those protocols.
Amateur radio support	Options for configuring support of devices that support the AX25 protocol.
IrDA (infrared) support	Options for configuring support of the infrared Data Association suite of protocols and devices that use these protocols.
Bluetooth support	Support for the Bluetooth wireless protocol. Includes options to support the Bluetooth protocols and hardware devices.
ISDN subsystem	Options to support Integrated Services Digital Networks protocols and devices. ISDN is a method of connection to a large area network digitally over conditioned telephone lines, largely found to connect users to ISPs.
Telephony support	Support for devices that enable the use of regular telephone lines to support VoIP applications. This section does not handle the configuration of modems.
Input device support	Options for configuring Universal Serial Bus (USB) Human Interface Devices (HID). These include keyboards, mice, and joysticks.
Character devices	Configuration options for devices that communicate to the server in sequential characters. This is a large subsection containing the drivers for several motherboard chipsets.

TABLE 32.1 Continued

Name	Description
Multimedia devices	Drivers for hardware implementations of video and sound devices such as video capture boards, TV cards, and AM/FM radio adapter cards.
Graphics support	Configures VGA text console, video mode selection, and support for frame buffer cards.
Sound	Large subsection to configure supported sound card drivers and chipset support for the kernel.
USB support	Universal Serial Bus configuration options. Includes configuration for USB devices, as well as vendor-specific versions of USB.
File system	Configuration options for supported file system types.
Additional device driver support	A section for third-party patches.
Profiling support	Profiling kernel behavior to aid in debugging and development.
Kernel hacking	This section determines whether the kernel will contain advanced debugging options. Most users will not want to include this option in their production kernels because it increases the kernel size and slows performance by adding extra routines.
Security options	Determines whether NSA Security Enhanced Linux (SELinux) is enabled.
Cryptographic options	Support for cryptography hardware (Ubuntu patches not found in the “vanilla” kernel sources).
Library routines	Contains zlib compression support.

After you select all the options you want, you can save the configuration file and continue with step 7 in the kernel compiling checklist shown earlier.

Creating an Initial RAM Disk Image

If you require special device drivers to be loaded in order to mount the root file system (for SCSI drives, network cards, or exotic file systems, for example), you must create an initial RAM disk image named `/boot/initrd.img`. For most users, it is not necessary to create this file, but if you are not certain, it really does not hurt. To create an `initrd.img` file, use the shell script `/sbin/mkinitrd`.

The format for the command is the following:

```
/sbin/mkinitrd file_name kernel_version
```

where *`file_name`* is the name of the image file you want created.

`mkinitrd` looks at `/etc/fstab`, `/etc/modprobe.conf`, and `/etc/ raidtab` to obtain the information it needs to determine which modules should be loaded during boot. For our system, we use

```
# mkinitrd initrd-2.6.7-1.img 2.6.7-1
```

When Something Goes Wrong

Several things might go wrong during a kernel compile and installation, and several clues will point to the true problem. You will see error messages printed to the screen, and some error messages will be printed to the file `/var/log/messages`, which can be examined with a text editor. If you have followed our directions for patching the kernel, you will need to examine a special error log as well. Do not worry about errors because many problems are easily fixed with some research on your part. Some errors may be unfixable, however, depending on your skill level and the availability of technical information.

Errors During Compile

Although it is rare that the kernel will not compile, there is always a chance that something has slipped though the regression testing. Let's take a look at an example of a problem that might crop up during the compile.

It is possible that the kernel compile will crash and not complete successfully, especially if you attempt to use experimental patches, add untested features, or build newer and perhaps unstable modules on an older system.

At this juncture, you have two options:

- Fix the errors and recompile.
- Remove the offending module or option and wait for the errors to be fixed by the kernel team.

Most users will be unable to fix some errors because of the complexity of the kernel code, although you should not rule out this option. It is possible that someone else discovered the same error during testing of the kernel and developed a patch for the problem: check the Linux kernel mailing list archive. If the problem is not mentioned there, a search on Google might turn up something.

The second option, removing the code, is the easiest and is what most people do in cases in which the offending code is not required. In the case of the NTFS module failing, it is almost expected because NTFS support is still considered experimental and subject to errors. This is primarily because the code for the file system is reverse-engineered instead of implemented via documented standards. Read-only support has gotten better in recent kernels; write support is still experimental.

Finally, should you want to take on the task of trying to fix the problem yourself, this is a great opportunity to get involved with the Linux kernel and make a contribution that could help many others.

If you are knowledgeable about coding and kernel matters, you might want to look in the `MAINTAINERS` file in the `/usr/src/linux-2.6/` directory of the kernel source and find the maintainer of the code. The recommended course of action is to contact the maintainer and see if he is aware of the problems you are having. If nothing has been documented for the specific error, submitting the error to the kernel mailing list is an option. The guidelines for doing this are in the `README` file in the base directory of the kernel source under the section "If Something Goes Wrong."

Runtime Errors, Boot Loader Problems, and Kernel Oops

Runtime errors occur as the kernel is loading. Error messages are displayed on the screen or written to the `/var/log/messages` file. Bootloader problems display messages to the screen; no log file is produced. *Kernel oops* are errors in a running kernel, and error messages are written to the `/var/log/messages` file.

Excellent documentation on the Internet exists for troubleshooting just about every type of error that LILO, GRUB, or the kernel could give during boot. The best way to find this documentation is to go to your favorite search engine and type in the keywords of the error you received. You will need to adjust the keywords you use as you focus your search.

In this category, the most common problems deal with LILO configuration issues. Diagnosis and solutions to these problems can be found in the LILO mini-HOWTO found on the Linux Documentation project's website at http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/LILO.html.

If you have GRUB problems, the GRUB manual is online at http://www.gnu.org/software/grub/manual/.

TIP

For best results, go to http://www.google.com/linux to find all things Linux on the Internet. Google has specifically created a Linux area of its database, which should allow faster access to information on Linux than any other search engine.

The Usenet newsgroup postings are searchable at http://www.google.com/grphp.

Mail list discussions can be searched in the Mailing list ARChives (MARC) at http://marc.theaimsgroup.com/.

Relevant Ubuntu and Linux Commands

You will use the following commands when managing the kernel and its modules in Ubuntu:

- `gcc`—The GNU compiler system
- `make`—GNU project and file management command
- `mkbootdisk`—Ubuntu's boot disk creation tool
- `sysctl`—The interface to manipulating kernel variables at runtime
- `mkinitrd`—Create a RAM-disk file system for bootloading support

Reference

- http://www.kernel.org/—Linux Kernel Archives. The source of all development discussion for the Linux kernel.
- http://www.kerneltraffic.org/kernel-traffic/index.html—Linux Kernel Traffic. Summarized version and commentary of the Linux Kernel mailing list produced weekly.
- http://www.gnu.org/—Free Software Foundation. Source of manuals and software for programs used throughout the kernel compilation process. Tools such as `make` and `gcc` have their official documentation here.
- http://slashdot.org/article.pl?sid=01/08/22/1453228&mode=thread—The famous AC Patches from Alan Cox, for whom they are named.
- http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html—The Linux Kernel Rebuild Guide; configuration, compilation, and troubleshooting.
- http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/KernelAnalysis-HOWTO.html—KernelAnalysis HOWTO. Describes the mysterious inner workings of the kernel.
- http://www.tldp.org/HOWTO/Module-HOWTO/—Kernel Module HOWTO. Includes a good discussion about unresolved symbols.
- http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Modules.html—The Linux Kernel Modules Installation HOWTO; an older document discussing recompiling kernel modules.
- http://www.tldp.org/—The Linux Documentation Project. The Mecca of all Linux documentation. Excellent source of HOWTO documentation, as well as FAQs and online books, all about Linux.
- http://www.minix.org/—The unofficial minix website. It contains a selection of links to information about minix and a link to the actual homepage. Although minix is still copyrighted, the owner has granted unlimited rights to everyone. See for yourself the OS used to develop Linux.
- http://jungla.dit.upm.es/~jmseyas/linux/kernel/hackers-docs.html—Web page with links to Linux kernel documentation, books, hacker tomes, and other helpful information for budding and experienced Linux kernel and kernel module developers. This list will also be found in the file `/usr/src/linux-2.6/Documentation/kernel-docs.txt` if you install the Ubuntu kernel sources.

This page intentionally left blank

PART VII
Appendixes

IN THIS PART

This page intentionally left blank

APPENDIX A
Ubuntu Under the Hood

IN THIS APPENDIX

As with any new thing, it's worthwhile finding out a bit about its history. Ubuntu is no different, and in this Appendix you'll learn a little more about where Linux (and Ubuntu) came from.

What Is Linux?

Linux is the core, or kernel, of a free operating system first developed and released to the world by Linus Benedict Torvalds in 1991. Torvalds, then a graduate student at the University of Helsinki, Finland, is now a Fellow at the Open Source Development Lab (http://www.osdl.org/). He is an engineer and previously worked for the CPU design and fabrication company Transmeta, Inc. Fortunately for all Linux users, Torvalds chose to distribute Linux under a free software license named the GNU General Public License (GPL).

NOTE

The free online resource Wikipedia has a great biography of Linus Torvalds that examines his life and notable achievements. You can find it at http://en.wikipedia.org/wiki/Linus_Torvalds. Or you can head on over to http://groups.google.com/group/comp.os.minix/msg/b813d52cbc5a044b?hl=en to read a copy of Linus's first post about Linux to the world.

The GNU GPL is the brainchild of Richard M. Stallman, the founder of the Free Software Foundation. Stallman, the famous author of the Emacs editing environment and GCC compiler system, crafted the GPL to ensure that software that used the GPL for licensing would always be free and available in source-code form. The GPL is the guiding document for Linux and its ownership, distribution, and copyright. Torvalds holds the rights to the Linux trademark, but thanks to a combination of his generosity, the Internet, thousands of programmers around the world, GNU software, and the GNU GPL, Linux will remain forever free and unencumbered by licensing or royalty issues. See the "Licensing" section later in this Introduction to learn more about the GNU GPL and other software licenses.

Distribution Version and Kernel Numbering Schema

There is a specific numbering system for Linux kernels, kernel development, and Ubuntu's kernel versions. Note that these numbers bear no relation to the version number of your Ubuntu Linux distribution. Ubuntu distribution version numbers are assigned by the Ubuntu developers, whereas most of the Linux kernel version numbers are assigned by Linus Torvalds and his legion of kernel developers.

To see the date your Linux kernel was compiled, use the `uname` command with its `-v` command-line option. To see the version of your Linux kernel, use the `-r` option. The numbers, such as 2.6.17-14, represent the major version (2), minor version (6), and patch level (17). The final number (14) is the developer patch level and is assigned by the Ubuntu developers.

Even minor numbers are considered "stable" and generally fit for use in production environments, whereas odd minor numbers (such as a Linux 2.7 source tree, not yet in existence) represent versions of the Linux kernel under development and testing. You will find only stable versions of the Linux kernel included with this book. You can choose to download and install a beta (test) version of the kernel, but this is not recommended for a system destined for everyday use. Most often, beta kernels are installed to provide support and testing of new hardware or operating system features.

Linux, pronounced "lih-nucks," is free software. Combining the Linux kernel with GNU software tools—drivers, utilities, user interfaces, and other software such as the X.Org Foundation's X Window System—creates a Linux distribution. There are many different Linux distributions from different vendors, but many derive from or closely mimic the Debian Linux distribution, on which Ubuntu is founded.

