

Red Hat Linux 6.0

The Official Red Hat Linux Getting Started Guide

Red Hat Software, Inc.
Durham, North Carolina

Copyright © 1999 Red Hat Software, Inc.

Red Hat is a registered trademark and the Red Hat Shadow Man logo, RPM, the RPM logo, and Glint are trademarks of Red Hat Software, Inc.

Linux is a registered trademark of Linus Torvalds.

Motif and UNIX are registered trademarks of The Open Group.

Alpha is a trademark of Digital Equipment Corporation.

SPARC is a registered trademark of SPARC International, Inc. Products bearing the SPARC trademarks are based on an architecture developed by Sun Microsystems, Inc.

Netscape is a registered trademark of Netscape Communications Corporation in the United States and other countries.

Windows is a registered trademark of Microsoft Corporation.

All other trademarks and copyrights referred to are the property of their respective owners.

Revision: GSG-6.0-HTML-RHS (04/99)

Red Hat Software, Inc.

2600 Meridian Parkway

Durham, NC 27713

P. O. Box 13588

Research Triangle Park, NC 27709

(919) 547-0012

redhat@redhat.com

<http://www.redhat.com>

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

The *Official Red Hat Linux Getting Started Guide* may be reproduced and distributed in whole or in part, in any medium, physical or electronic, so long as this copyright notice remains intact and unchanged on all copies. Commercial redistribution is permitted and encouraged, but you may not redistribute it, in whole or in part, under terms more restrictive than those under which you received it.

Contents

Introduction	4
1.1 A Note About Conventions	6
1.2 The Root of the Matter	7
1.3 How to Quit	10
1.4 From Console Mode	11
1.5 X Marks the Spot	11
1.6 A Brand New You	13
1.7 Shutting Down	19
1.8 Pulling Yourself Up by the Boot	21
1.9 A Good ``Man" Is Easy to Find	24
1.10 What is Rescue Mode?	26
2 You Are Here	29
2.1 Finding Yourself With <code>pwd</code>	29
2.2 Getting From Here to There: <code>cd</code>	30
2.3 Looking Around With <code>ls</code>	34
2.4 A Larger Picture of the Filesystem	39
2.5 ``Washing" the Window	41
2.6 Using <code>cat</code>	42
2.7 Using Redirection	43
2.8 Appending Standard Output	45
2.9 Redirecting Standard Input	46
2.10 Pipes	46
2.11 Stringing Commands Together	47
2.12 Ownership and Permissions	48
2.13 Fun with Numbers in <code>chmod</code>	53
3 Managing Files and Directories	56
3.1 Shell Collecting	56
3.2 Locating Files and Directories	57
3.3 Command History and Tab Completion	57
3.4 Identifying and Working with File Types	58
3.5 Copying, Moving and Renaming Files and Directories	62
4 What Do I Do Now?	65
4.2 The X Window System	65
4.3 Configuring Your Red Hat Linux System For Sound	68
4.4 World Wide Web	70
4.5 Good luck and enjoy!	71

Introduction

Welcome to Red Hat Linux 6.0!

At Red Hat Software, we believe we offer the best Linux distribution on the market. We hope you'll agree that the time and the money you spent for Red Hat Linux was well spent, indeed.

Recently, Linux has gained quite a bit of attention from the national and international media. What began as a ``hacker's hobby" several years ago has been embraced as a powerful and economical computer operating system.

If you count yourself among the many Linux users who are discovering Red Hat Linux for the first time, this book is for you!

Inside, you'll find valuable tips which will help you get acquainted with your new desktop environment and with the way your Red Hat Linux system works. You'll be able to learn some basics and you'll find pointers to places where you can turn for more information.

The Newbie's Guide to Red Hat Linux

Are you rattled by terms like *root* and *user account*? The following is for you!

The second part of the Red Hat Linux Getting Started Guide, this ``newbie's guide" will help you gain a toehold on the basics of your new Linux system -- from creating a new account to working with files in a non-graphical environment.

There's nothing wrong with a little hand-holding -- and that's what you'll find in these remaining chapters.

Here's a glimpse of what you can find:

- **Chapter 1: Welcome to Linux** -- Learn how to create your own user account to maximize your system's safety. You'll also find out how to shut down your system, create rescue disks and more.
- **Chapter 2: You Are Here** -- Learn how to navigate through your system at the shell prompt, how to combine commands and see how everything fits together.
- **Chapter 3: Managing Files and Directories** -- Here, you'll learn more about the powerful *shell* you're using, how to save yourself time and frustration when you're typing in commands and how to rename, copy, delete and move files and directories.
- **Chapter 4: What Do I Do Now?** -- Looking for pointers to more information about your Red Hat Linux system? Here, you'll find tips about where you can find plenty of documentation and help. You can also learn more about your X Window System and how to work with other system tasks.

More to Come

As Linux evolves, so does the support you'll find for Red Hat Linux. The Red Hat Linux Getting Started Guide is part of that support -- and evolution. In coming editions, expect to find more essential information to help you get the utmost from your Red Hat Linux system.

That's also where you come in.

Send in Your Feedback

If you'd like to make suggestions about the Red Hat Linux Getting Started Guide, please mention this guide's identifier:

GSG-6.0-HTML-RHS (04/99)

That way we'll know exactly which version of the guide you have. You can send mail to:

docs@redhat.com

A Thousand Thanks

This guide is the definition of a group project, since so many provided valuable assistance, from offering suggestions and sharing knowledge to proofreading.

Thank you to Edward C. Bailey, the documentation department's manager. Ed was there from concept to ``when the rubber hit the road," offering his expert advice on style and substance.

Thank you also to Sandra A. Moore, in charge of the Official Red Hat Linux Installation Guide, for her patience and help in formatting and proofing. And to David Mason, RHAD Labs' technical writer, who worked like to a demon to put together the GNOME User's Guide.

Red Hat Software's support team -- particularly Stephen Smoogen and Eric Rahn Nolen ("Thor") were more than generous in offering their time and advice.

And to the engineers, who build the best Linux distribution, a big "thank you"! It is their work which makes Red Hat Linux so worthwhile.

And, of course, thank you to Linus Torvalds and the thousands of Linux developers around the world. Ultimately, this is *their* operating system -- and it is a wonder.

Paul Gallagher

Congratulations!

As a new Red Hat Linux user, you've successfully installed one of today's most advanced computer operating systems.

What began in 1991 as a hobby for a young Finnish student named Linus Torvalds has ballooned from a "hacker's darling" into an important tool for both home and business users.

Just six years ago, there were an estimated 100,000 users. Today, about 12 million users worldwide depend on Linux to manage finances, use and control Internet services, create artwork and more. That number is rapidly growing; every day, new users are discovering the power and potential of Linux. This free, UNIX-like operating system is a multitasking, multi-user environment that has superior memory management, great security features, and more.

