

## CHAPTER 5

# First Steps with Fedora

Most modern Linux distros are a world away from versions that were available only four or five years ago. Sporting a highly polished graphical interface, Fedora gives you graphical tools with which to carry out most system administration tasks. However, things can occasionally go wrong and it is important that you know what to do in the event that you cannot use the GUI interface. In this chapter we will look at some of the basics of Fedora, laying the foundations for other chapters in this book. We will cover the Linux file system, as well as working with essential user information and accessing useful documentation available for Fedora. We will also take a look at working with the shell, otherwise known as the command line interface. Then we will explore the various text editors that can be used with Fedora, as well as examine the fundamentals of file permissions. Finally, we will clearly explain the importance of the root or super-user account in the maintenance and administration of your system.

Some of the basic command-line skills covered in this chapter include

- **Performing routine tasks**—Logging in and out, using the text console, changing passwords, and listing and navigating directories
- **Basic file management**—Creating, renaming, and deleting files and directories
- **Basic user management**—Creating and deleting users from the command line
- **Basic system management**—Shutting down and rebooting, reading man pages and other documentation, and using text-based tools to edit system configuration files

### IN THIS CHAPTER

- Working with the Linux File System
- Logging In to and Working with Linux
- Changing Your User Information
- Reading Documentation
- Using the Shell
- Using the Text Editors
- Working with Permissions
- Working As Root
- Reference

Read this chapter if you are migrating to Linux from another platform; the information here is valuable for individual users or system administrators who are new to Linux and are learning to use the command line for the first time.

**TIP**

For those of you who have used MS-DOS in the past, delving into the command line does not seem as scary as for those who, when faced with a black screen, automatically press the speed-dial button allocated for their neighborhood geek. Regardless of which user you are, knowledge of the command line serves you well. Because Linux is a UNIX-style operating system, if you learn the command line functions on Linux, it will be easier for you to use other UNIX-like operating systems, such as the BSD's and even Mac OS X.

**NOTE**

On the whole, there are two types of users that access a Linux system: normal day-to-day users and the root user or super user. At large organizations, a few people might be granted access rights to the root or super-user account in order to complete necessary system administration tasks. Normal users do not normally need to have root access.

However, if you have Linux installed on your machine as a standalone PC, you automatically have access to the root account. This is so that you can complete necessary configuration and other tasks that require interacting with the system.

This particular aspect of Linux is the victim of a lot of bad press because new users can see it as a significant obstacle to them getting their work done. This is not the case: The root account, as mentioned earlier, is fundamental to a successful Linux system. A lot of work has gone into creating easy-to-use administration tools that take away a lot of the difficulty in maintaining your system. It is still the case that a bit of command-line knowledge serves you well, especially in an emergency.

## Working with the Linux File System

Fedora uses a *file system*, or layout of hierarchical directories similar to that used by other UNIX variants (such as Mac OS X). Nearly all Linux distributions use a similar directory structure, and Linux distribution vendors have generally agreed on the naming and location of critical Linux files and directories.

**NOTE**

The effort to build a consensus regarding the Linux directory structure began in 1993 with the Filesystem Hierarchy Standard (FHS), a draft proposal that addressed not only Linux issues, but also those of other operating systems, such as BSD. Red Hat has stated that it is committed to staying compliant with the FHS, which specifies the location and names of files and directories.

Fedora uses the current 2.3 standard. Key additions to this include the creation of a `/media` directory for removable storage devices such as DVD drives and zip disks (although keeping `/mnt` as a temporary mount point). `udev` has also been added to Fedora; although it is not a part of

FHS 2.3, it enables `/dev` to become a dynamically managed folder allowing the hot-plugging of devices onto the system and the creation (on demand) of device nodes.

The commonality of how the Linux directory structure is laid out is very useful for open source developers because it cuts down the amount of work they have to do to get their programs to work with different distributions. For a programmer to know, for example, that the `useradd` command is always under `/usr/sbin` means that he can create shell scripts and other utilities that take advantage of this, and know that they will work universally. Perhaps unsurprisingly, given the ancestry of Linux, you will find that other UNIX-like operating systems follow the same directory organization. Of course, you do not get the most of knowing this secret unless you actually learn a little about how the directories are organized, along with the contents of files and directories, and where software should be installed and files stored.

A good knowledge of the Linux file system pays dividends to pretty much every system administrator. Because you have full control (through the root or super-user account), this information is invaluable in keeping your system running smoothly and securely.

## Viewing the Linux File System

Look at the layout of a typical Fedora system by using the list directory contents command, `ls`, like this:

```
$ ls /
bin  dev  home  lib          media mnt  proc  sbin    srv  tftpboot  usr
boot etc  initrd lost+found  misc  opt   root  selinux sys  tmp       var
```

### NOTE

This section provides an overview of the Fedora file system. You might find a fewer or greater number of directories than discussed here in your own system. When some software packages are installed, they create new directories. Updating software packages might also remove or change the name of some directories. See Chapter 7, “Managing Software and System Resources,” for more information on installing, upgrading, and removing software from your Linux system.

To get a more detailed picture, use the `tree` command to show the root or base directory layout, along with associated subdirectories, like this (note that your system’s `/usr/src` directory might be somewhat different, depending on the version of Fedora you have installed or if you have updated Fedora with a new kernel, and that not all subdirectories are listed):

```
$ tree -dx /
/
|-- bin
|-- boot
|-- dev
```

```
|-- etc
|   |-- 4Suite
|   |-- X11
|   |-- cron.d
|   |-- ppp
|   |-- rc.d
|   |-- selinux
|   |-- sysconfig
|-- home
|   |-- andrew
|-- lib
|   |-- modules
|-- lost+found
|-- media
|   |-- cdrom
|   |-- floppy
|-- misc
|-- mnt
|   |-- hgfs
|-- net
|-- opt
|-- proc
|-- root
|-- sbin
|-- selinux
|-- srv
|-- sys
|-- tmp
|-- usr
|   |-- X11R6
|   |   |-- bin
|   |   |-- lib
|   |-- bin
|   |-- include
|   |-- lib
|   |-- local
|   |-- sbin
|   |-- share
|   |-- src
|   |   |-- kernels
|   |   |-- redhat
|-- var
|   |-- ftp
|   |-- log
|   |-- spool
```

This example (pruned from more than 30,000 directories) shows the higher-level directories and corresponds to the directories and descriptions in Table 5.1.

**TABLE 5.1** Basic Linux Directories

Name	Description
/	The root directory
/bin	Essential commands
/boot	Boot loader files, Linux kernel
/dev	Device files
/etc	System configuration files
/home	User home directories
/initrd	Initial RAM disk boot support (used during boot time)
/lib	Shared libraries, kernel modules
/lost+found	Directory for recovered files (if found after a file system check)
/media	Mount point for removable media such as DVDs and floppy disks
/mnt	Usual mount point for local, remote file systems
/opt	Add-on software packages
/proc	Kernel information, process control
/root	Super-user (root home)
/sbin	System commands (mostly root only)
/selinux	Holds the data for SELinux, the security component of Fedora
/sys	Real-time information on devices used by the kernel
/tftpboot	Network boot support
/tmp	Temporary files
/usr	Secondary software file hierarchy
/var	Variable data (such as logs); spooled files

Some of the important directories in Table 5.1, such as those containing user and root commands or system configuration files, are discussed in the following sections. You use and edit files under these directories when you use Fedora.

### Use Essential Commands from the /bin and /sbin Directories

The /bin directory (about 5MB if you do a full install) contains essential commands used by the system when running and booting Linux. In general, only the root operator uses the commands in the /sbin directory. Many (though not all) of these commands are *statically* linked; such commands do not depend on software libraries residing under the /lib or /usr/lib directories. Nearly all the other applications on your system are *dynamically* linked—meaning that they require external software libraries (also known as *shared* libraries) in order to run.

**TIP**

Because the system contains dynamically linked applications, you might sometimes get dependency errors when installing or upgrading software packages; in those situations, a supporting library (or application) might not be present. See Chapter 7 for more information on working with dynamically linked applications and other methods of avoiding such problems. Thankfully, “dependency hell” is largely a thing of the past due to programs such as yum.

---

## Store the Booted Kernel and View Stored Devices in the `/boot` and `/dev` Directories

The `/boot` directory contains a compressed version of the Linux kernel (loaded at boot time), along with other files that describe the kernel or provide information for booting Linux. When you rebuild or install a new kernel, the kernel and related files are placed in this directory (see Chapter 39, “Kernel and Module Management,” for more information on rebuilding or installing a kernel).

Linux device files are contained under the `/dev` directory. Note that under Linux, nearly everything on your system is a file. This means that (with the exception of network interfaces; see the note that follows the upcoming list) regular files; directories; hard drive partitions; serial, printer, or USB ports; and video and sound devices all are files!

The `/dev` directory contains more than 7,500 files representing devices that may or may not be in use on your system. Some of the most commonly used devices in this directory include

- IDE (Integrated Drive Electronics) hard drives, such as `/dev/hda` and `/dev/hdb`
- CD-ROM drives; some which are IDE, others which are CD-RW (CD read/write) drives emulated as SCSI (Small Computer Systems Interface) devices, such as `/dev/scd0`
- Serial ports, such as `/dev/ttyS0` for COM1, `/dev/ttyS1` for COM2, and so on
- Pointing devices, including `/dev/input/mice` and others
- Printers, such as `/dev/lp0`

**NOTE**

Network interfaces (such as `eth0` or `ppp0`) are not represented by Linux device files, but are created in memory when activated. See Chapter 18, “Network Connectivity,” for more information.