NOTE

To see just how many distributions are based on Debian Linux, go to http://www.linux.org/, click Distributions, and search for "Debian-based." At the time of writing, 52 distributions owe their existence to Debian.

Why Use Linux?

Millions of clever computer users have been putting Linux to work for more than 14 years. Over the past year, many individuals, small office/home office (SOHO) users, businesses and corporations, colleges, nonprofits, and government agencies (local, state, and federal) in a number of countries have incorporated Linux with great success. And, today, Linux is being incorporated into many information service/information technology (IS/IT) environments as part of improvements in efficiency, security, and cost savings. Using Linux is a good idea for a number of reasons, including the following:

- **Linux provides an excellent return on investment (ROI)**—There is little or no cost on a per-seat basis. Unlike commercial operating systems, Linux has no royalty or licensing fees, and a single Linux distribution on CD-ROM or network shared folder can form the basis of an enterprise-wide software distribution, replete with applications and productivity software. Custom corporate CD-ROMs can be easily crafted or network shares can be created to provide specific installs on enterprise-wide hardware. This feature alone can save hundreds of thousands, if not millions, of dollars in IS/IT costs—all without the threat of a software audit from the commercial software monopoly or the need for licensing accounting and controls of base operating system installations.

- **Linux can be put to work on the desktop**—Linux, in conjunction with its supporting graphical networking protocol and interface (the X Window System), has worked well as a consumer UNIX-like desktop operating system since the mid-1990s. The fact that UNIX is ready for the consumer desktop is now confirmed with the introduction, adoption, and rapid maturation of Apple Computer BSD UNIX—based on Mac OS X—supported, according to Apple, by more than 3,000 Mac OS X-specific programs that are known as native applications. This book's disc contains more than 800 software packages, including Internet connection utilities, games, a full office suite, many different fonts, and hundreds of graphics applications.

- **Linux can be put to work as a server platform**—Linux is fast, secure, stable, scalable, and robust. The latest versions of the Linux kernel easily support multiple-processor computers (optimized for eight CPUs), large amounts of system memory (up to 64GB RAM), individual file sizes in excess of hundreds of gigabytes, a choice of modern journaling file systems, hundreds of process monitoring and control utilities, and the (theoretical) capability to simultaneously support more than four billion users. IBM, Oracle, and other major database vendors all have versions of their enterprise software available for Linux.

- **Linux has a low entry and deployment cost barrier**—Maintenance costs can also be reduced because Linux works well on a variety of PCs, including legacy hardware, such as some Intel-based 486 and early Pentium CPUs. Although the best program performance will be realized with newer hardware, because clients can be recompiled and optimized for Pentium-class CPUs, base installs can even be performed on lower-end computers or embedded devices with only 8MB of RAM. This feature provides for a much wider user base; extends the life of older working hardware; and can help save money for home, small business, and corporate users.

- **Linux appeals to a wide audience in the hardware and software industry**—Versions of Linux exist for nearly every CPU. Embedded-systems developers now turn to Linux when crafting custom solutions using ARM, MIPS, and other low-power processors. Linux is the first full operating system available for Intel's Itanium

CPU, as well as the AMD64 group of CPUs; ports have also been available for HP/Compaq's Alpha and Sun Microsystems SPARC CPUs for some time. PowerPC users regularly use the PPC port of Linux on IBM and Apple hardware.

- **Linux provides a royalty-free development platform for cross-platform development**—Because of the open-source development model and availability of free, high-quality development tools, Linux provides a low-cost entry point to budding developers and tech industry start-ups.
- **Big-player support in the computer hardware industry from such titans as IBM now lends credibility to Linux as a viable platform**—IBM has enabled Linux on the company's entire line of computers, from low-end laptops through "Big Iron" mainframes. New corporate customers are lining up and using Linux as part of enterprise-level computing solutions. It has been used on some of the world's fastest computers, including IBM's Blue Gene/L. HP also certifies Linux across a large portion of its hardware offering.

Look forward to even more support as usage spreads worldwide throughout all levels of business in search of lower costs, better performance, and stable and secure implementations.

What Is Ubuntu?

Ubuntu is an operating system based on the Linux kernel; created, improved, refined, and distributed by the Ubuntu Community at http://www/ubuntu.com/. Ubuntu, sponsored by Canonical Software, is an open-source project supported by a worldwide community of software developers.

Roots of Ubuntu

Ubuntu is one of the newer Linux distributions currently available today, having released its first version in October 2004. It quickly gained a reputation for ease of installation and use, combined with the slightly wacky code names given to each release. However, Ubuntu itself is based on Debian, which is a much older distribution steeped in respect from the wider Linux community. Ubuntu describes Debian as being the rock on which it is founded, and this is a good way to describe the relationship between the two. It is also worth noting that Debian garnered a reputation for infrequent releases. The move from Debian 3.0 to 3.1 took almost three years, during which time many other Linux distros had moved far ahead of Debian.

Sponsored by Canonical Software and with the formidable resources of Mark Shuttleworth, Ubuntu got off to a great start with version 4.10, the Warty Warthog. From the start, Ubuntu specified clear goals: to provide a distribution that was easy to install and use, that did not overly confuse the user, and that came on a single CD (something increasingly rare these days when a distro can occupy four or five CDs). Releasing every six months, Ubuntu made rapid progress into the Linux community and is now one of the most popular Linux distros across the world.

Ubuntu Versions

As mentioned earlier, Ubuntu has chosen some peculiar code names for their releases since the first launch in October 2004. Doing away with the typical version numbering found elsewhere, Ubuntu decided to take the month and year of release and reverse them. Hence, the first release in October 2004 became 4.10, followed quickly by 5.04 (April 2005), 5.10, and 6.06.

The version covered in this book was released in October 2007, and thus bears the version number 7.10.

The code names are even better: 4.10 was christened the Warty Warthog in recognition that it was a first release, warts and all. The second release, 5.04, was dubbed the Hoary Hedgehog. Things got slightly better with 5.10, code-named the Breezy Badger. 6.06 was announced as the Dapper Drake and was the first Ubuntu distribution to carry the LTS (Long Term Support) badge, meaning that it is supported on the desktop for three years and on the server for five years. Beyond Dapper, there was the Edgy Eft (6.10) followed by the Feisty Fawn (7.04) and now the Gutsy Gibbon, which is what this book covers.

Ubuntu for Business

Linux has matured over the years, and features considered essential for use in enterprise-level environments, such as CPU architecture support, file systems, and memory handling, have been added and improved. The addition of virtual memory (the capability to swap portions of RAM to disk) was one of the first necessary ingredients, along with a copyright-free implementation of the TCP/IP stack (mainly due to BSD UNIX being tied up in legal entanglements at the time). Other features quickly followed, such as support for a variety of network protocols.

Ubuntu includes a Linux kernel that can use multiple processors, which allows you to use Ubuntu in more advanced computing environments with greater demands on CPU power. This kernel will support at least 16 CPUs; in reality, however, small business servers typically use only dual-CPU workstations or servers. However, Ubuntu can run Linux on more powerful hardware.

Ubuntu will automatically support your multiple-CPU Intel-based motherboard, and you will be able to take advantage of the benefits of symmetric multiprocessors (SMPs) for software development and other operations. The Linux kernels included with Ubuntu can use system RAM sizes up to 64GB, allow individual file sizes in excess of 2GB, and host the demands of—theoretically—billions of users.

Businesses that depend on high-availability, large-scale systems can also be served by Ubuntu, along with the specialist commercial support on offer from hundreds of support partners across the world.

However, Ubuntu can be used in many of these environments by customers with widely disparate computing needs. Some of the applications for Ubuntu include desktop support; small file, print, or mail servers; intranet web servers; and security firewalls deployed at strategic points inside and outside company LANs.

Commercial customers will also benefit from Debian's alliances with several top-tier system builders, including Hewlett-Packard.

Debian itself is also available for multiple architectures, and until recently was developed in tandem on 11 different architectures, from x86 to the older Motorola 680x0 chips (as found in the Commodore Amiga), along with several other architectures.

Small business owners can earn great rewards by stepping off the software licensing and upgrade treadmill and adopting a Linux-based solution. Using Ubuntu not only avoids the need for licensing accounting and the threat of software audits, but also provides viable alternatives to many types of commercial productivity software.

Using Ubuntu in a small business setting makes a lot of sense for other reasons, too, such as not having to invest in cutting-edge hardware to set up a productive shop. Ubuntu easily supports older, or *legacy*, hardware, and savings are compounded over time by avoiding unnecessary hardware upgrades. Additional savings will be realized because software and upgrades are free. New versions of applications can be downloaded and installed at little or no cost, and office suite software is free.

Ubuntu is easy to install on a network and plays well with others, meaning it works well in a mixed-computing situation with other operating systems such as Windows, Mac OS X, and, of course, UNIX. A simple Ubuntu server can be put to work as an initial partial solution or made to mimic file, mail, or print servers of other operating systems. Clerical staff should quickly adapt to using familiar Internet and productivity tools, while your business gets the additional benefits of stability, security, and a virus-free computing platform.

By carefully allocating monies spent on server hardware, a productive and efficient multiuser system can be built for much less than the cost of comparable commercial software. Combine these benefits with support for laptops, PDAs, and remote access, and you will find that Ubuntu supports the creation and use of an inexpensive yet efficient work environment.

Ubuntu in Your Home

Ubuntu installs a special set of preselected software packages onto your hard drive; these are suitable for small office/home office (SOHO) users. This option provides a wealth of productivity tools for document management, printing, communication, and personal productivity.

The standard installation requires nearly 2GB of hard drive space but should easily fit onto smaller hard drives in older Pentium-class PCs. The install also contains administrative tools, additional authoring and publishing clients, a variety of editors, a GNOME-based X11 desktop, support for sound, graphics editing programs, and graphical and text-based Internet tools.

64-Bit Ubuntu

Advances in computing saw the introduction of 64-bit x86-compatible CPUs from AMD in the spring of 2003. Intel's EM64T extensions for x86, which largely mirror the advances made by AMD, have further increased the availability of commodity x86-64 hardware.

Ubuntu fully supports AMD64 and Intel EM64T processors, and you are encouraged to use the 64-bit version of Ubuntu to get the maximum from these advanced processors.

As far as the Intel Itanium architecture goes, unfortunately Ubuntu does not currently support ia64. However, you may want to investigate Debian which is available for the ia64 platform.

Ubuntu on the PPC Platform

As well as being available on the popular x86 and x86-64 architectures, Ubuntu is compatible with the Power architecture developed by both IBM and Freescale. PowerPC, or PPC, uses a RISC processor that in some cases outperforms a similar Intel-based processor. Until recently, Apple produced all of their computers with PowerPC processors at the core. However, issues with supply and the delay in moving to a 3Ghz+ G5 chip forced Apple to look to Intel to provide the chips. However, both IBM and Freescale are heavily supporting the Power architecture, and companies such as Mercury and Genesi continue to provide workstations based around the Power chipset.

Getting the Most from Ubuntu and Linux Documentation

Nearly all commercial Linux distributions include shrink-wrapped manuals and documentation covering installation and configuration. You will not find official documentation included on the DVD provided with this book. However, you can read or get copies of assorted manuals or topic discussions online at http://www.ubuntu.com/. There you will find the links to various Ubuntu documentation projects.