In other words, power and, once you become more comfortable with Linux, ease of use.

Tip: Linux is most frequently pronounced with the short "i" and the accent on the first syllable, as in "LIH-nucks".

What do you do next? Relax.

In the chapters that follow, we hope to show you the basics of how to get the most out of your new system. If you're interested, we will also show you the roads to take which can lead you to becoming a Linux guru.

1.1 A Note About Conventions

At the time you installed your Red Hat Linux system, you were given the option of working entirely in a graphical environment, such as GNOME, or logging in from console mode, which is non-graphical. If you're like many new Linux users, you're familiar with graphical environments such as Microsoft Windows, Apple Macintosh or IBM's OS/2.

So it's a fairly safe bet that you chose to work in a graphical environment when you installed Red Hat Linux 6.0.

You'll find plenty of opportunities to "point and click" on applications -- either on your desktop or from the menu at the bottom of your desktop. But we're going to spend much of our time working from the "shell prompt."

Why? Because at the same time you accomplish tasks, you can learn a little more about how your Red Hat Linux system works.

Tip: Unlike a graphical presentation, a "shell prompt" is the way you can type commands directly to the "shell." You need a shell to use Linux, because it's the tool you use to interact with your operating system. You'll find more information about your shell in Chapter 3.



Figure 1: The GNOME footprint on the panel

There are plenty of ways to get a shell prompt, depending on the kind of graphical environment you're using, such as GNOME. Depending on the environment chosen, just by right- or left-clicking in a blank space on your desktop, you'll see a reference to `xterm`.

By "dragging" your cursor over that item you will open a shell prompt window. Other times, you'll find you can get a shell prompt window through the menu on your desktop.

In addition to `xterm`, other references which will give you the shell prompt include:

- terminal emulator window
- GNOME terminal
- Color Xterm

We'll use GNOME as our example.

To begin, take your cursor to the GNOME footprint on the panel at the bottom of the desktop.

Now, left-click once on the footprint (see Figure 1), and a menu of "folders" will pop up. These folders represent categories of various software groups on our system. There are utilities, graphics programs, Internet applications and much more.

Once the menu pops up, "drag" the cursor to the **Utilities** section of the menu by holding down the mouse button while raising the mouse to the Utilities folder.

Once the cursor is over the folder, a new menu pops up to the right of the Utilities folder.

Here, in the first entries of this new submenu, there is a choice of terminal windows: `Regular xterm`, `Color xterm` and `GNOME terminal` (as shown in Figure 2). To get a shell prompt, position the cursor over the terminal window of your choice, and release the mouse button.

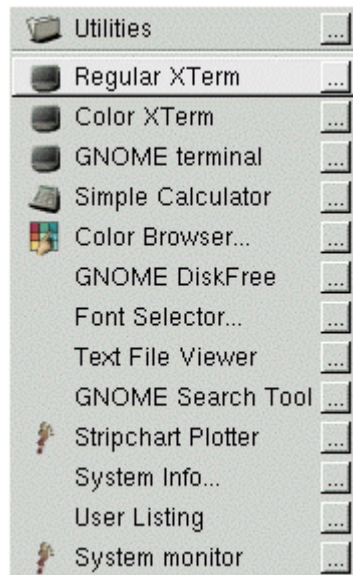


Figure 2: Shell prompt menus in Utilities

Tip: Of the choices in the **Panel -> Utilities** menu, the **GNOME terminal** offers the ability to set background color and other preferences most easily.

Now, it's time to take your first steps.

1.2 The Root of the Matter

When your Red Hat Linux system starts, you'll see an array of messages speeding past you on the screen. Many of these messages simply tell you what services are starting on your computer.

Tip: Want to read those startup messages more closely? At a shell prompt, type `dmesg | more`. You'll be able to read the file one screen at a time. To move forward, press the [Spacebar]; to quit, press [Q].

Finally, we'll come to "the login prompt" (as shown in Figure 3).

You'll find:

Login:

Password:

At this point, some new users can easily feel rattled, but don't panic. Instead, think back: When you installed Red Hat Linux 6.0, you were asked for a *root password*.



Figure 3: A sample screen of the graphical login prompt

In detail, that is the password you were asked to choose to log in to your root account. When you log in -- either in the root account or other accounts -- you're introducing yourself to the system. The root account, unlike all other accounts for your system, has access to everything. Also known as the *superuser*, the root account can control everything the system does.

Go ahead and login; at the `Login:` prompt just type:

```
root
```

and press [Enter] or the [Tab] key.

Tip: Case matters. Linux, like UNIX, makes a distinction between uppercase and lowercase letters. So `root` is not the same as `Root`. In fact, as far as Linux is concerned, they're two different accounts.

Don't worry about mistakes when you log in; you can always use the [Delete] key to start over. When you're asked, type in the password you chose when you installed Red Hat Linux. You won't see your password on the screen as you type; that's just one of the security features of your Linux system.

Tip: Be sure to type commands exactly as you see them -- spaces, dashes and all. To Linux, an extra space or letter can make all the difference in the world.

When you're finished typing in your password, press [Enter].

You'll be presented with your new desktop (similar to [4](#)). Once you become more comfortable with your new operating system and with GNOME, you'll probably be able to fill up that desktop quickly with applications.