---

## Use and Edit Files in the `/etc` Directory

More than 65MB of system configuration files and directories reside under the `/etc` directory if you install all the software included with this book. Some major software packages,

such as Apache, OpenSSH, and xinetd, have directories of configuration files under `/etc`. Other important system-related configuration files in `/etc` are

- `fstab`—The file system table is a text file listing each hard drive, CD-ROM, floppy, or other storage device attached to your PC. The table indexes each device’s partition information with a place in your Linux file system (directory layout) and lists other options for each device when used with Linux (see Chapter 39, “Managing the File System”). Nearly all entries in `fstab` can be manipulated by root using the `mount` command.
- `inittab`—The system initialization table defines the default runlevel, also known as *run-control* level or *system state*. Changes to this file can determine whether your system boots to a graphical or text login, as well as whether dial-up remote access is enabled. (You learn about default runlevels in the section “System Services and Runlevels” located in Chapter 15, “Automating Tasks.” See the section “Starting X,” located in Chapter 6, to learn more about changing `inittab` to boot to a graphical interface. The section “Configuring a Dial-In PPP Server” in Chapter 18 discusses editing `inittab` to enable dial-up remote access.)
- `modprobe.conf`—This configuration file contains directions and options used when loading kernel modules to enable various types of hardware, such as sound, USB, networking, and so on (discussed in the section “Managing Modules” in Chapter 39). The contents of this file are used during boot time, and the file can be manually edited or automatically updated by Fedora’s kudzu hardware management tool (if enabled, as you learn later in this section).
- `passwd`—The list of users for the system, along with user account information. The contents of this file can be changed by various programs, such as `useradd` or `chsh`.
- `printcap`—The system’s printer capabilities database (discussed in the section “Overview of Fedora Printing” in Chapter 12, “Printing with Fedora”).
- `shells`—A list of approved shells (command-line interfaces).

One of the most important directories under `/etc` for Fedora is `sysconfig`. This directory contains network activation scripts and hardware- and software-related information:

```
$ tree -afx /etc/sysconfig
/etc/sysconfig
|-- /etc/sysconfig/apm-scripts
|   |-- /etc/sysconfig/apm-scripts/apmscript
|-- /etc/sysconfig/apmd
|-- /etc/sysconfig/auditd
|-- /etc/sysconfig/authconfig
|-- /etc/sysconfig/autofs
|-- /etc/sysconfig/bluetooth
|-- /etc/sysconfig/clock
|-- /etc/sysconfig/console
|-- /etc/sysconfig/crond
```

```

|-- /etc/sysconfig/desktop
|-- /etc/sysconfig/diskdump
|-- /etc/sysconfig/dund
|-- /etc/sysconfig/firstboot
|-- /etc/sysconfig/grub
|-- /etc/sysconfig/harddisks
|-- /etc/sysconfig/hidd
|-- /etc/sysconfig/httpd
|-- /etc/sysconfig/hwconf
|-- /etc/sysconfig/i18n
|-- /etc/sysconfig/init
|-- /etc/sysconfig/installinfo
|-- /etc/sysconfig/iptables
|-- /etc/sysconfig/iptables-config
|-- /etc/sysconfig/irda
|-- /etc/sysconfig/kernel
|-- /etc/sysconfig/keyboard
|-- /etc/sysconfig/kudzu
|-- /etc/sysconfig/lm_sensors
|-- /etc/sysconfig/modules
|-- /etc/sysconfig/mouse
|-- /etc/sysconfig/mouse.BeforeVMwareToolsInstall
|-- /etc/sysconfig/named
|-- /etc/sysconfig/netdump
|-- /etc/sysconfig/netdump_id_dsa
|-- /etc/sysconfig/netdump_id_dsa.pub
|-- /etc/sysconfig/network
|-- /etc/sysconfig/network-scripts
|   |-- /etc/sysconfig/network-scripts/ifcfg-eth0
|   |-- /etc/sysconfig/network-scripts/ifcfg-lo
|   |-- /etc/sysconfig/network-scripts/ifdown -> ../../../../sbin/ifdown
|   |-- /etc/sysconfig/network-scripts/ifdown-aliases
|   |-- /etc/sysconfig/network-scripts/ifdown-bnep
|   |-- /etc/sysconfig/network-scripts/ifdown-eth
|   |-- /etc/sysconfig/network-scripts/ifdown-ipp
|   |-- /etc/sysconfig/network-scripts/ifdown-ipsec
|   |-- /etc/sysconfig/network-scripts/ifdown-ipv6
|   |-- /etc/sysconfig/network-scripts/ifdown-isdn -> ifdown-ipp
|   |-- /etc/sysconfig/network-scripts/ifdown-post
|   |-- /etc/sysconfig/network-scripts/ifdown-ppp
|   |-- /etc/sysconfig/network-scripts/ifdown-sit
|   |-- /etc/sysconfig/network-scripts/ifdown-sl
|   |-- /etc/sysconfig/network-scripts/ifup -> ../../../../sbin/ifup
|   |-- /etc/sysconfig/network-scripts/ifup-aliases
|   |-- /etc/sysconfig/network-scripts/ifup-bnep

```

```
| |-- /etc/sysconfig/network-scripts/ifup-eth
| |-- /etc/sysconfig/network-scripts/ifup-ipp
| |-- /etc/sysconfig/network-scripts/ifup-ipsec
| |-- /etc/sysconfig/network-scripts/ifup-ipv6
| |-- /etc/sysconfig/network-scripts/ifup-ipx
| |-- /etc/sysconfig/network-scripts/ifup-isdn -> ifup-ipp
| |-- /etc/sysconfig/network-scripts/ifup-plip
| |-- /etc/sysconfig/network-scripts/ifup-plusb
| |-- /etc/sysconfig/network-scripts/ifup-post
| |-- /etc/sysconfig/network-scripts/ifup-ppp
| |-- /etc/sysconfig/network-scripts/ifup-routes
| |-- /etc/sysconfig/network-scripts/ifup-sit
| |-- /etc/sysconfig/network-scripts/ifup-sl
| |-- /etc/sysconfig/network-scripts/ifup-wireless
| |-- /etc/sysconfig/network-scripts/init.ipv6-global
| |-- /etc/sysconfig/network-scripts/network-functions
| |-- /etc/sysconfig/network-scripts/network-functions-ipv6
|-- /etc/sysconfig/networking
| |-- /etc/sysconfig/networking/devices
| |   |-- /etc/sysconfig/networking/devices/ifcfg-eth0
| |   |-- /etc/sysconfig/networking/profiles
| |     |-- /etc/sysconfig/networking/profiles/default
| |       |-- /etc/sysconfig/networking/profiles/default/hosts
| |       |-- /etc/sysconfig/networking/profiles/default/ifcfg-eth0
| |       |-- /etc/sysconfig/networking/profiles/default/resolv.conf
|-- /etc/sysconfig/ntpd
|-- /etc/sysconfig/pand
|-- /etc/sysconfig/pcmcia
|-- /etc/sysconfig/prelink
|-- /etc/sysconfig/rhn
| |-- /etc/sysconfig/rhn/clientCaps.d
| |-- /etc/sysconfig/rhn/rhn-applet
| |-- /etc/sysconfig/rhn/rhnsd
| |-- /etc/sysconfig/rhn/sources
| |-- /etc/sysconfig/rhn/up2date
| |-- /etc/sysconfig/rhn/up2date-keyring.gpg
| |   |-- /etc/sysconfig/rhn/up2date-uuid
|-- /etc/sysconfig/samba
|-- /etc/sysconfig/saslauthd
|-- /etc/sysconfig/selinux -> /etc/selinux/config
|-- /etc/sysconfig/sendmail
|-- /etc/sysconfig/spamassassin
|-- /etc/sysconfig/squid
|-- /etc/sysconfig/syslog
|-- /etc/sysconfig/system-config-securitylevel
```

```
|-- /etc/sysconfig/system-config-users
|-- /etc/sysconfig/tux
|-- /etc/sysconfig/vncservers
`-- /etc/sysconfig/xinetd
```

10 directories, 94 files

For brevity, not all directories and files are discussed here. `/etc/sysconfig` contains many different hardware and software settings critical to the operation of your Fedora system. Knowing the location and contents of these files can be helpful if you need to troubleshoot new hardware configurations.

The settings in various files under `/etc/sysconfig` (such as keyboard, mouse, sound, and so on) are usually created automatically by a related Fedora graphical or console-based configuration utility.

These contents might dynamically change if you use the kudzu hardware configuration service. The kudzu service also prompts you at boot time to remove, configure, or ignore a related setting if kudzu detects new or different hardware (such as a new USB keyboard, network card, or monitor). kudzu creates a file called `hwconf` that contains a hardware profile of your PC's current state. Note that if kudzu is not enabled or running, you can use device-specific configuration utilities such as `system-config-keyboard`, or you can manually edit configuration files.

Information about the type of pointing device attached to the PC, for example, is contained in the file `/etc/sysconfig/mouse`:

```
MOUSETYPE="ps/2"
XMOUSETYPE="PS/2"
FULLNAME="Generic 3 Button Mouse (PS/2)"
XEMU3=no
```

If a different mouse, say a three-button USB device, is attached to the computer, you can edit this information to reflect the hardware change:

```
MOUSETYPE="ps/2"
XMOUSETYPE="IMPS/2"
FULLNAME="Generic 3 Button Mouse (USB)"
XEMU3=no
```

#### CAUTION

If you are new to Linux, the `system-config-mouse` client is the best tool to use to configure a new mouse. You should manually edit system hardware configuration files used by graphical management clients only as a last resort.