Documentation for Ubuntu (and many Linux software packages) is distributed and available in a variety of formats. Some guides are available in Portable Document Format (PDF) and can be read using Adobe's Acrobat Reader for Linux or the `evince` client. Guides are also available as bundled HTML files for reading with a web browser such as links, KDE's Konqueror, GNOME's Epiphany, or Firefox. Along with these guides, Ubuntu provides various tips, frequently asked questions (FAQs), and HOWTO documents.

You will find traditional Linux software package documentation, such as manual pages, under the `/usr/share/man` directory, with documentation for each installed software package under `/usr/share/doc`.

Linux manual pages are compressed text files containing succinct information about how to use a program. Each manual page generally provides a short summary of a command's use, a synopsis of command-line options, an explanation of the command's purpose, potential caveats or bugs, the name of the author, and a list of related configuration files and programs.

For example, you can learn how to read manual pages by using the `man` command to display its own manual page, as follows:

```
$ man man
```

After you press Enter, a page of text appears on the screen or in your window on the desktop. You can then scroll through the information using your keyboard's cursor keys, read, and then press the Q key to quit reading.

Many of the software packages also include separate documents known as HOWTOs that contain information regarding specific subjects or software.

If the HOWTO documents are simple text files in compressed form (with filenames ending in `.gz`), you can easily read the document by using the `zless` command, which is a text pager that enables you to scroll back and forth through documents. (Use the `less` command to read plain-text files.) You can start the command by using `less`, followed by the complete directory specification and name of the file, or *pathname*, like this:

```
$ less /usr/share/doc/httpd-2.0.50/README
```

To read a compressed version of this file, use the `zless` command in the same way:

```
$ zless /usr/share/doc/attr-2.4.1/CHANGES.gz
```

After you press Enter, you can scroll through the document using your cursor keys. Press the Q key to quit.

If the HOWTO document is in HTML format, you can simply read the information using a web browser, such as Firefox. Or if you are reading from a console, you can use the links or lynx text-only web browsers, like this:

```
$ links /usr/share/doc/stunnel-4.0.5/stunnel.html
```

The links browser offers drop-down menus, accessed by clicking at the top of the screen. You can also press the Q key to quit.

If the documentation is in PostScript format (with filenames ending in `.ps`), you can use the `gv` client to read or view the document like this:

```
$ gv /usr/share/doc/iproute-2.4.7/ip-crefs.ps
```

Finally, if you want to read a document in Portable Document Format (with a filename ending in `.pdf`), use the `evince` client, as follows:

```
$ evince /usr/share/doc/xfig/xfig-howto.pdf
```

NOTE

This book was developed and written using the standard Desktop CD. You can use the disc included with this book for your install or download your own copy, available as ISO9660 images (with filenames ending in `.iso`), and burn it onto a 700MB CD-R or a DVD. You can also order prepressed CDs directly from Ubuntu at http://shipit.ubuntu.com.

Along with the full distribution, you will get access to the complete source code to the Linux kernel and source for all software in the distribution—more than 55 million lines of C and nearly 5 million lines of C++ code. Browse to http://www.ubuntu.com/download/ to get started.

Ubuntu Developers and Documentation

If you are interested in helping with Ubuntu, you can assist in the effort by testing beta releases (known as preview releases, and usually named after the animal chosen for the release name), writing documentation, and contributing software for the core or contributed software repositories. You should have some experience in installing Linux distributions, a desire to help with translation of documentation into different languages, or be able to use various software project management system, such as CVS.

Mailing lists are also available as an outlet or forum for discussions about Ubuntu. The lists are categorized. For example, general users of Ubuntu discuss issues on the ubuntu-users mailing list. Beta testers and developers via the ubuntu-devel, and documentation contributors via the ubuntu-doc mailing list. You can subscribe to mailing lists by selecting the ones you are interested in at http://lists.ubuntu.com/mailman/listinfo/. Be warned, some of the mailing lists have in excess of 200 to 300 emails per day!

Reference

- http://www-1.ibm.com/linux/—Information from IBM regarding its efforts and offerings of Linux hardware, software, and services.
- http://www.dwheeler.com/sloc—David A. Wheeler's amazing white paper covering the current state of GNU/Linux, its size, worth, components, and market penetration.
- http://www.ubuntu.com/—Home page for Ubuntu, sponsored by Canonical Software, and your starting point for learning more about Ubuntu.
- http://help.ubuntu.com/—Web page with links to current Ubuntu documentation and release notes.
- http://www.tldp.org/—The definitive starting point for the latest updates to generic Linux FAQs, guides, and HOWTO documents.

- http://www.ciac.org/ciac/—The U.S. Department of Energy's Computer Incident Advisory website, with details of security problems and fixes pertaining not only to Ubuntu, but many other operating systems. This site is useful for federal software contractors, developers, and system administrators.
- http://www.justlinux.com/—One site to which new Linux users can browse to learn more about Linux.
- http://www.linuxquestions.org/—A popular site that allows users to communicate and support each other with questions and problems spanning the entire Linux world.

APPENDIX B

Installation Resources

Installing a new operating system is always a major event, especially if you have never had to install an OS before. This is especially true if you are used to running Microsoft Windows XP that was pre-installed for you on your computer. In many cases, "recovery" discs are supplied that contain a mirror image of how your system was the day it rolled off the production line, so in reality you are not actually installing Windows, just copying some files. This appendix is all about helping you prepare for installing Ubuntu, taking you through some of the considerations that you perhaps do not realize are important to think about.

Until relatively recently Linux has been pretty difficult to install. You had to know every conceivable fact and specification about all the components of your computer to ensure that the installation went smoothly. Now, however, Ubuntu does most of the hard work for you, having much improved hardware detection and auto-configuration. This is definitely a good thing, and vastly reduces the time needed to install Ubuntu. Another good thing is the advent of Live CDs for Ubuntu, which give you a fully functional operating system on a CD. If you have ever been concerned about whether your system is compatible with Linux then take one of these Live CDs for a spin to help you make your decision.

This chapter will help guide you toward an installation of Ubuntu that closely matches your requirements. We start off with a look at some of the things you should take in to account when considering moving to Linux, including what your aims and objectives are for using Ubuntu. We also take a look at the hardware requirements of Ubuntu, along with information on how to check whether your hardware is compatible with Ubuntu. Finally, you will get a general overview of what installing Ubuntu looks like as well as how to avoid pitfalls with partitioning your hard drive. By the end of this chapter you should recognize just how flexible Ubuntu really is, both in the software it provides and also by the many ways in which you can install it.

Planning Your Ubuntu Deployment

The first thing you need to decide is why you are installing Ubuntu. By working out the "end use scenario" for the proposed installation, you then can begin to make choices and decisions about hardware specifications as well as software options. Before planning the specific steps of an installation, you need to make decisions about the type of deployment you want to undertake. For example, if you were going to use Ubuntu for 3D graphics work, then you would need to factor in the amount of space needed to store the sometimes intricate 3D models and graphics, as well as the graphics card needed for rendering, not to mention the amount of system memory and processor speed. On the flip side, if all you are doing is providing an elderly relative with a quick and easy way to access the Internet, then RAM, hard drive storage and processor speed are less likely to be important rather than a decent monitor, keyboard and mouse. You learn more about these issues in the sections that follow. These sections also include a table you can use as a pre-deployment planning checklist and some final advice for planning the installation.

Business Considerations

Making a choice of operating system for business can often be a thorny issue. Certainly there is a monopoly already in place from Microsoft, and there are a lot of users that have only ever used Microsoft products. This alone is a powerful argument to go down the Microsoft path, but there are other ways to implement Ubuntu in business. Your company may have been the target of a virus attack, or perhaps you have had to deal with one too many spyware and adware outbreaks on users desktops. Making the switch to Linux can eradicate many of these problems, increasing the uptime of users and reducing the security risk. The important thing is to work closely with the business to ensure that whatever is delivered is in line with the business requirements. If you consider that Linux is still in a minority, you need to think about how other companies will be able to work with you. Staff training and overall cost of change needs to be closely monitored at all times to ensure a smooth delivery. However, don't expect it to be perfect; anyone who has worked on a project knows that unexpected problems can and will occur, and you need to be as prepared as possible to deal with them.

Bear in mind that what works for your company may not work for another, so when swapping stories over a beer with other long-suffering sysadmins, think about how their successes can be adapted to your enterprise, but also pay close attention to things that went wrong. Even better, get one of their business users to present to your users and management to demonstrate the impact that moving to Linux has had. It's surprising how much good a relationship with other companies can do for your own IT infrastucture.

NOTE

As an example of inter-company relationships, most of the large law firms in London have their own soccer teams that regularly meet to do battle on the soccer pitch. They also meet to discuss IT issues and swap ideas between each other which benefits all of them. Why not set up a local corporate Linux User Group in your area? You don't have to make it a sports-related meeting, just make it clear that you want to share ideas and best practice.

One of the great things about Linux is that it allows you to try it before committing yourself. What other operating system do you know that can be booted up from a single CD and allow you to have a fully operational system, complete with applications? Although it sound like black magic, this kind of thing actually exists in the form of Live CDs and there are plenty to choose from, including the two that are available for Ubuntu (Gnome and KDE-based). Boot your system with one of these CDs to give you an idea of how well your hardware will cope with Linux.

Of course, if you are happy with the move to Linux, then you can ease the change by downloading versions of OpenOffice.org, Firefox, and Thunderbird for your existing platform so users can test them out before the migration.

Sometimes it is not always the visible changes that make the most difference. You should give careful thought to the potential deployment of Linux into such areas as web servers, file and print servers. You can extend the life of existing hardware long beyond it's useful "Windows" life by deploying them as print or web servers. Thankfully, Linux and open source software is pervasive enough to provide plenty of flexibility should you decide to test the water before diving in. Nowadays, popular open source applications such as OpenOffice.org are available for both Windows and Mac platforms, allowing you to try the software before deciding to switch. Also consider the ability to change back-end systems across to Linux-based alternatives. There are many Linux equivalents to Microsoft Exchange, for example, that can handle email and calendaring. Other popular servers ripe for moving across to Linux include file and print servers, web servers, and firewalls.

Do not think that you have to switch everything over in one go. Thankfully Linux plays well in a mixed environment (including Mac OS X and Windows XP), so you can quite safely plan a step-by-step migration that allows you to implement each phase one at a time. Moving servers across to new operating systems should be done on a server by server basis. Luckily Linux can easily co-exist in a multi-operating system environment, being compatible with Mac OS X, Windows, and Unix.

We have collated some of the question that need to be asked when considering a move to Ubuntu in Table B.1, titled "Deploying Ubuntu." As mentioned earlier, you need to identify the need that is going to be satisfied by moving to Ubuntu. Any project needs to meet a specific objective to be considered a success, so having this clear right at the start is essential. Another area of consideration is the impact to the existing computing environment. How will users cope with moving onto Linux; are they dyed in the wool Windows users that will resist any move to a different platform? Do you have the full support of management, something which is critical for projects of all sizes? By making successful changes behind the scenes, management can quickly be won over by the flexibility and choice of Open Source.