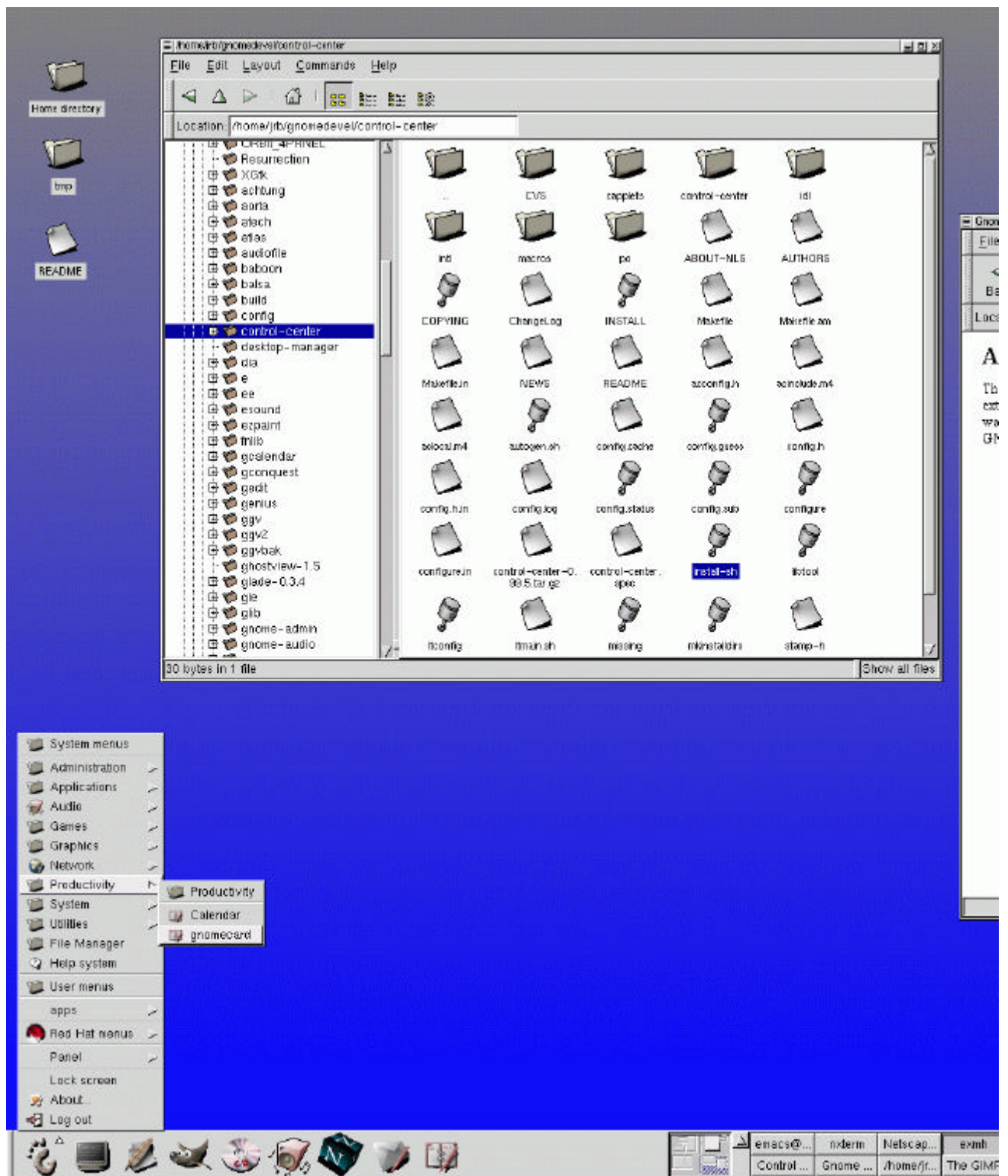


Figure 4: A sample GNOME desktop

There's plenty of space to fill up, not just on your current desktop but on numerous desktops -- four by default.

Tip: To learn specifics about GNOME, including applications and navigation, turn to the GNOME User's Guide section of this manual.

But for now, let's just concentrate on your current desktop.

Go ahead and look around. You can begin by double-clicking with the left mouse button on the file folder called Home Directory on the desktop.

Here, you'll find icons representing the various directories and files on your system.

From the panel on the bottom, left-click once with your mouse, and you can begin to investigate some of the applications which have been included with your environment.

From here, you can find ways to customize your workspace, search for files, write letters or other documents, start spreadsheets and more.

But before you get too daring...

While you're logged into the root account, avoid the temptation to make any changes to files or directories unless you know exactly what you're doing!

Here's why: Whenever the system recognizes you as the root account you're allowed to do just about anything: change configuration files, make new directories, create and manage accounts for users who are allowed to use your computer and more.

That kind of power comes with a price, and tinkering around with configuration files, accounts and directories can easily lead to disaster.

So how are you supposed to operate safely?

By creating a user account, which we'll cover shortly. With a user account, you can work and play with the assurance that you're not damaging your system.

1.3 How to Quit

When you're finished looking around for the first time, you can log out to quit your session (see [Figure 5](#)).

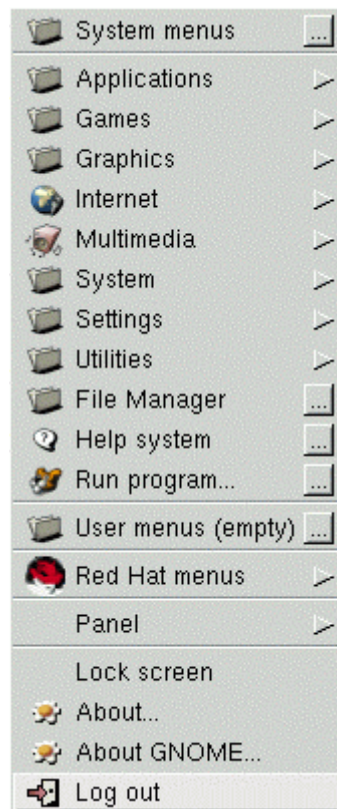


Figure 5: Locating the "Log out" selection

When you log out, you'll be returned to the opening screen you found when your system started.

To log out, just go to the **GNOME footprint** on the panel, left-click once and click on **Log out**.

You'll be presented with a box, asking you whether you want to log out. Click once on **Yes**.

After a few moments, you'll be returned to the log in screen.

Summary: At Login - type `root` At Password - type `yourrootpassword` To quit -- Left-click on the GNOME footprint, then click on **Log out**, click **Yes**.

1.4 From Console Mode

When you were installing Red Hat Linux 6.0, you were given the option of starting from a graphical or console -- non-graphical -- screen.

If you chose not to automatically start your computer in a graphical environment, you'll find a somewhat daunting, almost blank screen which will show you something like

```
Red Hat Linux release 6.0
Kernel 2.2 on an i686
login:
```

You can log in by typing `root` at the **Login:** prompt. Then, when **Password:** appears, type the password you chose at the time you installed Red Hat Linux 6.0.

Tip: Just like the graphical login screen, don't expect to see your password ``echoed" when you type in your password. Making sure that your password isn't seen is just one of Linux's many security features.

Now, you'll find a single shell prompt, which will appear similar to:

```
[root@localhost root]#
```