---

## Protect the Contents of User Directories—/home

The most important data on a Linux system resides in user's directories, found under the /home directory. Segregating the system and user data can be helpful in preventing data loss and making the process of backing up easier. For example, having user data reside on a separate file system or mounted from a remote computer on the network might help shield users from data loss in the event of a system hardware failure.

## Use the Contents of the /proc Directory to Interact with the Kernel

The content of the /proc directory is created from memory and only exists while Linux is running. This directory contains special "files" that either extract information from or send information to the kernel. Many Linux utilities extract information from dynamically created directories and files under this directory, also known as a *virtual file system*. For example, the free command obtains its information from a file named meminfo:

```
$ free
```

	total	used	free	shared	buffers	cached
Mem:	255040	250752	4288	0	1716	90964
-/+ buffers/cache:		158072	96968			
Swap:	524280	760	523520			

This information constantly changes as the system is used. You can get the same information by using the cat command to see the contents of the meminfo file:

```
$ cat /proc/meminfo
MemTotal:      255040 kB
MemFree:       4412 kB
Buffers:       2164 kB
Cached:        90580 kB
SwapCached:    0 kB
Active:        208884 kB
Inactive:      8976 kB
HighTotal:     0 kB
HighFree:      0 kB
LowTotal:      255040 kB
LowFree:       4412 kB
SwapTotal:     524280 kB
SwapFree:      523520 kB
Dirty:         48 kB
Writeback:     0 kB
Mapped:        201520 kB
Slab:          24772 kB
CommitLimit:   651800 kB
Committed_AS: 341544 kB
PageTables:    2880 kB
VmallocTotal:  770040 kB
VmallocUsed:   3432 kB
```

```
VmallocChunk: 762312 kB
HugePages_Total: 0
HugePages_Free: 0
Hugepagesize: 4096 kB
```

The `/proc` directory can also be used to dynamically alter the behavior of a running Linux kernel by echoing numerical values to specific files under the `/proc/sys` directory. For example, to turn on kernel protection against one type of denial of service (DOS) attack known as *SYN flooding*, use the `echo` command to send the number 1 (one) to the following `/proc` path:

```
# echo 1 >/proc/sys/net/ipv4/tcp_syncookies
```

#### NOTE

The Linux kernel has a number of built-in protections, but good system administration security policies and a secure firewall protecting your gateway, router, or Internet-connected system are the best protection you can use. See the section “Securing Your Network” in Chapter 18 for an overview of firewalling and examples of how to implement Red Hat’s network security tools included with Fedora.

Other ways to use the `/proc` directory include

- Getting CPU information, such as the family, type, and speed from `/proc/cpuinfo`.
- Viewing important networking information under `/proc/net`, such as active interfaces information under `/proc/net/dev`, routing information in `/proc/net/route`, and network statistics in `/proc/net/netstat`.
- Retrieving file system information.
- Reporting media mount point information via USB; for example, the Linux kernel reports what device to use to access files (such as `/dev/sda`) if a USB camera or hard drive is detected on the system. You can use the `dmesg` command to see this information or find information about these devices under the Device File System directory `/proc/devfs` (see the Linux Devfs FAQ at <http://www.atnf.csiro.au/~rgooch/linux/docs/devfs.html> if the source code for the Linux kernel is installed). The file `/usr/src/linux-2.6/Documentation/usb/proc_usb_info.txt` contains general information about USB and the `/proc` directory, as well as what to expect in files under this directory. Note that `devfs` might be supported, but will generally be obsolete in the 2.6 kernel because `/proc/udev` replaces it as a way of managing hot-plug devices on your system.
- Getting the kernel version in `/proc/version`, performance information such as uptime in `/proc/uptime`, or other statistics such as CPU load, swap file usage, and processes in `/proc/stat`.

## Work with Shared Data in the /usr Directory

The /usr directory (nearly 5GB in size if you do a full install) contains software applications, libraries, and other types of shared data for use by anyone on the system. Many Linux system administrators give /usr its own partition. A number of subdirectories under /usr contain the X Window System (/usr/X11R6), man pages (/usr/share/man), software package shared files (/usr/share/*name\_of\_package*, such as /usr/share/emacs), additional application or software package documentation (/usr/share/doc), and an entire subdirectory tree of locally built and installed software, /usr/local.

## Temporary File Storage in the /tmp Directory

As its name implies, the /tmp directory is used for temporary file storage; as you use Linux, various programs create files in this directory. The /tmp directory is cleaned of stale files each day by the tmpwatch command. (A *stale* file is any file not used after 10 days.) Fedora is configured by default to use tmpwatch to check /tmp each day by settings in your system's scheduling table, /etc/crontab.

## Access Variable Data Files in the /var Directory

The /var directory contains subdirectories used by various system services for spooling and logging. Many of these variable data files, such as print spooler queues, are temporary, whereas others, such as system and kernel logs, are renamed and rotated in use. Incoming electronic mail is usually directed to files under /var/spool/mail.

Linux also uses /var for other important system services. These include the topmost File Transfer Protocol (*FTP*) directory under /var/ftp (see Chapter 24, "Remote File Serving with FTP"), and the Apache web server's initial home page directory for the system, /var/www/html. (See Chapter 21, "Apache Web Server Management," for more information on using Apache.)

## Logging In to and Working with Linux

You can access and use a Linux system in a number of ways. One way is at the console with a monitor, keyboard, and mouse attached to the PC. Another way is via a serial console, either by dial-up via a modem or a PC running a terminal emulator and connected to the Linux PC via a null modem cable. You can also connect to your system through a wired or wireless network using the telnet or ssh commands. The information in this section shows you how to access and use the Linux system using physical and remote text-based logins.

### NOTE

This chapter focuses on text-based logins and use of Linux. Graphical logins and using a graphical desktop are described in the section "Starting X" in Chapter 6.

## Text-Based Console Login

If you sit down at your PC and log in to a Linux system that has not been booted to a graphical login, you see a prompt similar to this one:

```
Fedora release 4 (Stentz)
Kernel 2.6.13-1.1532_FC4 on an i686
```

login:

Your prompt might vary, depending on the version of Fedora you are using. In any event, at this prompt, type in your username and press Enter. When you are prompted for your password, type it in and press Enter.

### NOTE

Note that your password is not echoed back to you, which is a good idea. Why is it a good idea? Well, people are prevented from looking over your shoulder and seeing your screen input. It is not difficult to guess that a five-letter password might correspond to the user's spouse's first name!

## Working with Virtual Consoles

After logging in, you are using an interactive command prompt known as a *shell* in the Linux text-based or *console* mode. While you are sitting at your command prompt, you can also use one or more *virtual* consoles or terminals. Virtual consoles allow you to log in to Linux multiple times. (Each login is called a *session*.) This can be useful if you are not using a graphical desktop, but want to use several interactive programs, such as a text editor and web browser, at the same time. To do so, after you log in, run a program and then jump to another login prompt, log in, and start another session. Linux supports 63 virtual consoles, but only the first 6 are configured for use. (You can use 7 if you do not run X11.) Here's how to use virtual Linux consoles:

1. Log in. You use the first virtual console, or vt1 by default.
2. Press F2. You should then see another login prompt. Log in again, and you are then using vt2, the second Linux console.
3. Press Alt+F1 to jump back to vt1.
4. Press Alt+F2 to jump back to vt2.

You can jump back and forth between sessions by using the Alt key plus the F key number of the desired session, such as F3, F4, F5, or F6.

One caveat when using virtual consoles is that there is a default limit on the available number (usually six) if an active X Window session is occupying vt7. To jump to a virtual console from an X session, press Ctrl+Alt+F2; you will be at vt2. You can then jump back to your X session from the text console by pressing Alt+F7 (to go to vt7, in use by X). You

should also be careful to save any work in progress before you exit each session and to log out of each session when finished. If you do not, you could leave an open login and shell prompt available at the keyboard to anyone who walks by!

#### NOTE

In addition to virtual console keystrokes, the Linux console might also recognize the three-fingered salute (or Vulcan neck pinch), Ctrl+Alt+Del. This behavior (and the number of virtual terminals) can be controlled by the system administrator by editing the system's *initialization table*, `/etc/inittab`. See the Keyboard and Console HOWTO at <http://www.tldp.org/> for more details.

## Using Simple Keyboard and Mouse Techniques in a Linux Console Session

Working with Linux in a console-based session usually involves entering commands from the keyboard. However, you can also use simple mouse controls as well. Linux keyboard combinations and mouse support help provide virtual console navigation, start special system actions (such as rebooting or shutting down), provide shortcuts to save typing, and can aid in reading files or viewing program output.

For example, you can scroll the contents of your screen from the console by pressing Shift+PageUp or Shift+PageDown, and can copy and paste text using your mouse buttons. This section shows you how to access default or custom menus at the text console, which can be helpful to get system information or to launch new programs.

If you use a mouse with Linux (and you most likely do), you can use your pointing device for copy and paste operations. This support is provided by `gpm`, the general purpose mouse server. The `gpm` server must be enabled or started while booting Linux (see Chapter 15 for more details in the section “Controlling Services at Boot Using Administrative Tools”). To copy a section of text, click and drag text with the left mouse button (button 1) held down. To paste text, click an insertion point, and then press the middle mouse button (button 2).