One of the key buzzwords to have come out of the dot com era is *Total Cost of Ownership*, and it is one that is fiercely debated when people talk about Linux. Those against Linux argue that although the software is free, the real cost comes in the amount of retraining and support necessary to move users to a new operating system. This can be circumvented by implementing Linux in situations where the end users are not directly affected, such as that web server that you have been planning to retire or the file and print server that needs more drive space. What is also often unseen is the increased availability that Linux-based systems offer companies. Quite simply they very rarely go down, unlike their Windows counterparts. Stability counts for a lot in this modern world of e-commerce where even a few minutes can cost thousands of dollars in lost orders and new customers. Talking about stability, one of the great things about Linux is that it does not necessarily need the latest hardware to function effectively—I have a router at home that is based on an old 486 machine that I bought sometime in 1994 coupled with a minimalist Linux distro! Think how many computers are needlessly disposed of that could be used as print servers or Internet gateways. The savings generated by sensibly recycling existing hardware are very tempting, and easily obtainable if you choose the Linux route.

In all of this, you need to be very clear what the objectives are. Specify exactly what you want to achieve from the project, what the Linux implementation will deliver, and how it will be used to replace any existing machines. What is the Linux machine replacing and what resources will be needed to maintain and support it? If you are rolling out to end users, what specific applications will they be using that you will have to provide support for?

Research is definitely a must before you embark on any project. It is also sensible to set up a test environment so that you can examine the performance of the new machine under set conditions to ensure that it functions in the way that you require. It is crucial that you spend a decent amount of time on testing because doing so will pay off in the long run with fewer bugs to fix and more positive user feedback and end user experience.

System Considerations

Ubuntu is flexible enough to cope with a wide range of computing needs, but with any switch of operating system you need to be aware of some of the issues that switching might cause. Some of them are listed in Table B.1. For example, how you choose to use Ubuntu could affect your choice of computer hardware, might affect your network configuration, and could dictate software policy issues (such as access, security, and allowable protocols).

Linux-based operating systems can be used to provide many different services. For example, one server might be boot management for a thin-client network in which workstations boot to a desktop by drawing a kernel and remotely mounted file systems over a network. This mechanism is not supported out of the box, so some effort can be expended if such a system is required. Other services more easily implemented (literally in an hour or less) could be centralized server environments for file serving, web hosting for a company intranet, or bridging of networks and routing services.

Linux supports nearly every network protocol, which enables it to be used to good effect even in mixed operating system environments. The security features of the Linux kernel and companion security software also make Linux a good choice when security is a top priority. Although no operating system or software package is perfect, the benefit of open source of the kernel and other software for Linux allows peer review of pertinent code and rapid implementation of any necessary fixes. Even with the secure features of Linux, some effort will have to be made in designing and implementing gateways, firewalls, or secure network routers.

Ubuntu can serve as a development platform for applications, e-commerce sites, new operating systems, foreign hardware systems, or design of new network devices using Linux as an embedded operating system. Setting up workstations, required servers, source code control systems, and industrial security will require additional effort.

Hardware compatibility can be an issue to consider when setting up a Linux server or building a Linux-based network. Fortunately, most of the larger server manufacturers such as IBM, HP, and even Dell realize that Linux-based operating systems (like other open source operating systems such as BSD) are increasingly popular, support open standards, and offer technologies that can help rapid introduction of products into the market (through third-party developer communities).

Ubuntu can help ease system administration issues during migration. The latest suite of Ubuntu's configuration utilities provides intuitive and easy to use graphical interfaces for system administration of many common services, such as networking, printing, and Windows-based file sharing. Ubuntu can also be used to support a legacy application environment, such as DOS, if required.

User Considerations

Humans are creatures of habit. It can be hard to transition a workforce, customer base, or other community to a new environment. The Ubuntu desktop, however, provides a friendly and familiar interface with menus and icons that new users can readily learn and put to work.

Part of the migration process can involve addressing user concerns, especially if Linux will take over the desktop. Ubuntu can be deployed in stages to make the migration process a bit easier, but the issue of user training must be addressed early on. This is especially true if users will be required to develop new skills or be aware of any caveats when using Linux (such as deleting all files in one's home directory). Although Ubuntu can be configured to provide a "turn-key" desktop in which only several graphical applications (such as a web browser, organizer, or word processor) can be used, some users will want and need to learn more about Linux.

A Predeployment Planning Checklist

Table B.1 provides a minimal checklist you can use to help plan a deployment.

TABLE B.1 Deploying Ubuntu

Consideration	Description
Applicability	How is Ubuntu going to be used?
Boot Management	Will remote booting be required?
Connectivity	Will the system be used in an internal network, or connected to the Internet. Is there a requirement for wireless connectivity? What about bandwidth?
Context	How does this install fit in with academic, business, or corporate needs?
Consensus	Are managers and potential users on board with the project?
Comparison	Is this install part of platform comparison or benchmarking?
Development Platform	Will development tools be used?
Embedded Device	Is it an embedded device project?
Hardware	Are there any special hardware or device interfacing requirements?
Finance	How much is in the budget? Will cost comparison be required?
Marketing	Will a product or service be offered as a result?
Networking	What type of networking will be required?
Objective	Is there a specific objective of the install?
Pilot Project	Is this a pilot or test install?
Power Management	Any special power or energy requirements?
Public Relations	Does the public need to know?
Quality of Service	Is high availability or data integrity an issue?
Roadmap	What other steps might precede or follow the install?
Reporting	Are follow-up reports required?
Security	What level or type of security will be required?
Server	Is this a server installation?
Site Considerations	Does the location provide needed temperature and security, or does it even matter?
Software	Are any special device drivers needed for success?
Storage	Are there size or integrity needs? Has a backup plan been devised?
Timeline	Are there time constraints or deadlines to the install?
Training	Will special training be required for users or administrators?
Users	How many and what type of users are expected?
Workstation	Is this a workstation or personal desktop install? Is the workstation portable?

Do not forget to address follow-up issues on your migration roadmap. You should pay attention to how satisfied or how well new users, especially those new to Linux, are adapting if a new desktop is used. However, if Ubuntu is deployed in a mixed environment, many users might not even know (or need to know) that Linux is being used!

Planning the Installation

There are many factors in favor of using Ubuntu as a computing solution. Ubuntu can fill many different roles on various tiers and hardware platforms because of the huge variety of software on offer.

Addressing software concerns beforehand can help quell any worries or fears felt by new users. Some key factors for a successful installation include

- **Preparation**—Thoroughly discuss the migration or deployment, along with benefits, such as greater stability and availability of service.
- **Pre-configuration**—If possible, give users a voice in software choices or categories and poll for comments regarding concerns.
- **Correct installation**—Ensure that the installed systems are working properly, including access permissions, password systems, or other user-related issues and interaction with the deployment.
- **The right hardware to do the job**—Make sure that users have the hardware they need for their work, and that computers match the tasks required. For example, developers will have workstation requirements vastly different from administrative personnel.

Hardware Requirements

Ubuntu can be installed on and will run on a wide variety of Intel-based hardware. This does not include pre-Pentium legacy platforms, but many older PCs, workstations, rack-mounted systems, and multiprocessor servers are supported. Small-, medium-, and even large-scale deployments of specially tuned Linux distributions are available through a number of companies such as IBM, which offers hardware, software, and service solutions.

TIP

It is always a good idea to explore your hardware options extensively before jumping on board with a specific vendor. You can buy computer hardware with a Linux distribution preinstalled. At the time of this writing, Dell Computer offered systems complete with Ubuntu (such as desktop PCs and laptops) through http://www.dell.com/ubuntu/. IBM also offers Linux on its product line, and more information can be found through http://www.ibm.com/linux/. To find HP and preinstalled Linux systems, browse to http://www.hp.com/linux/.

The type of deployment you choose also determines the hardware required for a successful deployment of Linux—and post-deployment satisfaction. The range of Linux hardware requirements and compatible hardware types is quite wide, especially when you consider that Linux can be used with mainframe computers as well as embedded devices.

Meeting the Minimum Ubuntu Hardware Requirements

The Ubuntu Project publishes general minimum hardware requirements for installing and using its base distribution in a file named RELEASE NOTES on the first CD-ROM or DVD, or available at http://www.ubuntu.com/products/whatisubuntu/desktopedition. For the current release, your PC should at least have a Pentium III CPU, 4GB hard drive space, and 128MB RAM for using (and installing) Ubuntu without a graphical interface. For obvious reasons, a faster CPU, larger capacity hard drive, and more RAM are desired. Servers and development workstations require more storage and RAM.

Using Legacy Hardware

If you have an older PC based on an Intel 486 CPU with only 32MB RAM and a 500MB hard drive (which can be hard to find nowadays), you can install other Linux distributions such as Debian from The Debian Project at http://www.debian.org/.

Installing Ubuntu on legacy hardware will be easier if you choose to use more recent Pentium-class PCs, but even older Pentium PCs can be used and purchased at a fraction of their original cost. Such PCs can easily handle many mundane but useful tasks. Some of the tasks suitable for older hardware include

- Acting as a firewall, router, or gateway
- Audio jukebox and music file storage server
- Handling email
- Hosting a remote printer and providing remote printing services
- Network font server
- Providing FTP server access
- Remote logging capture
- Secondary network-attached backup server
- Serving as an Intranet (internal LAN) web server
- Unattended dial-up gateway, voice mailbox, or fax machine
- Use as a "thin client" workstation for basic desktop tasks

Older PCs can handle any task that does not require a CPU with a lot of horsepower. To get the most out of your hardware, do not install any more software than required (a good idea in any case, especially if you are building a server). To get a little performance boost, add as much RAM as economically and practically feasible. If you cannot do this, cut down on memory usage by turning off unwanted or unneeded services. You can also recompile a custom Linux kernel to save a bit more memory and increase performance.

Planning for Hard Drive Storage for Your Ubuntu Installation

Making room for Ubuntu requires you to decide on how to use existing hard drive space. You might decide to replace existing hard drives entirely, for example, or you might decide to use only one operating system on your computer, making partitioning unnecessary. A full install from this book's DVD will require at least 7GB hard drive space just for the software, so if you plan to install everything, a 10GB hard drive could be ideal for a workstation. Note that depending on how you plan to use Linux, a smaller capacity disk can be used, or a disk capacity many times the size of your system will be required.

Checking Hardware Compatibility

Ubuntu software for Intel-based PCs is compiled for the minimum x86 platform supported by the Linux kernel.

NOTE

The compatibility information in this chapter relates to Ubuntu. Other distributions might have different storage and CPU requirements. Also bear in mind that Ubuntu is available for x86-64 and PPC architectures as well. Consult the release notes to get a detailed specification for these versions.

Specific issues regarding Linux hardware compatibility can be researched online at a number of sites. The community offers a hardware compatibility database at http://www.ubuntuhcl.org/pub/.

Other sites, such as the Linux-USB device overview at http://www.qbik.ch/usb/devices/, offer an interactive browsing of supported devices, and printer compatibility can be researched at LinuxPrinting.org at http://linuxprinting.org/. Some hardware categories to consider in your research include

- **Controller cards**—Such as SCSI, IDE, SATA, FireWire
- **CPUs**—Intel, AMD, Power, 64 Bit, and Multi-Core
- **Input devices**—Keyboards
- **Modems**—External, PCMCIA, PCI, and controllerless workarounds
- **Network cards**—ISA, PCI, USB, and others
- **Pointing devices**—Mice, tablets, and possibly touchscreens
- **Printers**—Various printer models
- **RAM**—Issues regarding types of system memory
- **Sound cards**—Issues regarding support
- **Specific motherboard models**—Compatibility or other issues

- **Specific PCs, servers, and laptop models**—Compatibility reports, vendor certification
- **Storage devices**—Removables, fixed, and others
- **Video cards**—Console issues (X compatibility depends on the version of X or vendor-based X distribution used)

If you have a particular laptop or PC model, you should also check with its manufacturer for Linux support issues. Some manufacturers such as HP now offer a Linux operating system pre-installed, or have an in-house Linux hardware certification program. Laptop users will definitely want to browse to Linux on Laptops at http://linux-laptop.net/.