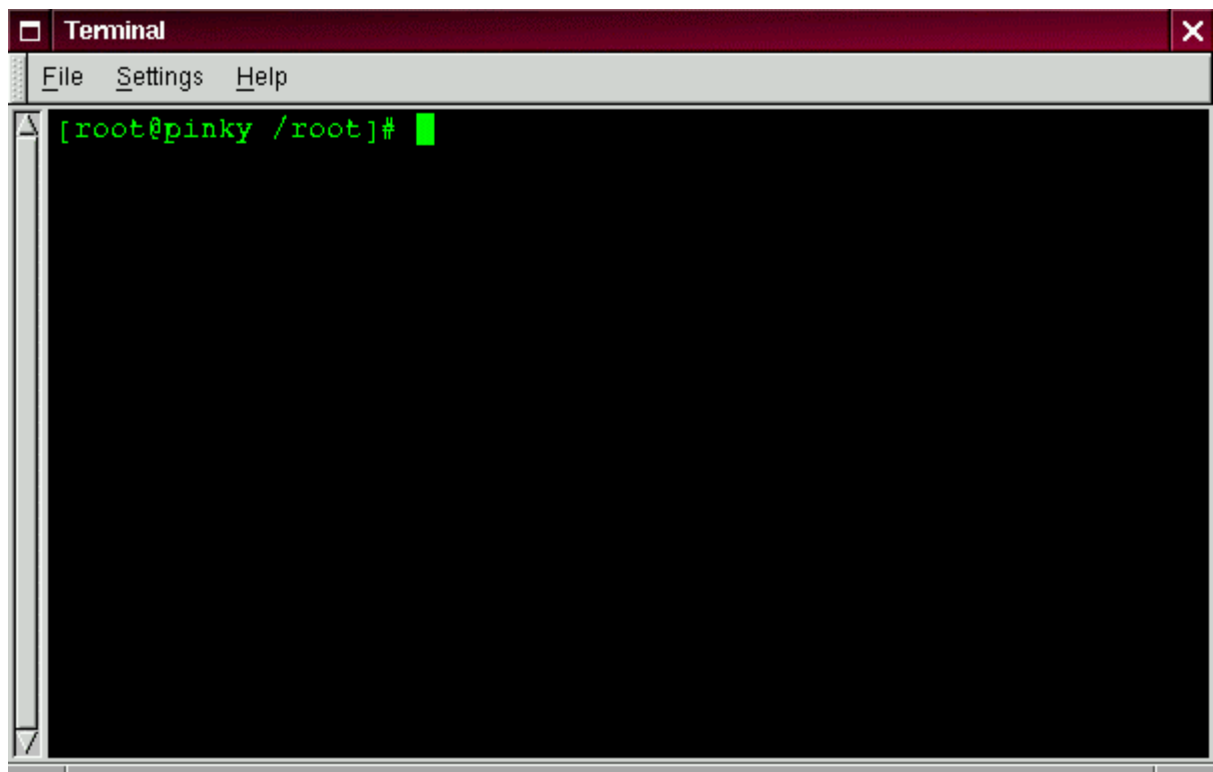


Figure 6: A sample shell prompt after your login

This tells you that you're logged in as root and in the directory called root (as shown in [Figure 6](#)). You can exit at any time simply by typing `logoff` or `exit`.

Tip: You can also press the [Ctrl] and [D] keys at the same time to return you to the login prompt.

1.5 X Marks the Spot

If you installed the X Window System (also known simply as X) at the time of your Red Hat Linux installation, you've got a pleasing, graphical environment in which to work.

If you didn't install X at that time, and you wish to use the X Window System, your best bet is to return to the CD at this time and re-install Red Hat Linux 6.0. (Sigh...)
Certainly, there *are* other ways of installing X, but if you're fairly new to Red Hat Linux, and if you're starting out with a brand new installation, you'll find it takes less time -- and frustration -- to simply redo the installation.

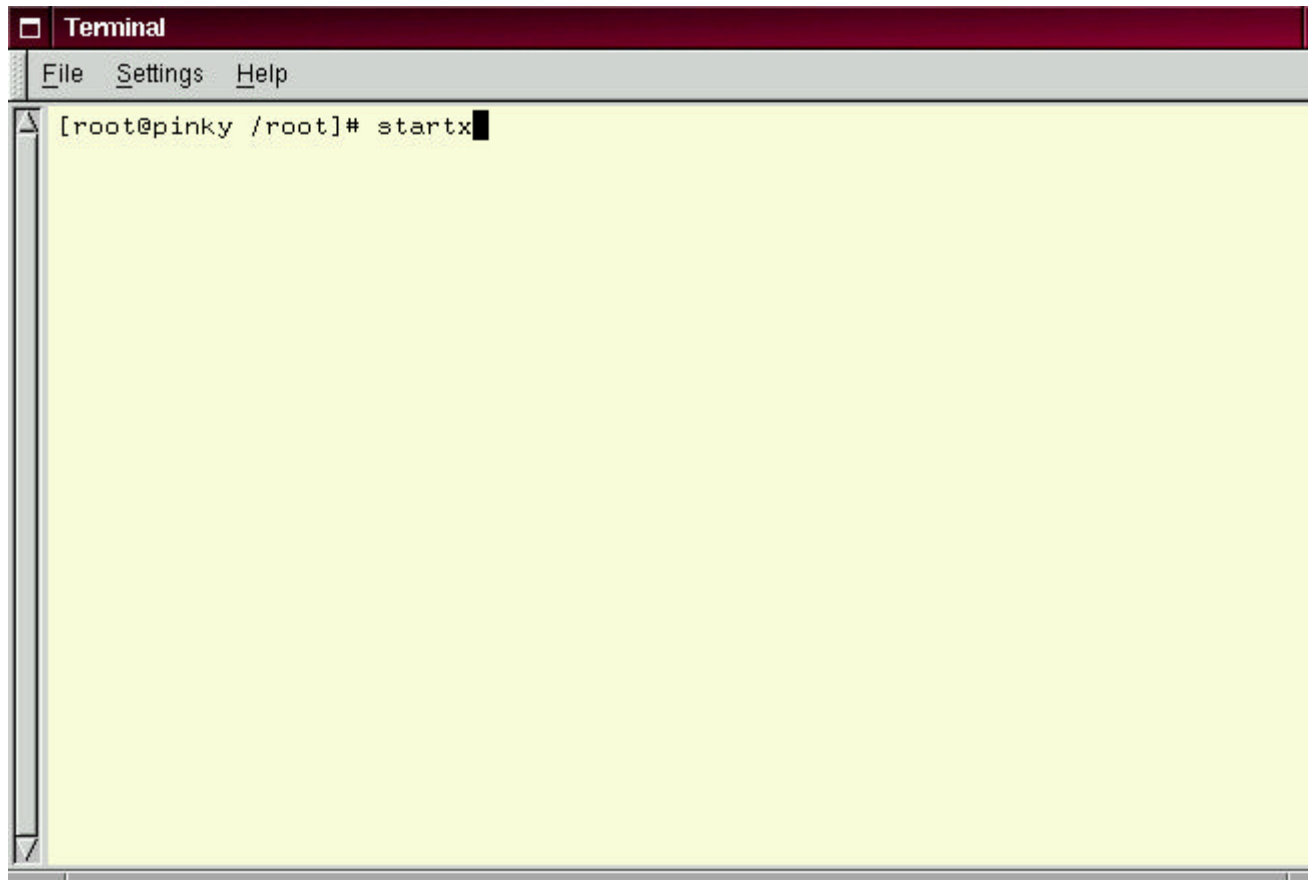


Figure 7: Starting the X Window System from the prompt

If you did install the X Window System -- but didn't choose to start GNOME automatically you're still ready to go.

At the shell prompt, type:

```
startx
```

and the X Window System will begin (refer to Figure 7).

Please Note: You're logged into the system as the root account, also known as superuser. There's a reason the root account is known as superuser: In this account, you can make changes to just about anything. Unless you know what you're doing, you can easily harm your system by mistakenly changing settings.