Button assignment, like all mouse controls during text console use, is managed by command-line options given to `gpm` when it is started. For example, if you look at the `gpm` startup script named `gpm` under the `/etc/rc.d/init.d/` directory, you will see that it uses the file named `mouse` under the `/etc/sysconfig/` directory to hold options:

```
# Additional options for gpm (e.g. acceleration), device
OPTIONS=""
DEVICE="/dev/mouse"
```

You can add options, detailed in the `gpm` man page, to change how your mouse works, enable or disable features, or assign special commands to a specific mouse button click. For example, to change your button order from 123 (left, middle, and right) to 321, edit the `/etc/sysconfig/mouse` file as root and change the `OPTIONS` entry like so:

```
OPTIONS="-B 321"
```

After saving your changes, restart `gpm` like so:

```
# /etc/rc.d/init.d/gpm restart
```

Your mouse buttons are now reversed!

To aid users with a two-button mouse, Linux supports three-button *emulation*; emulation lets users simultaneously press the right and left mouse buttons to simulate a press of the middle button. You can enable this feature during installation or by using the `mouseconfig` command. Refer to the section “Configuring Pointing Devices in Linux” in Chapter 4, “Post-Installation Configuration,” to see how to use `mouseconfig`.

The `gpm` server also provides the ability to reboot or shut down the system with the mouse. Depending on the combination of mouse buttons you press, you have several reboot or shutdown options. Begin by holding down either the left or right mouse button and triple-clicking the opposite button; depending on which mouse button you press next, one of these actions occurs:

- Pressing the left button causes an immediate reboot using the `init` command.
- Pressing the middle button reboots the system using the `shutdown` command.
- Pressing the right button causes the system to shut down immediately with the `shutdown` command.

You can also create custom menus that pop up at a text-based Linux console by editing the file `/etc/gpm-root.conf` (as root) and starting the `gpm-root` command. When you run `gpm-root` without making changes to its configuration file, by default a system status dialog (with the date, time, CPU load, free memory, and swap file usage) appears if you hold down the `Ctrl` key and press the middle mouse button.

You can change your keyboard layouts by using the `loadkeys` command. To use a different font for the console, try the `setfont` command. Fedora Linux comes with nearly 150 different console fonts, which are found under the `/lib/kbd/consolefonts` directory. Refer to the section “Configuring Keyboards with Linux” in Chapter 4 to see how to use these commands.

#### NOTE

A text-based, dial-up login, also known as a *shell* account, looks much the same as a text-based login at a PC running Linux. Details about setting up Linux to answer the phone and provide a login prompt via a modem are in Chapter 18. Using dial-up access has some limitations, such as the inability to use virtual consoles. From a shell account, however, you can start programs in the background (using the ampersand, `&`), run programs after logging out with the `nohup` command, or use the `screen` command to simulate virtual terminals (an approach that works much like using virtual consoles). For more information on different shells included with Linux, see Chapter 15.

---

## Logging Out

Use the `exit` or `logout` commands to exit your session. Type the command and press Enter. You are then returned to the login prompt. If you use virtual consoles, remember to exit each console before leaving your PC. (Otherwise, someone could easily sit down and use your account.)

## Logging In and Out from a Remote Computer

Although you can happily log in on your computer, an act known as a *local* log in, you can also log in to your computer via a network connection from a remote computer. Linux-based operating systems provide a number of remote access commands you can use to log in to other computers on your local area network (*LAN*), wide area network (*WAN*), or the Internet. Note that not only must you have an account on the remote computer, but the remote computer must be configured to support remote logins—otherwise, you won't be able to log in.

### NOTE

See Chapter 18 to see how to set up network interfaces with Linux to support remote network logins and Chapter 15 to see how to start remote access services (such as `ssh`).

The best and most secure way (barring future exploits) to log in to a remote Linux computer is to use the `ssh` or Secure Shell client. Your login and session are encrypted while you work on the remote computer. The `ssh` client features many different command-line options, but can be simply used with the name of the remote computer, like this:

```
[andrew@laptop ~]$ ssh desktop
```

```
The authenticity of host 'desktop (192.168.2.3)' can't be established.  
RSA key fingerprint is 91:7d:74:4b:1c:a1:96:06:ba:2f:d4:cf:78:44:ff:d7.  
Are you sure you want to continue connecting (yes/no)? yes
```

The first time you connect with a remote computer using `ssh`, Linux displays the remote computer's encrypted identity key and asks you to verify the connection. After you type **yes** and press Enter, you are warned that the remote computer's identity (key) has been entered in a file named `known_hosts` under the `.ssh` directory in your home directory. You are also prompted to enter your password:

```
Warning: Permanently added 'desktop,192.168.2.3' (RSA) \  
to the list of known hosts.  
andrew@'desktop's password:  
/usr/X11R6/bin/xauth: creating new authority file /home/winky/.Xauthority  
[andrew@desktop andrew]$
```

After entering your password, you can then work on the remote computer. Again, everything you enter on the keyboard in communication with the remote computer is encrypted. Use the `exit` or `logout` commands to exit your session and return to the shell on your computer.

#### CAUTION

The next remote access command, `telnet`, is shown as an example because it is included with most Linux distributions, but you shouldn't use it: `telnet` transmits your username and password in clear text across the network, posing a huge security risk for your system. Also, note that this service must be explicitly turned on and allowed on the remote computer (by editing the file named `telnet` under the `/etc/xinetd.d` directory and then restarting `xinetd`; see the section "Starting and Stopping Services Manually" in Chapter 15 for more information on starting or restarting a system service).

The `telnet` command can be used, along with the name of a remote host or Internet Protocol (*IP*) address, to log in to a remote computer. For example, to log in to the host named `desktop` from the host named `laptop`, you would enter the following:

```
[andrew@laptop andrew]$ telnet desktop
```

After you press Enter, you see some information presented by the remote computer, and you are then prompted for your username on the remote system:

```
Trying 192.168.2.70...
Connected to desktop.andbudson.co.uk (192.168.2.73).
Escape character is '^]'.
```

```
Linux 2.6.10-1.741_FC3 (desktop.andbudson.co.uk) (15:43 on Friday, 4 February 2005)
```

```
login: andrew
```

After you type your username (`andrew` in this example), press Enter, and you are prompted for your password on the remote system:

```
Password:
Last login: Fri Feb 4 15:42:26 from 192.168.2.2
[andrew@desktop andrew]$
```

After you type your password and press Enter (your password is not echoed back), you are informed of the last time you logged in, and you can then work on the remote computer. Use the `exit` or `logout` command to exit your session and return to the shell on your computer.

Although it is possible to use `telnet` to log in to a remote computer over a wired and wireless network, such use is not recommended, especially via the Internet. When you type your username, press Enter, and type your password, your username and password

are transmitted without encryption over the network. Transmitting usernames and passwords over a network without encryption is a bad idea for obvious reasons. However, if you have a physically secure internal network not connected to the Internet, have firewall policies in place, and don't use wireless networking, there is nothing wrong with using `telnet`. In fact, the encryption overhead of using `ssh` can reduce network transmission rates in some cases.

#### NOTE

It is possible to use `telnet` securely over an encrypted Virtual Private Network (VPN), but that is beyond the scope of this chapter and book. Besides, why bother when you can use SSH?

## Changing Your User Information

Linux users are assigned a name, known as a *username*, by the root operator. One method of assigning usernames is to use one's first initial and last name in lowercase; for example, Bernice Hudson would have a username of `bhudson`. Each user must also have a password, which is used with the username either at a graphical or text-based login.

#### NOTE

Older versions of Linux operating systems limited the length of usernames to 8 characters. The current version of Fedora limits usernames to 32 characters. Good passwords should be a minimum of 8 characters long and contain uppercase and lowercase letters, along with numbers. Random passwords for users can be generated using the `mkpasswd` command (which is included with the `expect` software package). For example, to generate a 10-character password automatically with three numbers and three digits, use `mkpasswd -l 10 -d 3 -C 3`. Good passwords are not birthdays, anniversaries, your pet's name, the name of your significant other, or the model of your first car!

You cannot change your username, but you can change your user information, such as address, phone, and so on. You make these changes using the `chfn` or *change finger information* command. This command modifies the contents of your entry in the system password file `/etc/passwd`, which is used by the `finger` command to display information about a system's user. For example, type **`chfn`** at the command line and press Enter:

```
$ chfn
Changing finger information for bhudson.
Password:
Name []: Bernice Hudson
Office []: Suite 56 N. Centennial Blvd.
Office Phone []: 919 555-1212
Home Phone []: 919 555-1213
Finger information changed.
```



```
partprobe      (8) - inform the OS of partition table changes
pvcreate       (8) - initialize a disk or partition for use by LVM
sfdisk         (8) - Partition table manipulator for Linux
```

To find a command and its documentation, you can use the `whereis` command. For example, if you are looking for the `fdisk` command, you can do this:

```
$ whereis fdisk
fdisk: /sbin/fdisk /usr/share/man/man8/fdisk.8.gz
```

## Using Man Pages

To learn more about a command or program, use the `man` command, followed by the name of the command. Man pages for Linux and X Window commands are within the `/usr/share/man`, `/usr/local/share/man`, and `/usr/X11R6/man` directories; so, for example, to read the `rm` command's man page, use the `man` command like this:

```
$ man rm
```

After you press Enter, the `less` command (a Linux command known as a *pager*) displays the man page. The `less` command is a text browser you can use to scroll forward and backward (even sideways) through the document to learn more about the command. Type the letter `h` to get help, use the forward slash to enter a search string, or press `q` to quit.