TIP

There is a company called EmperorLinux in the U.S. that supplies laptops from prominent manufacturers with Linux pre-installed complete with support. They have been in business for a few years now, and ensure 100% compatibility with the laptops that they sell. Check out their range at www.emperorlinux.com.

If you cannot find compatibility answers in various online databases, continue your research by reading the Linux Hardware HOWTO at http://www.tldp.org/HOWTO/Hardware-HOWTO/. At that address, you will find loads of general information and links to additional sources of information.

Keep in mind that when PC hardware is unsupported under Linux, it is generally because the manufacturer cannot or will not release technical specifications or because no one has taken the time and effort to develop a driver. If you hit a roadblock with a particular piece of hardware, check the hardware manufacturer's support web pages, or Google's Linux pages at http://www.google.com/linux. You can then type in a specific search request and hopefully find answers to how to make the hardware work with Linux. This is also a good way to research answers to questions about software issues.

Preparing for Potential Hardware Problems

Ubuntu will work "out-of-the-box" with nearly every Intel- or PowerPC-based desktop, server and laptop; drivers for thousands of different types of hardware peripherals are included. But you can sometimes run into problems if Linux does not recognize a hardware item, if Ubuntu does not correctly initialize the hardware, or if an initialized item is incorrectly configured. For these reasons, some hardware items are prone to creating problems during an install. In the sections that follow, you learn some important pointers for avoiding these problems or resolving those that do occur.

Controllerless Modems

As you read earlier, most Linux hardware-related installation problems stem from a lack of technical specifications from the manufacturer, thwarting efforts of open source developers to create a driver. In the recent past, one hardware item that triggered both types of difficulties was the controllerless modem, also colloquially known as a *WinModem*. The good news is that modem chipset manufacturers have been more forthcoming with driver details. Some original equipment manufacturers, such as IBM, have made a concerted effort to provide Linux support. Support for the ACP Mwave modem, used in ThinkPad 600/Es and 770s, is included in the Linux kernel. Drivers have been developed for many of the controllerless modem chipsets that formally did not work with Linux.

If a driver is not available for your controllerless modem, you have a few options. You can download the driver's source code and build the driver yourself. Alternatively, you can download a binary-only software package and install the driver.

Some controllerless modems might also need to be initialized and configured using a separate utility program. The modem, if supported, should work normally after installing and configuring the driver.

You can research Linux support for controllerless modems by browsing to http://www.linmodems.org/.

USB Devices

Ubuntu supports hundreds of different Universal Serial Bus devices. USB is a design specification and a protocol used to enable a host computer to talk to attached peripherals. Because of lack of manufacturer and device ID information or lack of technical specifications regarding certain chipsets, some devices might not work with Ubuntu. USB 1.1 devices are designed to support data transfer speeds between 1.5 and 12Mbps.

Common USB devices include cameras, keyboards, mice, modems, network interfaces, printers, scanners, storage devices, video (such as webcams), and hubs (to chain additional devices).

Although some enlightened manufacturers are aware of opportunities in the Linux marketplace, most still do not support Linux. It pays to determine Linux support before you buy any USB device; again, research Linux USB support and its current state of development by browsing to http://www.qbik.ch/usb/devices/.

The newer USB 2.0 specification enables devices (such as hard and CD drives) to use speeds up to 480Mbps. Ubuntu supports USB 2.0 with the `ehci-hcd` kernel module. This driver, in development since early 2001, enables the use of many forms of newer USB 2.0 devices as long as you have a supported USB controller. Check out the current state of Linux USB 2.0 support by browsing to http://www.linux-usb.org/usb2.html.

Motherboard-Based Hardware

Small form factor PCs, thin clients, notebooks, and embedded devices are part of a growing trend in the PC industry. Manufacturers are cramming more functionality into fewer chips to simplify design and lower power requirements. Today, many computers come with built-in video graphics, audio chipsets, and network interfaces, along with a host of peripheral support.

Common modern (1996-onward) PC motherboard form factors are designed according to industry-assigned specifications (usually from Intel), and are ATX (12–9.6 inches); MicroATX (9.6–9.6 inches); and FlexATX (9–7.5 inches). One of the newest and even smaller motherboard forms is from VIA Technologies, Inc.—the mini-ITX (approximately 6.5–6.5 inches), which has an embedded CPU. CPUs commonly used in all these motherboards will vary, and have different socketing requirements based on chipset pins: Socket 478 for K7-type CPUs (from AMD); Socket 939 for some Athlon and Sempron processors; Socket AM2 for newer Athlon 64 and AMD FX processors; Socket 370 for Pentium IIIs and Celerons from Intel, or C3s from VIA; Socket 478 for Intel's Pentium 4s (early versions of which used a 423-pin socket) and socket LGA775 for newer Core 2 and Pentium D processors. Older socket types are Socket A, Socket 7 (and Super 7), Slot 1, and Slot B.

Fortunately, nearly all controllers, bridges, and other chipsets are supported by Linux. Although flaky or unsupported built-in hardware can (usually) be sidestepped by installing a comparable PCI card component, cutting-edge notebook users are at the most risk for compatibility problems because internal components are not user-replaceable. Potential pitfalls can be avoided through careful research (vote with your money for Linux-compatible hardware), or by choosing PC motherboards with a minimum of built-in features, and then using PCI (Peripheral Component Interconnect), AGP (Accelerated Graphics Port), or PCI Express cards known to work.

CPU, Symmetric Multiprocessing, and Memory Problems

Ubuntu supports all Pentium class x86 compatible CPUs. Code is included in the Linux kernel to recognize the CPU type when booting, and to then implement any required fixes to overcome architecture bugs (such as the now-infamous divide-by-zero error). After you install Ubuntu, you can also rebuild the Linux kernel to specifically support and take advantage of the host PC's CPU. You might not realize extreme improvements in computational speed, but you'll be assured that Linux is crafted for your CPU's architecture, which can help stability and reliability. Some of the x86-based CPUs with specific supporting code for Linux include those from Advanced Micro Devices, Transmeta, and VIA Technologies.

Ubuntu's Linux kernel also should automatically recognize and use the amount of installed RAM. The Linux kernel should also recognize and map out any memory holes in system memory (perhaps used for video graphics).

If you are installing Ubuntu on a working, stable PC, you should not have any problems related to the system's memory. If you are putting together a new system, you need to avoid combining or configuring the system in ways that will interfere with its capability to process data. Some issues to be aware of are

- Do not expect similar CPU performance across product lines from different manufacturers, such as AMD or VIA. Some CPU models offer better floating point or integer math operations, which are important for a number of CPU-intensive tasks (such as graphics, audio, and video rendering or conversion). If you need better performance, try to find a faster CPU compatible with your motherboard, or switch to a CPU with better Floating Point Unit (FPU) performance.

- Overclocking can cause problems with overheating, memory access, and other hardware performance, and it is not a good idea for any Linux system. Overclocking is a popular geek pastime and a great way to get a bit of performance boost out of slower CPUs by altering voltage settings and/or clock timings via the BIOS. You can try to push your CPU to higher speeds, but this approach is not recommended if your aim is system stability. The Linux kernel will report the recognized CPU speed on booting (which you can view using the `dmesg` command).
- Along the same lines, CPU and motherboard overheating will cause problems. Proper attachment of the CPU's heatsink using a quality thermal paste (never use thermal tape), along with one or more fans providing adequate airflow lessens the chance of hardware damage and system failure.
- You can run into problems if you switch the type of CPU installed in your computer, and especially if your PC's BIOS does not automatically recognize or configure for newly installed mainboard hardware and components. In some instances, a system reinstall is warranted, but BIOS issues should be resolved first.
- Not all CPUs support symmetric multiprocessing, or SMP. Ubuntu readily supports use of two or more CPUs and, during installation, automatically installs an appropriate Linux kernel. You can avoid problems by reading the Linux SMP HOWTO (available through http://www.tldp.org/). Note that some CPUs, such as the current crop of VIA C3s, might not be used for SMP. Also, SMP motherboards require that all CPUs be identical. This means that you need two identical CPUs to take advantage of SMP.
- Faulty or bad memory causes Linux kernel panics or Signal 11 errors (segmentation faults), causing a system crash or a program to abort execution. Linux is quite sensitive to faulty hardware, but runs with great stability in a correctly configured system with good hardware. Problems can arise from incorrect BIOS settings, especially if video memory must occupy and use a portion of system RAM. Always install quality (and appropriate) memory in your PC to avoid problems.

Preparing and Using a Hardware Inventory

Buying a turn-key Linux solution is one way to avoid hardware problems, and many vendors are standing by, ready to prescribe solutions. However, managing deployments aimed at using existing hardware requires some information collection.

If you are a small business or individual user, you are well advised to prepare detailed checklists of existing hardware before attempting a migration to Linux. Not only do you benefit from the collected information, but you might also be able to sidestep or anticipate problems before, during, or after installation. Problems are most likely to occur with newer hardware, cutting-edge hardware such as new motherboard chipsets and video cards, or extraneous hardware such as operating system–specific scanners, printers, or wireless devices.

Table B.2 provides a comprehensive checklist you can use to take inventory of target hardware, such as the computer and any peripherals. Veteran Linux users can take the collected information to build custom systems by adding known hardware or substituting cheaper but equivalent hardware.

TABLE B.2 System and Peripheral Inventory Checklist

Item	Errata
Audio Devices	Microphone:
	Line out:
	Line in:
BIOS	Type:
	Revision:
	ACPI:
	APM:
CD-ROM Drive	Brand:
	Type:
CD-RW Drive	Brand:
	Type:
	CDR Write Speed:
	CD Re-Write Speed:
	CD-ROM Read Speed:
DVD Drive	Brand:
	Type:
DVD+/-RW Drive	Brand:
	Type:
	Dual layer?:
Digital Camera	Brand:
	Model:
	Interface:
CPU	Brand:
	Socket type:
	Speed:
FireWire (IEEE 1394)	Chipset:
	Device(s):
IrDA Port	Device number:
	Port IRQ:
Keyboard	Brand:
	Type:
Laptop	Brand:
	Model:
	Hibernation partition:

TABLE B.2 Continued

Item	Errata
Legacy Ports	Parallel type:
	Parallel IRQ:
	RS-232 number(s):
	RS-232 IRQ(s):
Mice	Brand:
	Type:
Modem	Brand:
	Type:
Motherboard	Brand:
	Type:
	Chipset:
Monitor(s)	Brand:
	Model:
	Horizontal freq:
	Vertical freq:
	Max. resolution:
Network Card	Wireless:
	Brand:
	Type:
	Speed:
PCI Bus	Version:
	Model:
	Type:
PCMCIA	Controller:
	Cardbus:
	Brand:
	Type:
Printer(s)	Brand:
	Model:
System RAM	Amount:
	Type:
	Speed:
S-Video Port	X Support:
Scanner	Brand:
	Model:
	Interface type:

TABLE B.2 Continued

Item	Errata
Sound Card	Chipset:
	Type:
	I/O Addr:
	IRQ:
	DMA:
	MPU Addr:
Storage Device(s)	Removable:
	Size:
	Brand:
	Model:
	Controller(s):
Storage Device Controller	Type:
Tablet	Brand:
	Model:
	Interface:
Universal Serial Bus	Controller:
	BIOS MPS Setting:
	BIOS Plug-n-Play Setting:
	Device(s):
Video Device(s)	Brand:
	Model:
	Xinerama:
	Chipset:
	VRAM:

Use the checklist in Table B.2 as a general guideline for recording your computer's hardware and other capabilities. You can get quite a bit of information through hardware manuals or other documentation included with your PC, video, sound, or network interface card. Don't worry if you cannot fill out the entire checklist; Ubuntu will most likely recognize and automatically configure your PC's hardware during installation. Much of this information can be displayed by the `dmesg` command after booting. However, some of these details, such as your video card's graphics chipset and installed video RAM, can come in handy if you need to perform troubleshooting. You can also use the list as a post-installation check-off sheet to see how well Ubuntu works with your system.