Although you may be tempted to modify files or directories, you should resist making any changes until you've created a user account.

To log out of X, bring your mouse cursor to the **GNOME panel**, then left-click on the **GNOME footprint**. A menu of applications, utilities, games and other programs will pop up.

``Drag" your cursor to the item labeled **Log out** (as shown in Figure 5).

Tip: You can ``drag" your cursor by keeping the mouse button depressed with your finger while moving the cursor to your selected item. Once the cursor is over the item on the menu, releasing the mouse button will start the program.

Now, a separate window will appear, asking you to confirm your decision to log out.

Click on **Yes**, and you'll be returned to the console.

You now will be back at your original shell prompt, so if you are done for the day, you should log out here too.

Whenever you want to start another X session, just type `startx` from the prompt.

Summary: At the prompt, type `startx`; to exit -- **GNOME panel -> Log out**

1.6 A Brand New You

Now, let's create a "user account."

If you're familiar with MS-DOS or, to a lesser extent, Windows 98, you might be a little befuddled by the requirement of creating a user account.

After all, if you can navigate the system and use programs in your root account, you might think that having two accounts on a single machine is excessive.

Nothing could be further from the truth. Here's why:

Linux is a multi-tasking, multi-user system, which means it can safely and securely accommodate many users at one time, performing plenty of tasks each user requires. But only one account can be root -- capable of changing the way the operating system works.

Because "rooting around" can easily lead to havoc, it's important to safeguard against accidents. That's why just about every Linux user -- even if they're the system administrator -- has their own user account.

Once you're logged in as root, you have two ways to conveniently add a user to the system: from the within X and from the shell prompt.

Both methods are quick and painless.

Let's say that the account you want to choose is called "billy."

From X:

One of the most powerful tools you can use for system administration is *Linuxconf*. You can use Linuxconf for adding and manipulating accounts, monitoring system activities and plenty of other system features.

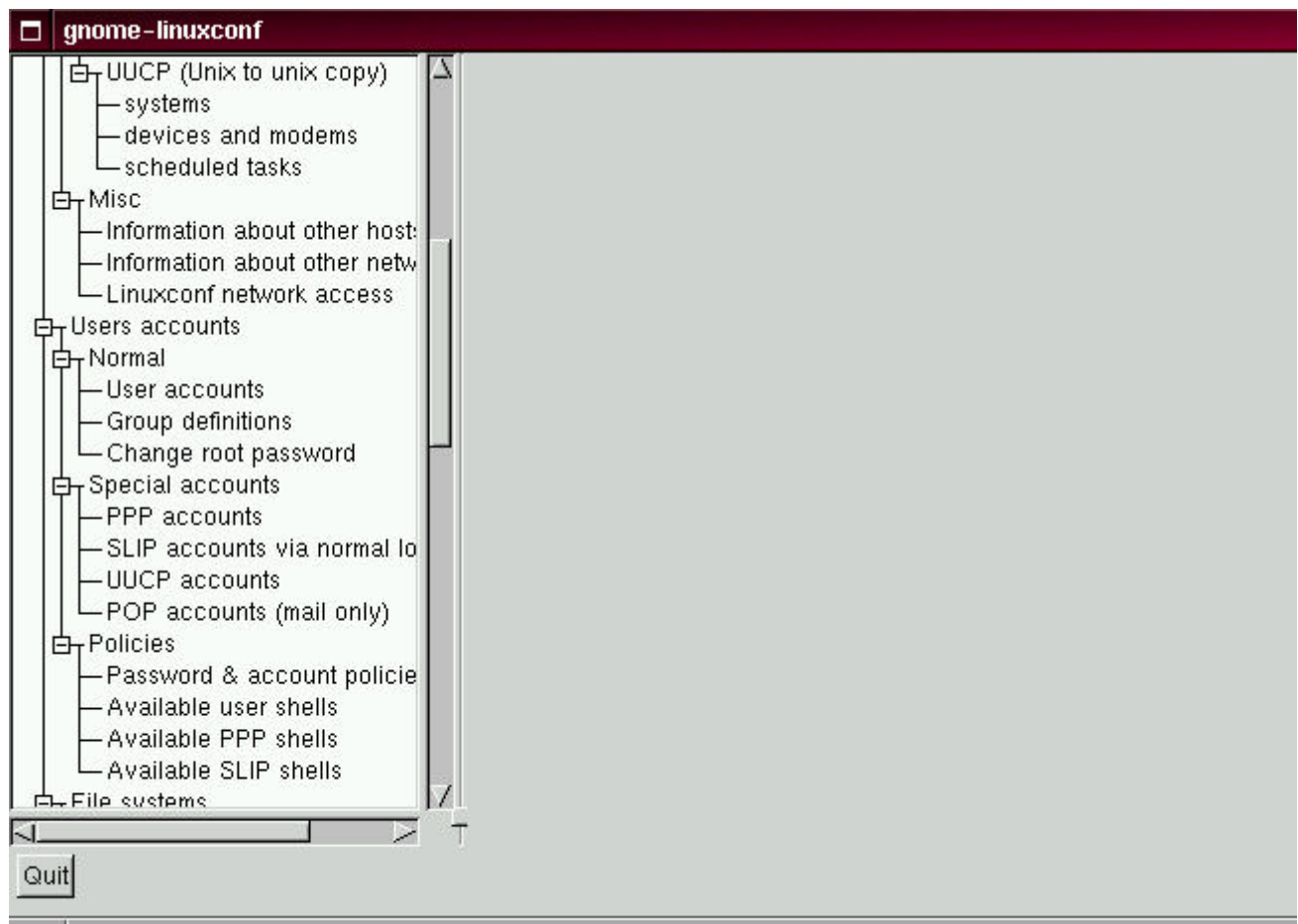


Figure 8: Finding the Users accounts entry in Linuxconf

Completely documenting all the features of this utility would take much more space than we have here. For a more detailed look at the application's features -- including greater depth on manipulating

accounts -- turn to the "System Configuration with Linuxconf" chapter in the Red Hat Linux Installation Guide.

Tip: You can learn more about Linuxconf by visiting the official Linuxconf website:

<http://www.solucorp.gc.ca/linuxconf/>.

One of the easiest ways to access Linuxconf is from the shell prompt. At the prompt, type:

```
linuxconf
```

We want to add an account, so let's scroll about a third of the way down the menu in the left panel, to the entry marked **Users accounts**. If the entry has a + next to it, go ahead and click on the "+".

Now, the menu will expand to show entries in the **User accounts** listing (as shown in Figure 8).

The subentries will look like the following:

```
|
|-Users accounts
|   |
|   | Normal
|   |   |
|   |   | User accounts
|   |   |
|   |   | Group definitions
|   |   |
|   |   | Change root password
```

Left-click with your mouse button on the **User accounts** entry, under **Normal**.

In the right panel, you'll now see a box of the current user accounts (as shown in Figure 9).