### NOTE

Although nearly all the hundreds of GNU commands included with Linux each have a man page, detailed information about using a GNU command must be read using the `info` command. For example, to learn even more about `bash` (which has a rather extensive manual page), use the `info` command like this:

```
$ info bash
```

Press the `n` and `p` keys to navigate through the document, or scroll down to a menu item on the screen and press Enter to read about a specific feature. Press `q` to quit reading.

## Finding and Reading Software Package Documentation

Documentation for various software packages is included in the `/usr/share/doc` directory; that directory is stored in another directory that's labeled with the associated package's name. You can find other Linux documentation, known as HOWTOs and Frequently Asked Questions (*FAQs*), online by browsing to <http://www.tldp.org/>. HOWTO documents contain specific information related to a particular subject, such as printing, setting up a network, programming a serial port, or using a CD-ROM drive with Linux. These documents can be read by using your web browser. Of course, one of the best online tools you can use is a good search engine, such as Google.

You can read document formats such as text with `less` or another pager or text reader. For example, to read a copy of the GNU General Public License (*GPL*), a file named `GPL_V2` under the `/usr/share/apps/LICENSES` directory, use `less` like this:

```
$ less /usr/share/apps/LICENSES/GPL_V2
```

After you press `Enter`, you can scroll back and forth through the file. Press `q` to quit reading. If a document is in compressed form (ending in `.gz`), use the `zless` pager, which decompresses a document first:

```
$ zless /usr/share/man/es/man1/README.gz
```

Most users read document formats such as HTML using a web browser in a graphical desktop. Fedora includes at least two versatile text-based web browsers, however, accessed with the `lynx` and `links` commands. To browse an HTML file on your system without using X11, use either command, along with the path to the file. For example, to read an HTML version of the GNU GPL with `links`, use the command like this:

```
$ links /usr/share/doc/HTML/en/common/gpl-license.html
```

After you press `Enter`, use your `Up` and `Down` cursor keys to scroll back and forth through the file. Press `q`, `Enter` to quit reading. If you have configured a mouse, click the left button near the top of the screen, and `links` displays its menus.

## Using the Shell

The shell is an interactive command prompt with many different features:

- Input and output redirection
- Background processing
- Job control
- History editing
- Built-in help
- Command-line completion
- Command-line editing

The shell interprets keyboard commands and is generally used to launch other commands or programs using the shell's interpreter language known as *shell scripts*.

### NOTE

Shell scripts are discussed in Chapter 15.

The shell you use is assigned by the last field in your entry in the system's `/etc/passwd` file. This example, for a user named `andrew`, shows that the login shell is `bash`:

```
andrew:x:502:502::/home/andrew:/bin/bash
```

The default shell for most Linux distributions, including Fedora, is the GNU `bash` or Bourne Again SHell, but other shells, such as `tcsh`, `ksh`, and `zsh` are available for use. You can use a different shell by typing its name at the command line. Alternatively, the root operator might assign a user to another shell when creating that user account (see “Working As Root,” later in this chapter).

#### CAUTION

If you are interested in trying a different shell with Linux, you can change your login shell using the `chsh` command, but make sure that the shell is actually installed on your system. For example, to change your default shell to `tcsh`, first use the `which` command to verify that it is installed:

```
$ which tcsh
/bin/tcsh
```

This example shows that the `tcsh` shell is installed under the `/bin` directory. The `tcsh` shell should also be listed in your system's list of approved shells, `/etc/shells`. Check to make sure that it is listed:

```
$ grep tcsh /etc/shells
/bin/tcsh
```

You can also use the `chsh` command's `-l` option to list valid system shells in order to verify that using `tcsh` is allowed. Because `tcsh` is installed and listed in `/etc/shells`, you can then change your shell using the `chsh` command:

```
$ chsh
Changing shell for andrew.
Password:
New shell [/bin/bash]: /bin/ksh
chsh: "/bin/ksh" does not exist.
```

Note that the `chsh` command reports an error if you enter the name of a shell not installed on your system.

```
$ chsh -s /bin/tcsh
Changing shell for andrew.
Password:
Shell changed.
```

If you now take a look at your `/etc/passwd` entry, you will see `/bin/tcsh` as your default shell. The next time you log in, you can use `tcsh`.

## Using Environment Variables

A number of in-memory variables are assigned and loaded by default when the user logs in. These variables are known as shell *environment variables*, which can be used by various commands to get information about your environment, such as the type of system you are running, your home directory, and the shell in use. Environment variables are used by Linux operating systems to help tailor the computing environment of your system, and include helpful specifications and setup, such as default locations of executable files and software libraries. If you begin writing shell scripts, you might use environment variables in your scripts. Until then, you only need to be aware of what environment variables are and do.

The following list includes a number of environment variables, along with descriptions of how the shell uses them:

- **PWD**—To provide the name of the current working directory, used by the `pwd` command (such as `/home/andrew/foo`)
- **USER**—To declare the user's name, such as `andrew`
- **LANG**—To set language defaults, such as English
- **SHELL**—To declare the name and location of the current shell, such as `/bin/bash`
- **PATH**—To set the default location of executable files, such as `/bin`, `/usr/bin`, and so on
- **LD\_LIBRARY\_PATH**—To declare the location of important software libraries (because most, but not all, Linux commands use shared resources)
- **TERM**—To set the type of terminal in use, such as `vt100`, which can be important when using screen-oriented programs, such as text editors
- **MACHINE**—To declare system type, system architecture, and so on

### NOTE

Each shell can have its own feature set and language syntax, as well as a unique set of default environment variables. See Chapter 15 for more information about using the different shells included with Fedora.

At the command line, you can use the `env` or `printenv` commands to display these environment variables, like so:

```
$ env
```

```
PWD=/home/andrew
HOSTNAME=laptop.andbudson.co.uk
USER=andrew
MACHTYPE=i386-redhat-linux-gnu
MAIL=/var/spool/mail/andrew
```

```
BASH_ENV=/home/andrew/.bashrc
LANG=en_GB
DISPLAY=:0
LOGNAME=andrew
SHLVL=1
PATH=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin: \
/usr/X11R6/bin:/home/andrew/bin
SHELL=/bin/bash
HOSTTYPE=i386
OSTYPE=linux-gnu
HISTSIZE=1000
TERM=xterm
HOME=/home/andrew
```

This abbreviated list shows a few common variables. These variables are set by configuration or *resource* files contained in the `/etc`, `/etc/skel`, or user `/home` directory. You can find default settings for `bash`, for example, in `/etc/profile`, `/etc/bashrc`, `.bashrc`, or `.bash_profile` files installed in your home directory. Read the man page for `bash` for details about using these configuration files.

One of the most important environment variables is `$PATH`, which defines the location of executable files. For example: If, as a regular user, you try to use a command that is not located in your `$PATH` (such as the `ifconfig` command), you will see something like this:

```
$ ifconfig
-bash: ifconfig: command not found
```

However, you might know that `ifconfig` is definitely installed on your system, and you can verify this using the `whereis` command like so:

```
$ whereis ifconfig
ifconfig: /sbin/ifconfig /usr/share/man/man8/ifconfig.8.gz
```

You can also run the command by typing its full pathname, or complete directory specification like this:

```
$ /sbin/ifconfig
```

As you can see in this example, the `ifconfig` command is indeed installed. What happened is that by default, the `/sbin` directory is not in your `$PATH`. One of the reasons for this is that commands under the `/sbin` directory are normally intended to be run only by root. You can add `/sbin` to your `$PATH` by editing the file `.bash_profile` in your home directory (if you use the `bash` shell by default, like most Linux users). Look for the following line:

```
PATH=$PATH:$HOME/bin
```

You can then edit this file, perhaps using the vi editor (discussed in this chapter), to add the /sbin directory like so:

```
PATH=$PATH:/sbin:$HOME/bin
```

Save the file. The next time you log in, the /sbin directory is in your \$PATH. One way to use this change right away is to read in the new settings in .bash\_profile by using the bash shell's source command like so:

```
$ source .bash_profile
```

You can now run ifconfig without the need to explicitly type its full pathname.

Some Linux commands also use environment variables, for example, to acquire configuration information (such as a communications program looking for a variable such as BAUD\_RATE, which might denote a default modem speed).

To experiment with the environment variables, you can modify the PS1 variable to manipulate the appearance of your shell prompt. If you are working with bash, you can use its built-in export command to change the shell prompt. For example, if your default shell prompt looks like

```
[andrew@laptop ~]$
```

You can change its appearance by using the PS1 variable like this:

```
$ PS1='$OSTYPE r00lz ->'
```

After you press Enter, you see

```
linux-gnu r00lz ->
```

#### NOTE

See the bash man page for other variables you can use for prompt settings.

## Navigating and Searching with the Shell

Use the cd command (built into the shell) to navigate through the Fedora file system. This command is generally used with a specific directory location, or pathname like this:

```
$ cd /usr/X11R6/lib/X11/doc
```

Under Fedora, the cd command can also be used with several shortcuts. For example, to quickly move up to the *parent* (higher-level) directory, use the cd command like this:

```
$ cd ..
```

To return to one's home directory from anywhere in the Linux file system, use the `cd` command like this:

```
$ cd
```

You can also use the `$HOME` shell environment variable to accomplish the same thing. Type this command and press Enter to return to your home directory:

```
$ cd $HOME
```

You can accomplish the same thing by using the tilde (`~`) like this:

```
$ cd ~
```

Linux also includes a number of GNU commands you can use to search the file system. These include

- `whereis command`—Returns the location of the command and its man page.
- `whatis command`—Returns a one-line synopsis from the command's man page.
- `locate file`—Returns locations of all matching file(s); an extremely fast method of searching your system because `locate` searches a database containing an index of all files on your system. However, this database (about 4MB in size and named `slocate.db` under the `/var/lib/slocate` directory) is built daily at 4:20 a.m. by default, and does not contain pathnames to files created during the workday or in the evening. If you do not keep your machine on constantly, you can run the `updatedb` command as the super-user to manually start the building of the database.
- `apropos subject`—Returns a list of commands related to subject.