Preparing for the Install Process

The basic steps in installing Ubuntu are to plan, install, and configure. You have to decide how to boot to an install and how much room to devote to Linux. Then perform the install (a sample step-by-step installation is discussed in Chapter 1, “Installing Ubuntu”) and afterward, configure your system to host new users and specific software services. Much of the initial work is done during the install process because the installer, Anaconda, walks you through partitioning, configuring the desktop, and configuration of any recognized network adapter.

There are many different ways to install Ubuntu, and selecting an installation method might depend on the equipment on hand, existing bandwidth, or equipment limitations. Here are some of the most commonly used installation methods:

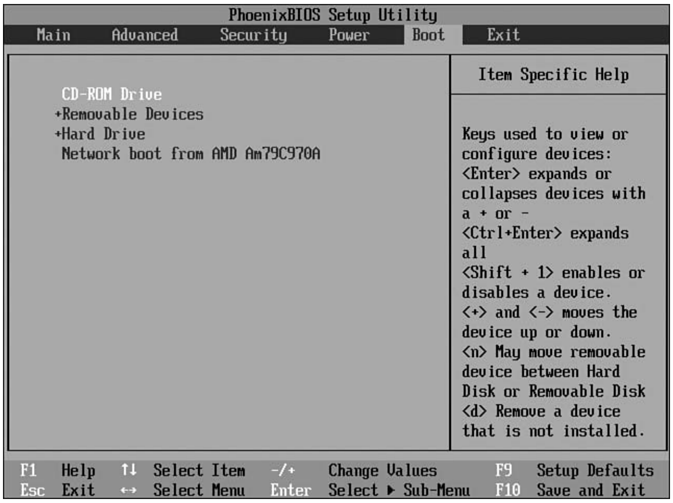
- **CD-ROM/DVD**—Using a compatible CD-ROM or DVD drive attached to the computer (laptop users with an external CD-ROM drive will need PCMCIA support from a driver disk image included under the first CD-ROM’s `images` directory).
- **Network File System (NFS)**—You can install Ubuntu from a remotely mounted hard drive containing the Ubuntu software. To perform this installation, you must have an installed and supported network interface card, along with a boot floppy with network support. (You learn how to make boot floppies later in this section of the chapter.)
- **File Transfer Protocol (FTP)**—As with an NFS install, installation via FTP requires that the Ubuntu software be available on a public FTP server. You also need an installed and supported network interface card, along with a boot floppy with network support.
- **Installation via the Internet**—If you have the bandwidth, it might be possible to install Ubuntu via the Internet; however, this method might not be as reliable as using a Local Area Network (LAN) because of availability and current use of The Ubuntu Project or other servers on mirror sites.
- **A hard drive partition**—By copying the `.iso` images to a hard drive partition, you can then boot to an install.
- **Pre-installed media**—It is also possible to install Linux on another hard drive and then transfer the hard drive to your computer. This is handy, especially if your site uses removable hard drives or other media.

After booting and choosing to use either a graphical or text-based install interface, the installation procedure is nearly the same for each type of install. Chapter 1 walks you through a typical installation.

Preparing to Install from a CD-ROM

Installing Ubuntu can be as simple as inserting the first CD/DVD into your computer’s CD drive and rebooting the computer. But if you choose this method, you should first make sure that your system’s BIOS is set to boot from CD-ROM.

Entering the BIOS to make this change is usually accomplished by depressing a particular key, such as Del or F2, immediately after turning on the computer. After entering the BIOS, navigate to the BIOS Boot menu, perhaps such as that shown in Figure B.1.



```
                    PhoenixBIOS Setup Utility
  Main    Advanced    Security    Power    Boot    Exit

                                         Item Specific Help
     CD-ROM Drive
    +Removable Devices
    +Hard Drive                          Keys used to view or
     Network boot from AMD Am79C970A     configure devices:
                                         <Enter> expands or
                                         collapses devices with
                                         a + or -
                                         <Ctrl+Enter> expands
                                         all
                                         <Shift + 1> enables or
                                         disables a device.
                                         <+> and <-> moves the
                                         device up or down.
                                         <n> May move removable
                                         device between Hard
                                         Disk or Removable Disk
                                         <d> Remove a device
                                         that is not installed.

 F1  Help  ↑↓ Select Item  -/+   Change Values     F9  Setup Defaults
 Esc Exit  ↔  Select Menu  Enter Select ▸ Sub-Menu F10 Save and Exit
```

FIGURE B.1 To boot to an install using your Ubuntu CD-ROM or DVD, set your BIOS to have your computer boot using its CD drive.

Partitioning Before and During Installation

Partitioning your hard drive for Linux can be done before or during installation.

If you plan to prepare your partitions before installing Linux, you will need to use commercial partitioning software. Some of the popular commercial software utilities you can use to create Linux partitions are Symantec's PartitionMagic or VCOM Products' Partition Commander. Alternatively, it might be possible to prepare partitions before installing Ubuntu by using the free `FIPS.EXE` command.

If you want to partition a hard drive using an existing Linux system, you can attach the hard drive to a spare IDE channel, and then use the Linux `fdisk` or GNU `parted` partitioning utilities. Both utilities offer a way to interactively partition and prepare storage media. Linux recognizes IDE hard drives using a device name such as `/dev/sda` (for the master device on IDE channel 0), `/dev/sdb` (for the slave device on IDE channel 0), `/dev/sdc` (for the master device on IDE channel 1), and `/dev/sdd` (for the slave device on IDE channel 1). With more modern computers that use the SATA interface, Linux will refer to drives as /dev/sda (for the master device on channel 0), /dev/sdb (for the slave device on channel 0), and so on.

If a new hard drive is properly attached to your PC and you then boot Linux, you can see whether the kernel recognizes the device by viewing the output of the `dmesg` command. You can then use `fdisk` with the device name to begin partitioning like so:

```
$ sudo fdisk /dev/sdb
```

Note that you will need root permission, and in this example, the new drive is attached as a slave on IDE channel 0. Do not change partitioning on your root device, or you will bork your system! The `fdisk` command is interactive, and you can press M to get help when using the utility. You can use `parted` in much the same way if you specify the `i`, or interactive option on the command line like so:

```
$ sudo parted -i /dev/sdb
```

To get help when using `parted` interactively, press `?` or type `help` followed by a command keyword. The `parted` command has other helpful features, such as the capability to copy a file system directly from one partition to another.

Finally, you can prepare partitions ahead of installation by booting your system using a live Linux distribution (such as the LNX Bootable Business Card, available at http://www.lnx-bbc.org/) and then using a native Linux utility such as `fdisk` to partition your drive.

TIP

You can use the first Ubuntu CD or the DVD to perform other tasks aside from installing Linux. The CD-ROM/DVD features a rescue mode and can also be used to partition and prepare a hard drive for Linux using fdisk as described above.

NOTE

It is possible to create a dual-boot configuration, which allows the choice of booting Ubuntu and another operating system, such as Windows XP. To configure your system for dual-booting, you must first install Windows and then install Linux. Note that many Windows system-restore CD-ROMs wipe out all data on your hard drive, including Linux. During installation of Ubuntu, you automatically install the GRUB Linux bootloader in the primary drive's Master Boot Record, or MBR. When properly configured, GRUB allows your system to reboot to Windows or Linux. Browse to http://www.gnu.org/software/grub/manual/ to read the GRUB manual online.

CAUTION

Before you begin partitioning your drive, get your safety nets in order. First, back up all critical data! Any changes to your system's hard drive or operating system put your existing data at risk. To prevent the loss of time and resources that inevitably follow data loss, do full backups before you make any changes to your system. Create a bootdisk during the install (you will be asked before the install finishes) so that you will be able to at least boot Linux if something goes wrong.

Choosing a Partitioning Scheme

As with deployment and installation of Linux, partitioning your hard drive to accept Ubuntu requires some forethought, especially if the target computer is going to be used other than as a home PC on which to learn Linux. If Linux is to be the only resident operating system, you can have the installer automatically create and use a partition scheme according to the type of installation you select during the install. If you plan to have a dual-boot system in which you can boot Linux or another operating system, you have to manually partition your hard drive before and possibly during the install.

The simplest and most basic partitioning scheme for a Linux system requires a Linux native root partition and a swap partition. On a single-drive system with 12GB storage and 512MB RAM, the scheme might look like this:

```
Hard Drive Partition    Mount Point    Size
/dev/sda1               /              10.74GB
/dev/sda2               swap           1GB
```

On a system running Windows, the scheme might look like this:

```
Hard Drive Partition    Mount Point    Size
/dev/sda1               /media/dos     4GB
/dev/sda2               /              7.74GB
/dev/sda3               swap           1GB
```

CAUTION

Notebook users should be careful when partitioning. Some notebooks use a special partition equal to the size of install RAM in order to perform suspend-to-disk or other hibernation operations. Always examine your computer’s initial partitioning scheme if configuring a dual-boot system, and leave the special partition alone! One way around this problem is to use a software suspend approach as outlined at www.suspend2.net/.

Hosting Parts of the Linux File System on Separate Partitions

Your choice of specific partitioning scheme will depend on how Ubuntu will be used. On a system being designed for expansion, greater capacity, or the capability to host additional software or users, you can use separate partitions to host various parts of the Linux file system. Some candidates for these separate partitions include

- `/home`—Users will store hundreds and hundreds of megabytes of data under their directories. This is important data, perhaps even more so than the system itself. Using a separate partition (on a different volume) to store this user data helps make the data easier to find and it segregates user and system data. You must decide ahead of time how much storage to allocate to users. For a single workstation, you should reserve several gigabytes of storage.

- `/opt`—As the home directory for additional software packages, this directory can have its own partition or remote file system. Ubuntu does not populate this directory, but it might be used by other software packages you install later. One gigabyte of storage should be adequate, depending on applications to be installed.
- `/tmp`—This directory can be used as temporary storage by users, especially if disk quotas are enforced; as such, it could be placed on its own partition. This directory can be as small as 100MB.
- `/usr`—This directory holds nearly all the software on a Ubuntu system and can become quite large if additional software is added, especially on a workstation configuration. Using a separate partition can make sense. A full install requires at least 6GB for this directory or more if additional software is added.
- `/var`—Placing this directory (or perhaps some of its subdirectories) on a separate partition can be a good idea, especially because security logs, mail, and print spooling take place under this tree. You should reserve at least one gigabyte of storage for `/var`, especially if using Ubuntu as a print server (as spooled documents will reside under `/var/spool`).