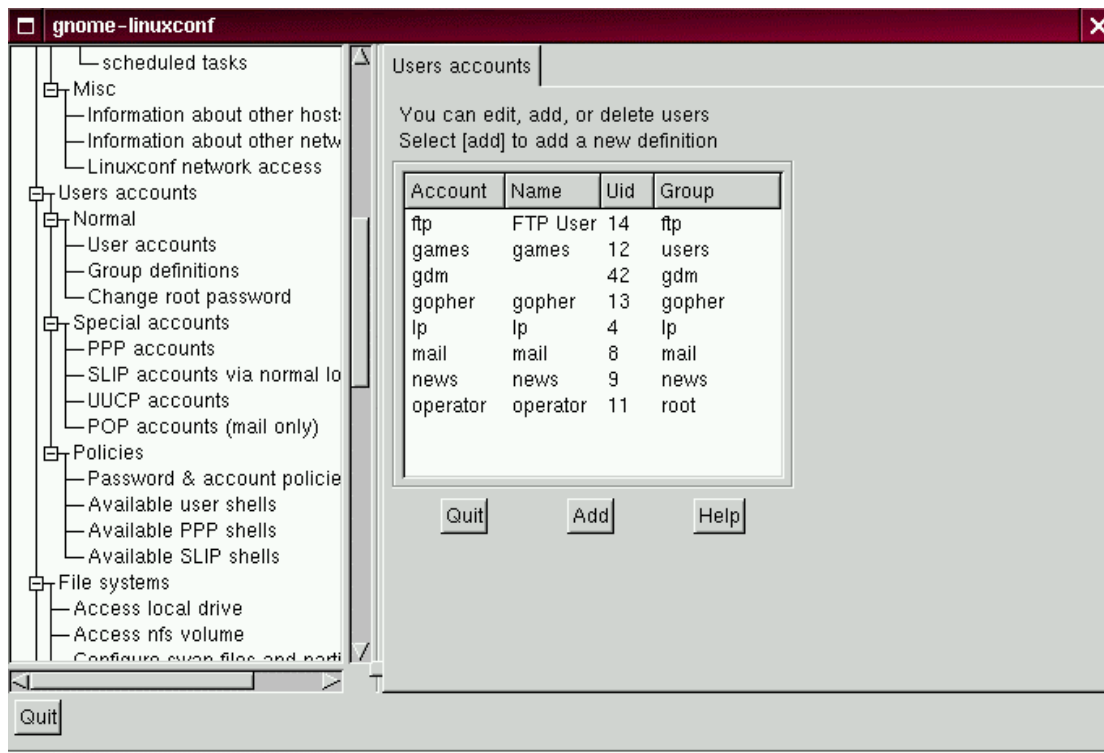


Figure 9: The Users accounts in Linuxconf

Toward the bottom of the right panel, click on the **Add** button, between the **Quit** and **Help** buttons.

Now, we'll see a dialog called **User account creation**. In here, we're going to fill in: **Login name**; **Full name** and **group**.

Make sure the button is indented next to the statement **The account is enabled**.

Now, let's type in a login name. It should be easy to remember (it's the password that should be complex, but more about that later...). Then, you can type in your full name.

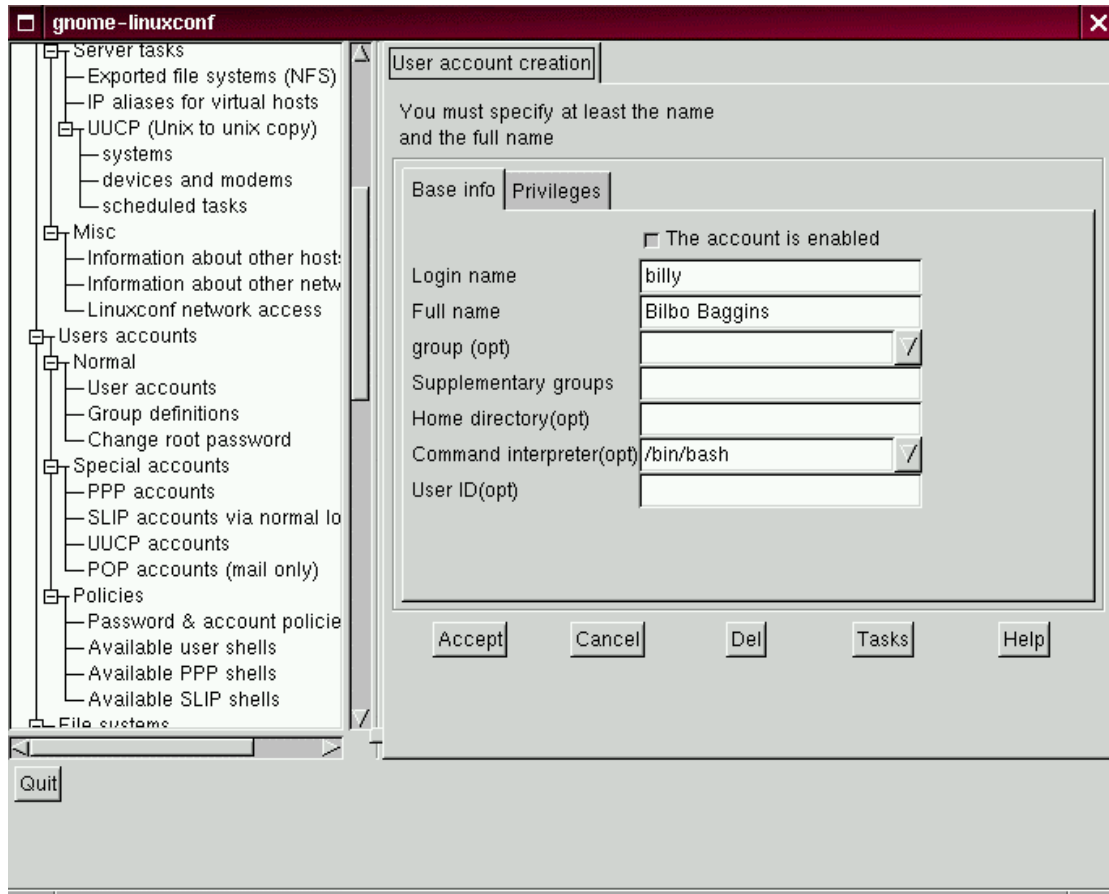


Figure 10: Adding a user in Linuxconf

Tip: Everyone's account belongs to at least one group. Groups are used to determine file access permissions. Unless you specify a group, the default group for your user account will be the login name you choose (for example, a group called **billy**).

When you're finished, your entries should look like the following:

Login name.....billy

Full name.....Bilbo Baggins

group (opt).....billy

Now, just click on the button marked **Accept**.

We're almost finished. Next, we've got to come up with a password.