## Managing Files with the Shell

Managing files in your home directory involves using one or more easily remembered commands. If you have any familiarity with the now-ancient DOS, you recognize some of these commands (although their names are different from those you remember). Basic file management operations include paging (reading), moving, renaming, copying, searching, and deleting files and directories. These commands include

- `cat filename`—Outputs contents of *filename* to display
- `less filename`—Allows scrolling while reading contents of *filename*
- `mv file1 file2`—Renames *file1* to *file2*
- `mv file dir`—Moves file to specified directory
- `cp file1 file2`—Copies *file1* and creates *file2*
- `rm file`—Deletes file

- `rmdir dir`—Deletes directory (if empty)
- `grep string file(s)`—Searches through files(s) and displays lines containing matching string

Note that each of these commands can be used with pattern-matching strings known as *wildcards* or *expressions*. For example, to delete all files in the current directory beginning with the letters `abc`, you can use an expression beginning with the first three letters of the desired filenames. An asterisk (\*) is then appended to match all these files. Use a command line with the `rm` command like this:

```
$ rm abc*
```

Linux shells recognize many types of filenaming wildcards, but this is different from the capabilities of Linux commands supporting the use of more complex expressions. You learn more about using wildcards in Chapter 15.

#### NOTE

Learn more about using expressions by reading the `ex` or `grep` man pages.

## Compressing and Decompressing Files Through the Shell

Another file management operation is compression and decompression of files, or the creation, listing, and expansion of file and directory archives. Linux distributions usually include several compression utilities that you can use to create, compress, expand, or list the contents of compressed files and archives. These commands include

- `bunzip2`—Expands a compressed file
- `bzip2`—Compresses or expands files and directories
- `gunzip`—Expands a compressed file
- `gzip`—Compresses or expands files and directories
- `shar file`—Creates a shell archive of files
- `tar`—Creates, expands, or lists the contents of compressed or uncompressed file or directory archives known as *tape archives* or *tarballs*
- `unshar`—Reassembles files from the shell archive
- `uudecode file.uu`—Decodes an uuencoded text file to its binary form
- `uuencode file`—Encodes a binary file to text file format for transmission via email

**NOTE**

The `rpm` command can also be used to manage archived data, but is generally used for software development and management. See Chapter 8 for more information on using RPM.

Most of these commands are easy to use. The `tar` command, however, has a somewhat complex (although capable) set of command-line options and syntax. Even so, you can quickly learn to use `tar` by remembering a few simple invocations on the command line. For example, to create a compressed archive of a directory, use `tar`'s `czf` options like this:

```
$ tar czf dirname.tgz dirname
```

The result is a compressed archive (a file ending in `.tgz`) of the specified directory (and all files and directories under it). Add the letter `v` to the preceding options to view the list of files added during compression and archiving. To list the contents of the compressed archive, substitute the `c` option with the letter `t`, like this:

```
$ tar tzf archive
```

Of course, if many files are in the archive, a better invocation (to easily read or scroll through the output) is

```
$ tar tzf archive | less
```

To expand the contents of a compressed archive, use `tar`'s `zxf` options, like so:

```
$ tar zxf archive
```

`tar` decompresses the specified archive and extracts the contents in the current directory.

## Using the Text Editors

Linux distributions include a number of applications known as *text editors* that you can use to create text files or edit system configuration files. Text editors are similar to word processing programs, but generally have fewer features, work only with text files, and might or might not support spell checking or formatting. The text editors range in features and ease of use, but are found on nearly every Linux distribution. The number of editors installed on your system depends on what software packages you've installed on the system.

Some of the console-based text editors are

- `ed`—A simple line editor without cursor support
- `emacs`—The comprehensive GNU `emacs` editing environment, which is much more than an editor; see the section “Working with `emacs`” later in this chapter
- `jed`—A programmer's text editor with features such as colorized highlighting of text to help syntax checking and editing of programs

- `joe`—Joe’s Own Editor, a text editor, which can be used to emulate other editors
- `mcedit`—A DOS-like text editor for UNIX-like systems
- `nano`—A simple text editor similar to the `pico` text editor included with the `pine` email program
- `sed`—A stream editor usually used in shell scripts (discussed in Chapter 15)
- `vim`—An improved, compatible version of the `vi` text editor (which we call `vi` in the rest of this chapter because it has a symbolic link named `vi` and a symbolically linked manual page)

Note that not all text editors described here are *screen-oriented*; editors such as `ed` and `sed` work on a line-by-line basis, or a stream of text, and do not support movement of a cursor on the screen. Some of the text editors for the X Window System, which provide a graphical interface, such as menu bars, buttons, scrollbars and so on, are

- `gedit`—A GUI text editor for GNOME
- `kate`—A simple KDE text editor
- `kedit`—A simple KDE text editor
- `nedit`—A programming text editor
- `kwrite`—A simple KDE text editor

A good reason to learn how to use a text-based editor, such as `vi`, is that system maintenance and recovery operations generally never take place during X Window sessions (negating the use of a GUI editor). Many larger, more complex and capable editors do not work when Linux is booted to its single-user or maintenance mode. See Chapter 15 for more information about how Fedora boots. If anything does go wrong with your system, you probably won’t be able to get into the X Window system, making knowledge and experience of using both the command line and text editors such as `vi` important. Make a point of opening some of the editors and playing around with them; you never know, you might just thank me someday!

Another reason to learn how to use a text-based editor under the Linux console mode is so that you can edit text files through dial-up or network shell sessions because many servers do not host graphical desktops.

## Working with `vi`

The editor found on nearly every UNIX and Linux system is, without a doubt, the `vi` editor, originally written by Bill Joy. This simple-to-use but incredibly capable editor features a somewhat cryptic command set, but you can put it to use with only a few commands. Although more experienced UNIX and Linux users continue to use `vi` extensively during computing sessions, many newer users might prefer learning an easier-to-use text editor such as `pico` or GNU `nano`. Die-hard GNU fans and programmers definitely use `emacs`.

That said, learning how to use `vi` is a good idea. You might need to edit files on a Linux system with a minimal install, or a remote server without a more extensive offering of installed text editors. Chances are better than good that `vi` will be available.

You can start an editing session by using the `vi` command like this:

```
$ vi file.txt
```

The `vi` command works by using an insert, or editing mode, and a viewing (or command) mode.

When you first start editing, you are in the viewing mode. You can use your cursor or other navigation keys (as shown later) to scroll through the text. To start editing, press the `i` key to insert text or the `a` key to append text. When finished, use the `Esc` key to toggle out of the insert or append modes and into the viewing (or command) mode. To enter a command, type a colon (`:`), followed by the command, such as `w` to write the file, and press `Enter`.

Although `vi` supports many complex editing operations and numerous commands, you can accomplish work by using a few basic commands. These basic `vi` commands are

- **Cursor movement**—`h`, `j`, `k`, `l` (left, down, up, and right)
- **Delete character**—`x`
- **Delete line**—`dd`
- **Mode toggle**—`Esc`, Insert (or `i`)
- **Quit**—`:q`
- **Quit without saving**—`:q!`
- **Run a shell command**—`:sh` (use `'exit'` to return)
- **Save file**—`:w`
- **Text search**—`/`

#### NOTE

Use the `vimtutor` command to quickly learn how to use `vi`'s keyboard commands. The tutorial takes less than 30 minutes, and it teaches new users how to start or stop the editor, navigate files, insert and delete text and perform search, replace, and insert operations.

## Working with emacs

Richard M. Stallman's GNU `emacs` editor, like `vi`, is included with Linux and nearly every other Linux distribution. Unlike other UNIX and Linux text editors, `emacs` is much more than a simple text editor—it is an editing environment and can be used to compile and build programs, act as an electronic diary, appointment book and calendar, compose and

send electronic mail, read Usenet news, and even play games. The reason for this capability is that emacs contains a built-in language interpreter that uses the Elisp (emacs LISP) programming language.

The GNU version of this editor requires more than 30MB of hard drive space. However, there are versions with less resource requirements, and at least one other text editor included with Linux, named `joe`, can be used as an emacs clone (albeit with fewer features).

You can start an emacs editing session like this:

```
$ emacs file.txt
```

#### TIP

If you start emacs when using X11, the editor launches in its own floating window. To force emacs to display inside a terminal window instead of its own window (which can be useful if the window is a login at a remote computer), use the `-nw` command-line option like this: **emacs -nw file.txt**.

The emacs editor uses an extensive set of keystroke and named commands, but you can work with it using a basic command subset. Many of these basic commands require you to hold down the Ctrl key, or to first press a *meta* key (generally mapped to the Alt key). The basic commands are listed in Table 5.2.

**TABLE 5.2**    emacs Editing Commands

Action	Command
Abort	Ctrl+g
Cursor left	Ctrl+b
Cursor down	Ctrl+n
Cursor right	Ctrl+f
Cursor up	Ctrl+p
Delete character	Ctrl+d
Delete line	Ctrl+k
Go to start of line	Ctrl+a
Go to end of line	Ctrl+e
Help	Ctrl+h
Quit	Ctrl+x, Ctrl+c
Save As	Ctrl+x, Ctrl+w
Save file	Ctrl+x, Ctrl+s
Search backward	Ctrl+r
Search forward	Ctrl+s
Start tutorial	Ctrl+h, t
Undo	Ctrl+x, u

**TIP**

One of the best reasons to learn how to use emacs is that you can use nearly all the same keystrokes to edit commands on the bash shell command line. Another reason is that like vi, emacs is universally available on nearly every UNIX and Linux system, including Apple's Mac OS X.