TIP

As a general rule, it is a good idea to segregate user and system data. Although a Linux system can be quickly restored, user data has a much greater value and can be much more difficult to replace. Segregating data can make the job of backing up and restoring much easier. If you ever have a problem accessing your partition, we recommend that you get the excellent Knoppix distribution that boots and runs entirely from CD. This will enable you to access your partitions and make any necessary repairs.

Reference

- http://www.yale.edu/pclt/BOOT/DEFAULT.HTM—A basic primer to partitioning that is operating system nonspecific.
- http://www-1.ibm.com/linux/—Home page for Linux at IBM, with links to products, services, and downloads.
- http://www-124.ibm.com/developerworks/opensource/linux390/—Home page for IBM S/390 Linux solutions.
- http://www.dell.com/linux/—Dell Computer’s Linux information pages.
- http://hardware.redhat.com/hcl/—Entry point to Red Hat’s hardware compatibility database.
- http://www.linux1394.org/—Home page for the Linux FireWire project, with information regarding the status of drivers and devices for this port.

- http://www.linux-usb.org/—Home page for the Linux USB project, with lists of supported devices and links to drivers.
- http://elks.sourceforge.net/—Home page for Linux for x286 and below CPUs, ELKS Linux.
- http://www.lnx-bbc.org/—Home page for the Bootable Business Card, a 50MB compressed Linux distribution that offers hundreds of networking clients, a live X session, web browsing, PDA backup, wireless networking, rescue sessions, and file recovery.
- http://www.coyotelinux.com/—Home page for several compact Linux distributions offering firewalling and VPN services. The floppy-based distribution works quite well on older PCs and does not require a hard drive.
- http://www.freesco.org/—Home page for a floppy-based Linux router solution that works on 386 PCs, requires only 6MB RAM, and provides bridging, firewalling, IP masquerading, DNS, DHCP, web, telnet, print, time, and remote access functions.
- http://www.bitwizard.nl/sig11/—A detailed overview of some root causes of Linux Signal 11 errors.
- http://www.gnu.org/software/parted/parted.html#introduction—Home page for the GNU parted utility.
- http://www.linux.org/vendors/systems.html—One place to check for a vendor near you selling Linux preinstalled on a PC, laptop, server, or hard drive.

APPENDIX C

Ubuntu and Linux Internet Resources

IN THIS APPENDIX

- ▶ Websites and Search Engines
- ▶ Usenet Newsgroups
- ▶ Mailing Lists
- ▶ Internet Relay Chat

Linux enjoys a wealth of Internet support in the form of websites with technical information, specific program documentation, targeted whitepapers, bug fixes, user experiences, third-party commercial technical support, and even free versions of specialized, fine-tuned clone distributions.

This appendix lists many of the supporting websites, FTP repositories, Usenet newsgroups, and electronic mailing lists that you can use to get more information and help with your copy of Ubuntu.

If You Need Help 24/7

If you are a small business, corporate, or enterprise-level Ubuntu user, do not forget that you can always turn to the source, Canonical, or third-party companies who supply Ubuntu support for commercial technical support on a 24/7 onsite basis, by phone, by electronic mail, or even on a per-incident basis. Canonical Software offers a spectrum of support options for its software products. You can read more about support options from the Ubuntu website: http://www.ubuntu.com/support/paid.

The appendix is divided into the following sections:

- ▶ Websites with Linux information arranged by category
- ▶ Usenet newsgroups pertaining to Linux
- ▶ Mailing lists providing Linux user and developer discussions
- ▶ Internet Relay Chat groups for Linux information

This appendix also lists websites that might be of general interest when using Ubuntu or specific components such as Xorg. Every effort has been made to ensure the accuracy of the URLs, but keep in mind that the Internet is always in flux!

Keep Up-to-Date

Keeping informed about bug fixes, security updates, and other errata is critical to the success and health of a Ubuntu system. To keep abreast of the most important developments when using Ubuntu, be sure to register with the Ubuntu Announcements mailing list. From there, you will learn which updates have been issued and what has been fixed as a result. Go to https://lists.ubuntu.com/mailman/listinfo/ubuntu-security-announce to register for this mailing list. You should also keep an eye out for Update Manager notifications in order to keep up with bug fixes, new software packages, and security updates.

Websites and Search Engines

Literally thousands of websites exist with information about Linux and Ubuntu. The key to getting the answers you need right away involves using the best search engines and techniques. Knowing how to search can mean the difference between frustration and success when troubleshooting problems. This section provides some Internet search tips and lists Ubuntu- and Linux-related sites sorted by various topics. The lists are not comprehensive, but have been checked and were available at the time of this writing.

Web Search Tips

Troubleshooting problems with Linux by searching the Web can be an efficient and productive way to get answers to vexing problems. One of the most basic rules for conducting productive searches is to use specific search terms to find specific answers. For example, if you simply search for “Ubuntu Linux,” you will end up with too many links and too much information. If you search for “Ubuntu sound,” however, you are more likely to find the information you need. If you’ve received an error message, use it; otherwise, use the Linux kernel diagnostic message as your search criterion.

Other effective techniques include the following:

- Using symbols in the search string, such as the plus sign (+) to force matches of web pages containing both strings (if such features are supported by the search engine used by web search site)
- Searching within returned results
- Sorting results (usually by date to get the latest information)
- Searching for related information
- Stemming searches; for example, specifying returns for not only “link” but also “linking” and “linked”

Invest some time and experiment with your favorite search engine's features—the result will be more productive searches. In addition to sharpening your search skills, also take the time to choose the best search engine for your needs.

Google Is Your Friend

Some of the fastest and most comprehensive search engines on the Web are powered by Linux, so it makes sense to use the best available resources. Out of the myriad number of websites with search engines, http://google.com stands out from the crowd, with millions of users per month. The site uses advanced hardware and software to bring speed and efficiency to your searches. If you are looking for specific Linux answers, take advantage of Google's Linux page at http://google.com/linux.

Why is Google (named after a math number) so powerful? You can get a quick overview from the Google folks at http://www.google.com/technology/. Part of its success is because of great algorithms, good programming, and simple interface design; but most users really seem to appreciate Google's uncanny capability to provide links to what you are looking for in the first page of a search return. Google's early success was also assured because the site ran its search engine on clusters of thousands of PCs running a version of Red Hat Linux! It is also rumored that an Ubuntu-based distribution is in use on desktops at Google, with the claimed moniker of Goobuntu.

Google has the largest database size of any search engine on the Web, with more than eight billion web pages searched and indexed. The database size is important because empty search results are useless to online users, and the capability to return hits on esoteric subjects can make the difference between success and failure or satisfaction and frustration. Some of Google's features include a GoogleScout link to return similar pages on the results page, the capability to see the exact version of a web page as returned to a search engine (known as a *cached* feature), advanced searches, and more recently, a link to an active Usenet news feed!

To get a better idea of what Google can offer you, browse to http://www.google.com/options/. You will find links to many different services and tools covering specialized searches, databases, information links, translators, and other helpful browsing tools.

Ubuntu Package Listings

You can quickly and easily view a list of the installed packages on your Ubuntu system, along with a short description of each package, by using `synaptic`.

`synaptic` also shows you descriptions of each package so you can decide whether to have it installed. For more information on `synaptic`, refer to Chapter 31, "Managing Software."

Certification

Linux certification courses are part of the rapidly growing information technology training industry. Hundreds of different vendors now offer courses about and testing of Linux skill sets. However, because Linux is open-source software, there are no formal rules or mandates concerning what knowledge or level of expertise is required for certification. If you are interested in certification and want to pursue a career or obtain employment with a company using Linux, you really should seek training. You can find a good list of companies that can provide training at http://www.ubuntu.com/training/, split down by geographical region.

Commercial Support

Commercial support for Ubuntu is an essential ingredient to the success of Linux in the corporate and business community. Although hundreds, if not thousands, of consultants well versed in Linux and UNIX are available on call for a fee, here is a short list of the best-known Linux support providers:

- http://www.ubuntu.com/support/paid—Go straight to the source for a range of support options. You can get help on Ubuntu direct from Canonical software, or from a local support provider.
- http://www.hp.com/linux—HP offer a comprehensive package of Linux services and hardware that cover almost everything that you would want to do, including consultancy, business solutions, and hardware specification and implementation.
- http://www.ibm.com/linux/—Linux services offered by IBM include e-business solutions, open-source consulting, database migration, clustering, servers, and support.

In addition to service-oriented support companies, nearly every commercial distributor of Linux has some form of easily purchased commercial support. There are various ways in which to take advantage of support services (such as remote management, onsite consulting, device driver development, and so on), but needs will vary according to customer circumstances and installations.

The Benefits of Joining a Linux User Group

Join a local *Linux Users Group (LUG)*! Joining and participating in a local LUG has many benefits. You will be able to get help, trade information, and learn many new and wonderful things about Linux. Most LUGs do not have membership dues, and many often sponsor regular lectures and discussions from leading Linux, GNU, and open-source experts. For one great place to start, browse to http://www.tux.org/luglist.html.

Documentation

Nearly all Linux distributions include thousands of pages of documentation in the form of manual pages, HOWTO documents (in various formats, such as text and HTML), mini-HOWTO documents, or software package documentation (usually found under the `/usr/share/doc/` directory). However, the definitive site for reading the latest versions of these documents is the Linux Documentation Project, found at http://www.tldp.org.

Linux Guides

If you are looking for more extensive and detailed information concerning a Linux subject, try reading one of the many Linux guides. These guides, available for a number of subjects, dwell on technical topics in more detail and at a more leisurely pace than a HOWTO. You can find copies of

- "Advanced Bash-Scripting Guide," by Mendel Cooper; a guide to shell scripting using `bash`
- "LDP Author Guide," by Mark F. Komarinski; how to write LDP documentation
- "Linux Administration Made Easy," by Steve Frampton
- "Linux Consultants Guide," by Joshua Drake; a worldwide listing of commercial Linux consultants
- "Linux from Scratch," by Gerard Beekmans; creating a Linux distribution from software
- "Linux Kernel Module Programming Guide," by Peter J Salzman, Michael Burian, and Ori Pomerantz; a good guide to building 2.4 and 2.6 series modules
- "Securing and Optimizing Linux," by Gerhard Mourani
- Linux certification
- "The Linux Network Administrator's Guide, Second Edition," by Olaf Kirch and Terry Dawson; a comprehensive Net admin guide

Ubuntu

- http://www.ubuntu.com—Home page for Ubuntu, Canonical's community-based free Linux distribution. Ubuntu is the main release of this Linux distribution and includes thousands of software packages that form the core of an up-to-date, cutting-edge Linux-based desktop. You can also find links to the other *buntus, such as Kubuntu, Xubuntu, and edubuntu.
- http://www.ubuntuforums.org—A good place to go if you need specific Ubuntu support.