Passwords are one of the best methods to safeguard against prying eyes or malicious behavior. If you've got a secure password, which only you know, you've taken a big step in your system's security.

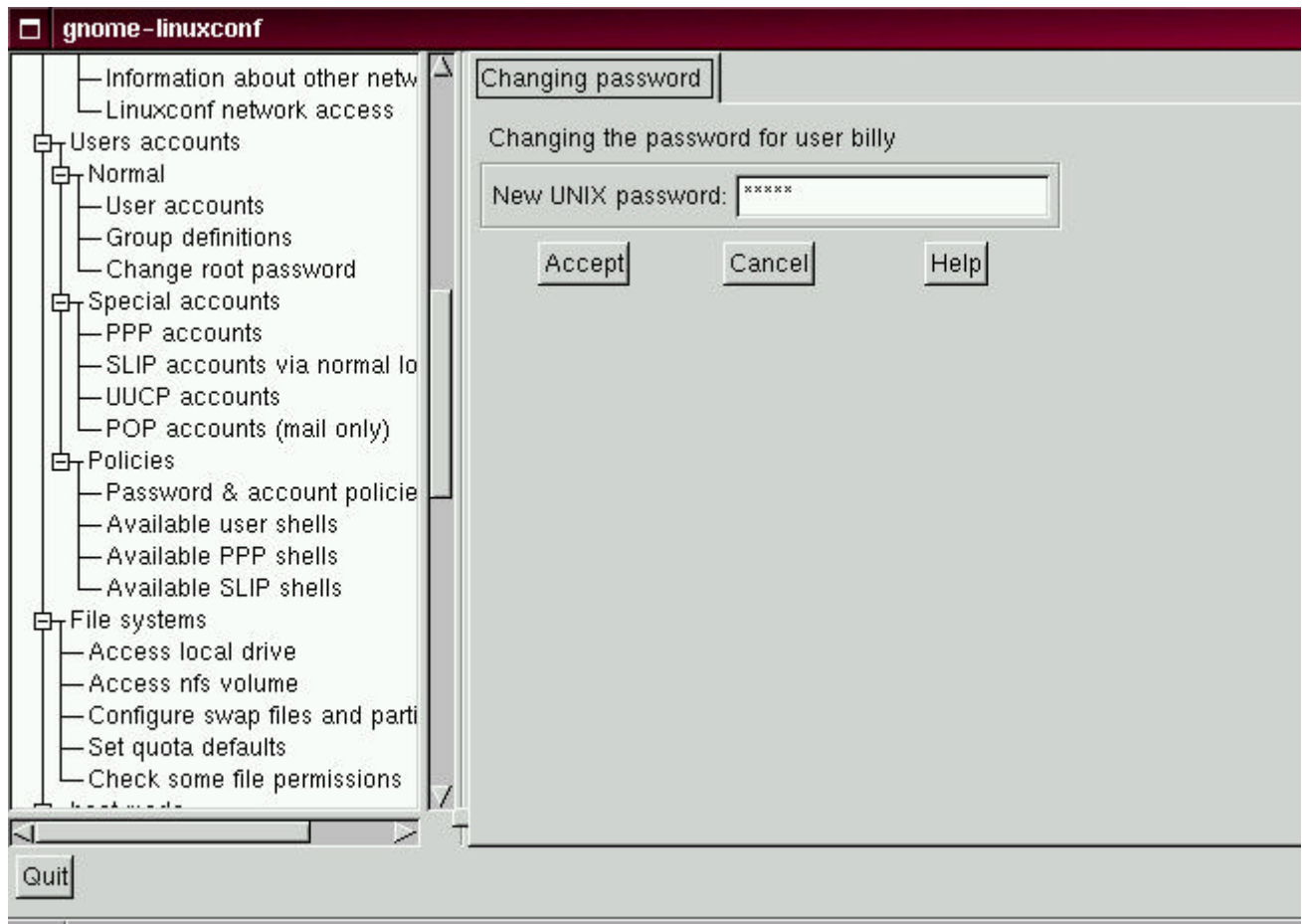


Figure 11: Creating a user account's password in Linuxconf

For both your root and user account, your passwords should be unique and easy enough for you to remember. (Passwords can't protect very well if they're jotted down on a piece of paper and taped to the monitor!)

What's both unique and easy to remember? Passwords which both numbers and letters. Here's an example:

- **Weak passwords:** airplane, icecream, california
- **Better passwords:** a!rpl8ne, !cec73am, c8Li70r&ia

One more thing: Passwords must be at least six characters -- upper and lowercase letters and/or numbers -- in length.

Once you decide on a password that you feel comfortable you'll remember, type it in the box provided. You won't see the password, except in a series of asterisks (as shown in Figure 11).

Then, click on the **Accept** button.

You'll be asked to retype the password for verification. Again, you won't see your password as you type it.

When you've finished, you'll see the account listed in the accounts panel (see Figure 12 for an example).

That's all there is to it.