## Working with Permissions

Under Linux (and UNIX), everything in the file system, including directories and devices, is a file. And every file on your system has an accompanying set of permissions based on ownership. These permissions form the basis for security under Linux, and designate each file's read, write, and execute permission for you, members of your group, and all others on the system.

You can examine the default permissions for a file you create by using the `umask` command, or as a practical example, by using the `touch` command and then the `ls` command's long-format listing like this:

```
$ touch file
$ ls -l file
-rw-rw-r-- 1 andrew andrew 0 Nov 11 12:28 file
```

In this example, the `touch` command is used to quickly create a file. The `ls` command then reports on the file, displaying information (from left to right) in the first field of output (such as `-rw-rw-r--` previously):

- **The first character of the field is the type of file created**—Common indicators of the type of file are a leading letter in the output. A blank (which is represented by a dash in the preceding example) designates a plain file, `d` designates a directory, `c` designates a character device (such as `/dev/ttyS0`), and `b` is used for a block device (such as `/dev/hda`).
- **Permissions**—Read, write, and execute permissions for the owner, group, and all others on the system. (You learn more about these permissions later in this section.)
- **Number of links to the file**—The number one (1) designates that there is only one file, whereas any other number indicates that there might be one or more hard-linked files. Links are created with the `ln` command. A hard-linked file is an exact copy of the file, but it might be located elsewhere on the system. Symbolic links of directories can also be created, but only the root operator can create a hard link of a directory.
- **The owner**—The account that created or owns the file; you can change this designation by using the `chown` command.
- **The group**—The group of users allowed to access the file; you can change this designation by using the `chgrp` command.

- **File size and creation/modification date**—The last two elements indicate the size of the file in bytes and the date the file was created or last modified.

## Assigning Permissions

Under Linux, permissions are grouped by owner, group, and others, with read, write, and execute permission assigned to each, like so:

Owner	Group	Others
rwx	rwx	rxw

Permissions can be indicated by mnemonic or octal characters. Mnemonic characters are

- **r** indicates permission for an owner, member of the owner's group, or others to open and read the file.
- **w** indicates permission for an owner, member of the owner's group, or others to open and write to the file.
- **x** indicates permission for an owner, member of the owner's group, or others to execute the file (or read a directory).

In the previous example for the file named `file`, the owner, `andrew`, has read and write permission, as does any member of the group named `andrew`. All other users may only read the file. Also note that default permissions for files created by the root operator will be different! This is because of `umask` settings assigned by the shell.

Many users prefer to represent permissions using numeric codes, based on octal (base 8) values. Here's what these values mean:

- 4 indicates read permission.
- 2 indicates write permission.
- 1 indicates execute permission.

In octal notation, the previous example file has a permission setting of `664` (read+write or `4+2`, read+write or `4+2`, read-only or `4`). Although you can use either form of permissions notation, octal is easy to use quickly after you visualize and understand how permissions are numbered.

### NOTE

In Linux, you can create groups to assign a number of users access to common directories and files, based on permissions. You might assign everyone in accounting to a group named `accounting`, for example, and allow that group access to accounts payable files while disallowing access by other departments. Defined groups are maintained by the root operator, but you can use the `newgrp` command to temporarily join other groups in order to access files (as long as the root operator has added you to the other groups). You can also allow or deny access to your files by other groups by modifying the group permissions of your files.

## Directory Permissions

Directories are also files under Linux. For example, again use the `ls` command to show permissions like this:

```
$ mkdir foo
$ ls -ld foo
drwxrwxr-x    2 andrew    andrew          4096 Jan 23 12:37 foo
```

In this example, the `mkdir` command is used to create a directory. The `ls` command and its `-ld` option is used to show the permissions and other information about the directory (not its contents). Here you can see that the directory has permission values of 775 (read+write+execute or 4+2+1, read+write+execute or 4+2+1, and read+execute or 4+1).

This shows that the owner and group members can read and write to the directory and, because of execute permission, also list the directory's contents. All other users can only list the directory contents. Note that directories require execute permission in order for anyone to be able to view their contents.

You should also notice that the `ls` command's output shows a leading `d` in the permissions field. This letter specifies that this file is a directory; normal files have a blank field in its place. Other files, such as those specifying a block or character device, have a different letter (see the section "Managing Files for Character Devices, Block Devices, and Special Devices" in Chapter 39 for more information about block devices).

For example, if you examine the device file for a Linux serial port, you will see

```
$ ls -l /dev/ttyS0
crw-rw----    1 root      uucp           4,  64 Jan 23 23:38 /dev/ttyS0
```

Here, `/dev/ttyS0` is a character device (such as a serial communications port and designated by a `c`) owned by `root` and available to anyone in the `uucp` group. The device has permissions of `660` (read+write, read+write, no permission).

On the other hand, if you examine the device file for an IDE hard drive, you see

```
$ ls -l /dev/hda
brw-rw----    1 root      disk           3,   0 Jan 23 23:37 /dev/hda
```

In this example, `b` designates a block device (a device that transfers and caches data in blocks) with similar permissions. Other device entries you will run across on your Linux system include symbolic links, designated by `s`.

You can use the `chmod` command to alter a file's permissions. This command uses various forms of command syntax, including octal or a mnemonic form (such as `u`, `g`, `o`, or `a` and `rx`, and so on) to specify a desired change. The `chmod` command can be used to add, remove, or modify file or directory permissions to protect, hide, or open up access to a file by other users (except for `root`, which can access any file or directory on a Linux system).

The mnemonic forms of `chmod`'s options (when used with a plus character, `+`, to add, or a minus sign, `-`, to take away) designate the following:

- u—Adds or removes user (owner) read, write, or execute permission
- g—Adds or removes group read, write, or execute permission
- o—Adds or removes read, write, or execute permission for others not in a file's group
- a—Adds or removes read, write, or execute permission for all users
- r—Adds or removes read permission
- w—Adds or removes write permission
- x—Adds or removes execution permission

For example, if you create a file, such as a `readme.txt`, the file will have default permissions (set by the `umask` setting in `/etc/bashrc`) of

```
-rw-rw-r--  1 andrew  andrew      12 Jan  2 16:48 readme.txt
```

As you can see, you and members of your group can read and write the file. Anyone else can only read the file (and only if it is outside of your home directory, which will have read, write, and execute permission set only for you, the owner). You can remove all write permission for anyone by using `chmod`, the minus sign, and `aw` like so:

```
$ chmod -aw readme.txt
$ ls -l readme.txt
-r--r--r--  1 andrew  andrew      12 Jan  2 16:48 readme.txt
```

Now, no one can write to the file (except you, if the file is in your home or `/tmp` directory because of directory permissions). To restore read and write permission for only you as the owner, use the plus sign and the `u` and `rw` options like so:

```
$ chmod u+rw readme.txt
$ ls -l readme.txt
-rw-----  1 andrew  andrew      12 Jan  2 16:48 readme.txt
```

You can also use the octal form of the `chmod` command, for example, to modify a file's permissions so that only you, the owner, can read and write a file. Use the `chmod` command and a file permission of `600`, like this:

```
$ chmod 600 readme.txt
```

If you take away execution permission for a directory, files might be hidden inside and may not be listed or accessed by anyone else (except the root operator, of course, who has access to any file on your system). By using various combinations of permission settings, you can quickly and easily set up a more secure environment, even as a normal user in your home directory.

## Understanding Set User ID and Set Group ID Permissions

Another type of permission is “set user ID,” known as *suid*, and “set group ID” (*sgid*) permissions. These settings, when used in a program, enable any user running that program to have program owner or group owner permissions for that program. These settings enable the program to be run effectively by anyone, without requiring that each user’s permissions be altered to include specific permissions for that program.

One commonly used program with *suid* permissions is the `passwd` command:

```
$ ls -l /usr/bin/passwd
-r-s--x--x  1 root  root      13536 Jan 12  2000 /usr/bin/passwd
```

This setting allows normal users to execute the command (as root) to make changes to a root-only accessible file, `/etc/passwd`.

You also can assign similar permission using the `chfn` command. This command allows users to update or change `finger` information in `/etc/passwd`. You accomplish this permission modification by using a leading 4 (or the mnemonic `s`) in front of the three octal values.

### NOTE

Other files that might have *suid* or *guid* permissions include `at`, `rcp`, `rlogin`, `rsh`, `chage`, `chsh`, `ssh`, `crontab`, `sudo`, `sendmail`, `ping`, `mount`, and several UNIX-to-UNIX Copy (*UUCP*) utilities. Many programs (such as games) might also have this type of permission in order to access a sound device.

Files or programs that have *suid* or *guid* permissions can sometimes present security holes because they bypass normal permissions. This problem is especially compounded if the permission extends to an executable binary (a command) with an inherent security flaw because it could lead to any system user or intruder gaining root access. In past exploits, this typically happened when a user fed a vulnerable command with unexpected input (such as a long pathname or option); the command would bomb out, and the user would be presented a root prompt. Although Linux developers are constantly on the lookout for poor programming practices, new exploits are found all the time, and can crop up unexpectedly, especially in newer software packages that haven’t had the benefit of peer developer review.