Mini-CD Linux Distributions

Mini-CD Linux distributions are used for many different purposes. Some distributions are used to boot to a read-only firewall configuration; others are used to provide as complete a rescue environment as possible; whereas others are used to either install or help jump-start an install of a full distribution. Mini-CDs are available in a wide variety of sizes, such as 3" CD-Rs (or CD-RW) with sizes ranging from 185MB to 210MB. You can also download an `.iso` image and create a Linux bootable business card, typically fitting on a 40MB or 50MB credit-card–size CD-R. (Consider using a mini–CD-RW, especially if you want to upgrade your distribution often.) Here are some links to these distributions:

- http://www.lnx-bbc.com—Home page for the Linux BBC, a 40MB image hosting a rather complete live Linux session with X, a web browser, and a host of networking tools.
- http://crux.nu/—Home page of the CRUX i686–optimized Linux distribution.
- http://www.smoothwall.org—The 69MB SmoothWall distribution is used to install a web-administered firewall, router, or gateway with SSH, HTTP, and other services.

Various Intel-Based Linux Distributions

Choosing a Linux *distribution (distro)* for an Intel-based PC is generally a matter of personal preference or need. Many Linux users prefer Red Hat's distro because of its excellent support, commercial support options, and widespread use around the world. However, many different Linux distributions are available for download. One of the best places to start looking for a new distro or new version of your favorite distro is http://www.distrowatch.com:

- http://www.xandros.net—Home of the original and improved version of Corel's Debian-based Linux
- http://www.debian.org—The Debian Linux distribution, consisting only of software distributed under the GNU GPL license. Ubuntu, itself, is based upon Debian.
- http://www.slackware.com—Home page for download of the newest version of one of the oldest Linux distributions, Slackware
- http://www.opensuse.org—Home page for SUSE Linux, also available for the PowerPC and x86_64 platforms
- http://www.mepis.org—An increasingly popular distribution based on Debian
- http://www.mandrivalinux.com—A Pentium-optimized, RPM-based distribution, originally based on Red Hat's Linux distribution

PowerPC-Based Linux Distributions

- http://penguinppc.org/—Home page for the PowerPC GNU/Linux distribution
- http://www.opensuse.org—SUSE PPC Linux
- http://www.yellowdoglinux.com—Home page for Terra Soft Solutions's Yellow Dog Linux for the PowerPC, which is based on Fedora

Linux on Laptops and PDAs

One of the definitive sites for getting information about running Linux on your laptop is Kenneth Harker's Linux Laptop site. Although not as actively updated as in the past, this site (http://www.linux-laptop.net) still contains the world's largest collection of Linux and laptop information, with links to user experiences and details concerning specific laptop models.

Another site to check is Werner Heuser's Tuxmobil-Mobile UNIX website at http://www.tuxmobil.org. You will find links to information such as IrDA, Linux PDAs, and cell phones. Linux Zaurus PDA users can browse to http://www.openzaurus.org to download a complete open-source replacement operating system for the Zaurus 5000 and 5500 models.

The X Window System

Although much technical information is available on the Internet regarding the X Window System, finding answers to specific questions when troubleshooting can prove problematic. If you are having a problem using X, first try to determine whether the problem is software or hardware related. When searching or asking for help (such as on Usenet's `comp.os.linux.x` newsgroup, which you can access through Google's Groups link; see the next section for other helpful Linux newsgroups), try to be as specific as possible. Some critical factors or information needed to adequately assess a problem include the Linux distribution in use; the kernel version used; the version of X used; the brand, name, and model of your video card; the names, brands, and models of your monitor and other related hardware.

This section lists just some of the basic resources for Linux X users. Definitive technical information regarding X is available from http://www.X.org:

- http://www.lesstif.org/—Home page for the GPL'd OSF/Motif clone, LessTif
- http://www.rahul.net/kenton/index.shtml—Ken Lee's X and Motif website with numerous links to tutorial, development, and other information about X
- http://www.x.org—Home page for X.org, the X server used in Ubuntu
- http://www.xig.com/—Home page for a commercial version of X for Linux (along with other software products)

Usenet Newsgroups

Linux-related Usenet newsgroups are another good source of information if you're having trouble using Linux. If your ISP does not offer a comprehensive selection of Linux newsgroups, you can browse to http://groups.google.com/.

The primary Linux and Linux-related newsgroups are as follows:

- alt.os.linux.dial-up—Using PPP for dial-up
- alt.os.linux.mandriva—All about Mandriva Linux
- alt.os.linux.slackware—Using Slackware Linux
- alt.os.linux.ubuntu—Using Ubuntu Linux
- comp.os.linux.advocacy—Heated discussions about Linux and other related issues
- comp.os.linux.alpha—Using Linux on the Alpha CPU
- comp.os.linux.announce—General Linux announcements
- comp.os.linux.answers—Releases of new Linux FAQs and other information
- comp.os.linux.development.apps—Using Linux development tools
- comp.os.linux.development.system—Building the Linux kernel
- comp.os.linux.embedded—Linux embedded device development
- comp.os.linux.hardware—Configuring Linux for various hardware devices
- comp.os.linux.m68k—Linux on Motorola's 68K-family CPUs
- comp.os.linux.misc—Miscellaneous Linux topics
- comp.os.linux.networking—Networking and Linux
- comp.os.linux.portable—Using Linux on laptops
- comp.os.linux.powerpc—Using PPC Linux
- comp.os.linux.security—Linux security issues
- comp.os.linux.setup—Linux installation topics
- comp.os.linux.x—Linux and the X Window System
- comp.windows.x.apps—Using X-based clients
- comp.windows.x.i386unix—X for Unix PCs
- comp.windows.x.intrinsics—X Toolkit library topics
- comp.windows.x.kde—Using KDE and X discussions
- comp.windows.x.motif—All about Motif programming
- comp.windows.x—Discussions about X
- linux.admin.*—Two newsgroups for Linux administrators
- linux.debian.*—30 newsgroups about Debian

- `linux.dev.*`—25 or more Linux development newsgroups
- `linux.help`—Get help with Linux
- `linux.kernel`—The Linux kernel

Mailing Lists

Mailing lists are interactive or digest-form electronic discussions about nearly any topic. To use a mailing list, you must generally send an email request to be subscribed to the list, and then verify the subscription with a return message from the master list mailer. After subscribing to an interactive form of list, each message sent to the list will appear in your email inbox. However, many lists provide a digest form of subscription in which a single- or half-day's traffic is condensed in a single message. The digest form is generally preferred unless you have set up electronic mail filtering.

The main Ubuntu mailing lists are detailed here, but there are quite a few Linux-related lists. You can search for nearly all online mailing lists by using a typical mailing list search web page, such as the one at http://www.lsoft.com/lists/list_q.html.

GNOME and KDE Mailing Lists

GNOME users and developers should know that more than two dozen mailing lists are available through http://mail.gnome.org/. KDE users will also benefit by perusing the KDE-related mailing lists at http://www.kde.org/mailinglists.html.

Ubuntu Project Mailing Lists

The Ubuntu Project is always expanding, as many new users continue to find Ubuntu for the first time. You will find many other knowledgeable users with answers to your questions by participating in one of Ubuntu's mailing lists. The lists are focused on using, testing, and developing and participating in Ubuntu's development:

- http://lists.ubuntu.com/mailman/listinfo/ubuntu-security-announce—Security announcements from the Ubuntu developers
- http://lists.ubuntu.com/mailman/listinfo/ubuntu-announce—Announcements concerning Ubuntu
- http://lists.ubuntu.com/mailman/listinfo/ubuntu-users—Discussions among users of Ubuntu releases
- http://lists.ubuntu.com/mailman/listinfo/ubuntu-devel—Queries and reports from developers and testers of Ubuntu test releases

Internet Relay Chat

Internet Relay Chat (IRC) is a popular form and forum of communication for many Linux developers and users because it allows an interactive, real-time exchange of information and ideas. To use IRC, you must have an IRC client and the address of a network and server hosting the desired chat channel for your discussions.

You can use the `irc.freenode.net` IRC server, or one listed at http://www.freenode.net/ to chat with other Ubuntu users. Some current channels are as follows:

- `#Ubuntu`—General chat about Ubuntu
- `#edubuntu`—General chat about EdUbuntu
- `#xubuntu`—General chat about Xubuntu
- `#kubuntu`—General chat about Kubuntu

For more channels to look at head on over to https://help.ubuntu.com/community/InternetRelayChat for an exhaustive list of "official" channels.

However, Google can help you find other channels to explore. Simply enter in the distro name and IRC into the search options to retrieve information on any IRC channels relevant to your requirements. To get help with getting started with IRC, browse to http://www.irchelp.org/. Some of the channels of interest might be

- `#linux`—General discussions about Linux
- `#linuxhelp`—A help chat discussion for new users

Most IRC networks provide one or more Linux channels, although some providers require signup and registration before you can access any chat channel.

Index

How can we make this index more useful? Email us at indexes@samspublishing.com

How can we make this index more useful? Email us at indexes@samspublishing.com

How can we make this index more useful? Email us at indexes@samspublishing.com

How can we make this index more useful? Email us at indexes@samspublishing.com

How can we make this index more useful? Email us at indexes@samspublishing.com

What's on the DVD

The book's DVD includes the binary version of Ubuntu 7.10—the equivalent of at least four CDs.

DESKTOPS

- X.Org 7.2-5
- GNOME 2.20.0

ACCESSIBILITY

- GNOME On-screen keyboard
- Screen reader and magnifier

ACCESSORIES

- Alacarte Menu Editor
- Ark
- Calculator
- Character Map
- Dictionary
- GVim Text Editor
- KAlarm
- KArm
- Katapult
- KCalc
- KjobViewer
- Klipper
- KNotes
- KPager
- KPilot
- KSig
- KTip
- Mousepad
- Screenshot
- SpeedCrunch
- Terminal
- Text Editor
- Tomboy Notes
- Xarchiver
- Xfburn
- Xfce 4 Appfinder

EDUCATION

- Kalzium
- Kanagram
- KBruch
- KEduca
- KHangMan
- Kig
- KLatin
- KLettres

EDUCATION (continued)

- KmPlot
- KPercentage
- KStars
- KTouch
- KTurtle
- Kverbos
- KVocTrain
- GROMMACS

GAMES

- Ace of Penguins
- Atlantik
- Atlantik Designer
- Atomix
- Chess
- Educational suite gcompris
- Four-in-a-row
- FreeCell Solitaire
- KAsteroids
- KAtomic
- KBackgammon
- KBattleship
- KBlackBox
- KBounce
- Kenolaba
- KFoulEggs
- KGoldrunner
- KJumpingCube
- Klickety
- KMahjongg
- KMines
- Kolf
- Konquest
- KPoker
- KReversi
- KSirtet
- KSmileTris
- KSnakeRace
- KSokoban
- KSpaceDuel
- KTron
- KWin4
- Lieutenant Skat
- Mahjongg

GAMES (continued)

- Mines
- Patience
- Potato Guy
- Robots
- SameGame
- Shisen-Sho
- Sudoku
- TuxMath
- TuxTyping
- Zatacka

GRAPHICS

- Dia
- digiKam
- Document Viewer
- FontForge
- F-Spot Photo Manager
- GIMP Image Editor
- GNU Paint
- GQview
- gThumb Image Viewer
- Gwenview
- Image Viewer
- Inkscape Vector Illustrator
- KFaxView
- KGhostView
- Kooka
- KPDF
- Krita
- KSnapshot
- OpenOffice.org Drawing
- QCaD
- Scribus
- XaoS
- XSane Image Scanner

INTERNET

- Akregator
- BitTorrent
- Bluetooth
- Bluetooth OBEX Client
- Bluetooth OBEX Server
- Ekiga Softphone
- Epiphany Web Browser