Savvy Linux system administrators keep the number of *suid* or *guid* files present on a system to a minimum. The `find` command can be used to display all such files on your system:

```
# find / -type f -perm +6000 -exec ls -l {} \;
```

**NOTE**

The `find` command is quite helpful and can be used for many purposes, such as before or during backup operations. See the section “Using Backup Software” in Chapter 17, “Backing Up, Restoring, and Recovery.”

Note that the programs do not necessarily have to be removed from your system. If your users really do not need to use the program, you can remove execute permission of the program for anyone. You have to decide, as the root operator, whether your users are allowed to, for example, mount and unmount CD-ROMs or other media on your system. Although Linux-based operating systems can be set up to accommodate ease of use and convenience, allowing programs such as `mount` to be `suid` might not be the best security policy. Other candidates for `suid` permission change could include the `chsh`, `at`, or `chage` commands.

## Working As Root

The root, or super-user account, is a special account and user on UNIX and Linux systems. Super-user permissions are required in part because of the restrictive file permissions assigned to important system configuration files. You must have root permission to edit these files or to access or modify certain devices (such as hard drives). When logged in as root, you have total control over your system, which can be dangerous.

When you work in root, you have the ability to destroy a running system with a simple invocation of the `rm` command like this:

```
# rm -fr /
```

This command line not only deletes files and directories, but also could wipe out file systems on other partitions and even remote computers. This alone is reason enough to take precautions when using root access.

The only time you should run Linux as the super-user is when booting to runlevel 1, or system maintenance mode, to configure the file system, for example, or to repair or maintain the system. Logging in and using Linux as the root operator isn’t a good idea because it defeats the entire concept of file permissions.

Knowing how to run commands as root without logging in as root can help avoid serious missteps when configuring your system. Linux comes with a command named `su` that allows you to run one or more commands as root and then quickly return you to normal user status. For example, if you would like to edit your system’s file system table (a simple text file that describes local or remote storage devices, their type, and location), you can use the `su` command like this:

```
$ su -c "nano -w /etc/fstab"
Password:
```

After you press Enter, you are prompted for a password that gives you access to root. This extra step can also help you “think before you leap” into the command. Enter the root password, and you are then editing `/etc/fstab` using the nano editor with line wrapping disabled.

#### CAUTION

Before editing any important system or software service configuration file, make a backup copy. Then make sure to launch your text editor with line wrapping disabled. If you edit a configuration file without disabling line wrapping, you could insert spurious carriage returns and line feeds into its contents, causing the configured service to fail when restarting. By convention, nearly all configuration files are formatted for 80-character text width, but this is not always the case. By default, the `vi` and `emacs` editors don’t use line wrap.

You can use `sudo` to assign specific users permission to perform specific tasks (similar to BSD UNIX and its “wheel” group of users). The `sudo` command works by first examining the file named `sudors` under the `/etc` directory; you modify this file with the `visudo` command. See the section “Granting Root Privileges on Occasion—The `sudo` Command” in Chapter 14, “Managing Users,” for details on how to configure and use `sudo`.

## Creating Users

When a Linux system administrator creates a user, an entry in `/etc/passwd` for the user is created. The system also creates a directory, labeled with the user’s username, in the `/home` directory. For example, if you create a user named `bernice`, the user’s home directory is `/home/bernice`.

#### NOTE

In this chapter, you learn how to manage users from the command line. See Chapter 14 for more information on user administration with Fedora using graphical administration utilities, such as the `system-config-users` client.

Use the `useradd` command, along with a user’s name to quickly create a user:

```
# useradd andrew
```

After creating the user, you must also create the user’s initial password with the `passwd` command:

```
# passwd andrew
Changing password for user andrew.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Enter the new password twice. If you do not create an initial password for a new user, the user will not be able to log in.

You can view `useradd`'s default new user settings by using the command and its `-D` option like this:

```
# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

These options display the default group ID, home directory, account and password policy (active forever with no password expiration), the default shell, and the directory containing defaults for the shell.

The `useradd` command has many different command-line options. The command can be used to set policies and dates for the new user's password, assign a login shell, assign group membership, and other aspects of a user's account.

## Deleting Users

Use the `userdel` command to delete users from your system. This command removes a user's entry in the system's `/etc/passwd` file. You should also use the command's `-r` option to remove all the user's files and directories (such as the user's mail spool file under `/var/spool/mail`):

```
# userdel -r winky
```

If you do not use the `-r` option, you have to manually delete the user's directory under `/home`, along with the user's `/var/spool/mail` queue.

## Shutting Down the System

Use the `shutdown` command to shut down your system. The `shutdown` command has a number of different command-line options (such as shutting down at a predetermined time), but the fastest way to cleanly shut down Linux is to use the `-h` or `halt` option, followed by the word `now` or the numeral zero (`0`), like this:

```
# shutdown -h now
```

or

```
# shutdown -h 0
```

To incorporate a timed shutdown and a pertinent message to all active users, use shutdown's time and message options, like so:

```
# shutdown -h 18:30 "System is going down for maintenance this evening"
```

This example shuts down your system and provides a warning to all active users 15 minutes before the shutdown (or reboot). Shutting down a running server can be considered drastic, especially if there are active users or exchanges of important data occurring (such as a backup in progress). One good approach is to warn users ahead of time. This can be done by editing the system Message of the Day (*MOTD*) `motd` file, which displays a message to users after login. To create your custom MOTD, use a text editor and change the contents of `/etc/motd`. You can also make downtimes part of a regular schedule, perhaps to coincide with security audits, software updates, or hardware maintenance.

You should shut down Fedora only for a few very specific reasons:

- You are not using the computer and want to conserve electrical power.
- You need to perform system maintenance that requires any or all system services to be stopped.
- You want to replace integral hardware.

#### TIP

Do not shut down your computer if you suspect that one or more intruders has infiltrated your system; instead, disconnect the machine from any or all networks and make a backup copy of your hard drives. You might want to also keep the machine running to examine the contents of memory and to examine system logs. See Chapter 18 and the section “Securing Your Network” on how to protect and monitor a network-connected system.

## Rebooting the System

You should also use the shutdown command to reboot your system. The fastest way to cleanly reboot Linux is to use the `-r` option, and the word `now` or the numeral zero (`0`):

```
# shutdown -r now
```

or

```
# shutdown -r 0
```

Both rebooting and shutting down can both have dire consequences if performed at the wrong time (such as during backups or critical file transfers, which arouse the ire of your system's users). However, Linux-based operating systems are designed to properly stop active system services in an orderly fashion. Other commands you can use to shut down and reboot Linux are the `halt` and `reboot` commands, but the `shutdown` command is more flexible.

**RELATED FEDORA AND LINUX COMMANDS**

The following programs and built-in shell commands are commonly used when working at the command line. These commands are organized by category to help you understand the command's purpose. If you need to find full information for using the command, you can find that information under the command's man page.

**Managing users and groups**—chage, chfn, chsh, edquota, gpasswd, groupadd, groupdel, groupmod, groups, mkpasswd, newgrp, newusers, passwd, umask, useradd, userdel, usermod

**Managing files and file systems**—cat, cd, chattr, chmod, chown, compress, cp, dd, fdisk, find, gzip, ln, mkdir, mksfs, mount, mv, rm, rmdir, rpm, sort, swapon, swapoff, tar, touch, umount, uncompress, uniq, unzip, zip

**Managing running programs**—bg, fg, kill, killall, nice, ps, pstree, renice, top, watch

**Getting information**—apropos, cal, cat, cmp, date, diff, df, dir, dmesg, du, env, file, free, grep, head, info, last, less, locate, ls, lsattr, man, more, pinfo, ps, pwd, stat, strings, tac, tail, top, uname, uptime, vdir, vmstat, w, wc, whatis, whereis, which, who, whoami

**Console text editors**—ed, jed, joe, mcedit, nano, red, sed, vim

**Console Internet and network commands**—bing, elm, ftp, host, hostname, ifconfig, links, lynx, mail, mutt, ncftp, netconfig, netstat, pine, ping, pump, rdate, route, scp, sftp, ssh, tcpdump, traceroute, whois, wire-test

---

## Reference

The migration to a new computer operating system does not have to be painful to management and users. Providing easy-to-understand directions, some background information, and preconfiguration of an installed system can help the transition.

This section lists some additional points of reference with background information on the standards and commands discussed in this chapter. Browse these links to learn more about some of the concepts discussed in this chapter and to expand your knowledge of your new Linux community.

<http://www.winntmag.com/Articles/Index.cfm?ArticleID=7420>—An article by a Windows NT user who, when experimenting with Linux, blithely confesses to rebooting the system after not knowing how to read a text file at the Linux console.

<http://standards.ieee.org/regauth/posix/>—IEEE's POSIX information page.

<http://www.itworld.com/Comp/2362/lw-01-government/#sidebar>—Discussion of Linux and POSIX compliance.

<http://www.pathname.com/fhs/>—Home page for the Linux FHS (Linux Filesystem Hierarchy Standard).

<http://www.tldp.org/>—Browse the HOWTO section to find and read The Linux Keyboard and Console HOWTO—Andries Brouwer's somewhat dated but eminently useful guide to using the Linux keyboard and console.

<http://www.gnu.org/software/emacs/emacs.html>—Home page for the FSF's GNU `emacs` editing environment; you can find additional documentation and links to the source code for the latest version here.

<http://www.vim.org/>—Home page for the `vim` (`vi` clone) editor included with Linux distributions. Check here for updates, bug fixes, and news about this editor.

<http://www.courtesan.com/sudo/>—Home page for the `sudo` command. Check here for the latest updates, security features, and bug fixes.