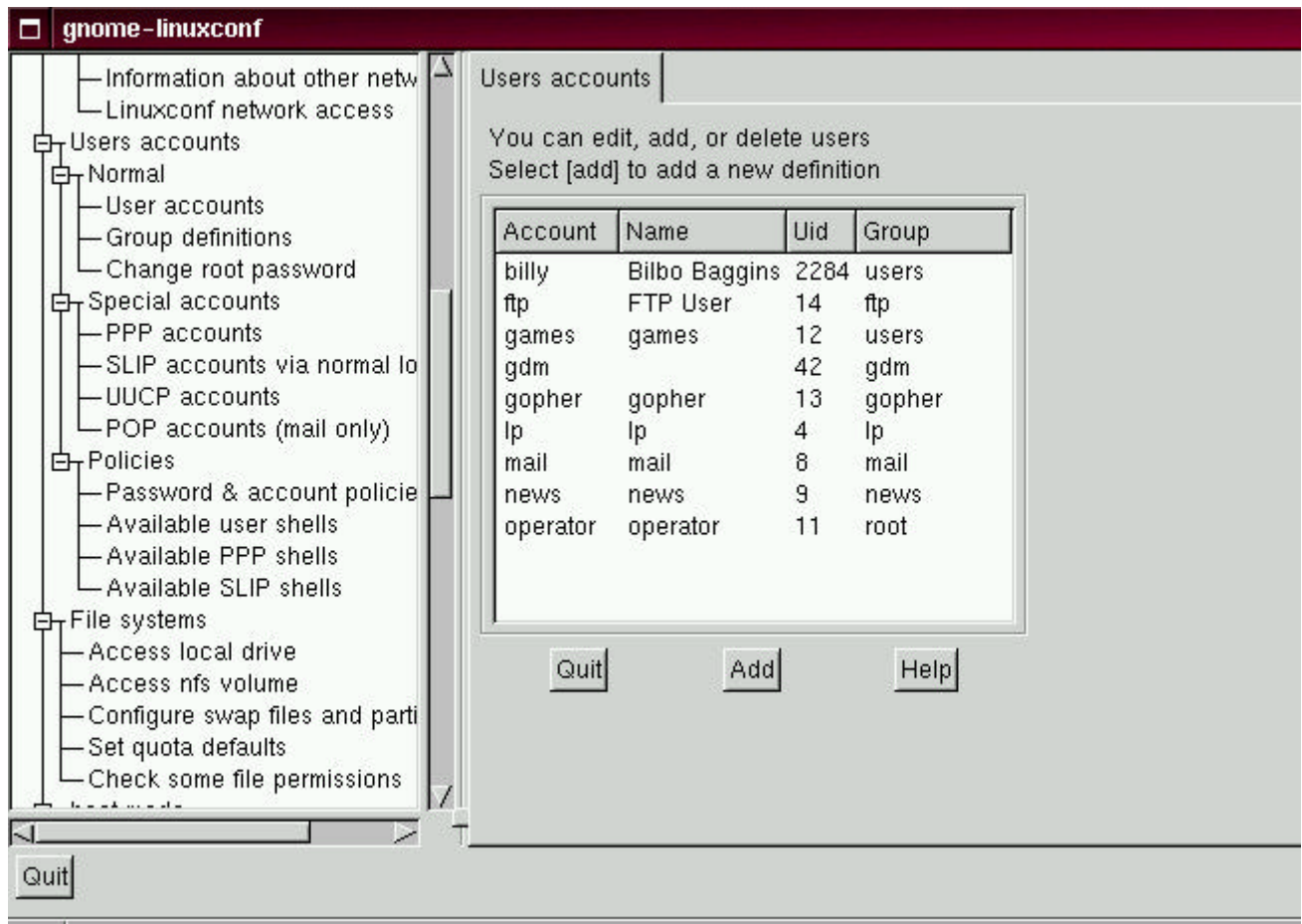


Figure 12: The new user account in Linuxconf

For additional security, you should change your passwords every now and then. From your user account, you can change your password by clicking on the account name, then clicking on the box marked **Password** at the bottom.

Tip: Now that you've created your user account, you might want to reconsider whether your root account's password is secure enough. You can change this password easily, from within Linuxconf, by clicking on the **Change root password** item.

To learn more about how to modify your account or perform other account procedures, turn to the System Configuration chapter in the Red Hat Linux Installation Guide.

From the shell prompt:

At the prompt, type `useradd billy`. It should look like this:

```
[root@localhost root]# useradd billy
[root@localhost root]#
```

(See Figure 13.)

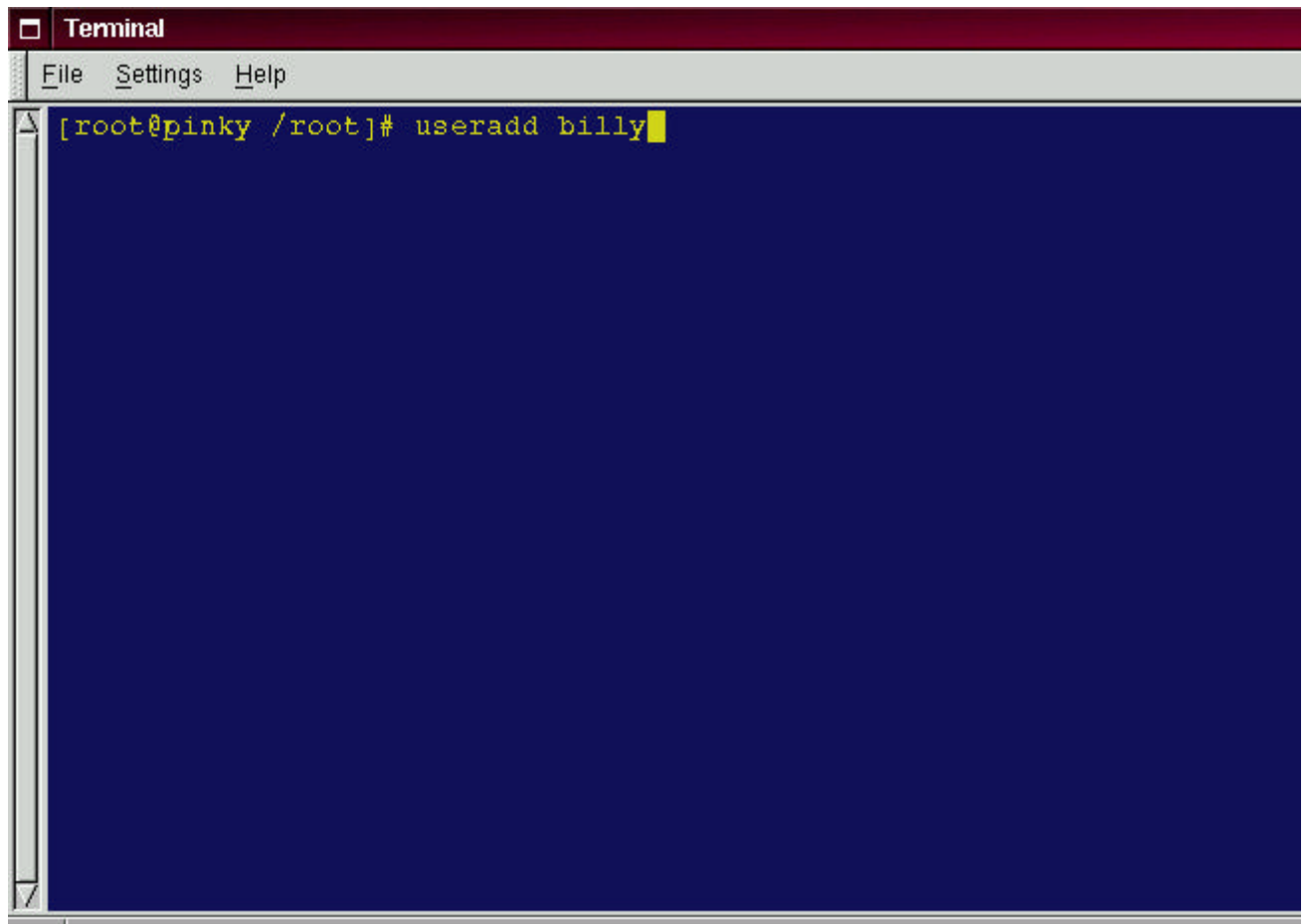


Figure 13: Adding a user at the shell prompt

Looks like nothing's changed, right? Wrong. Although you've got the same prompt, an entry has already been made for the new account. Now, it's time to specify a password. At the prompt, type `passwd billy`. It should look like this:

```
[root@localhost root]# passwd billy  
[root@localhost root]#
```

(See [Figure 14](#).)

Remember that the password must be easy to recall and a unique mixture of letters, symbols and/or numbers.

That's it.

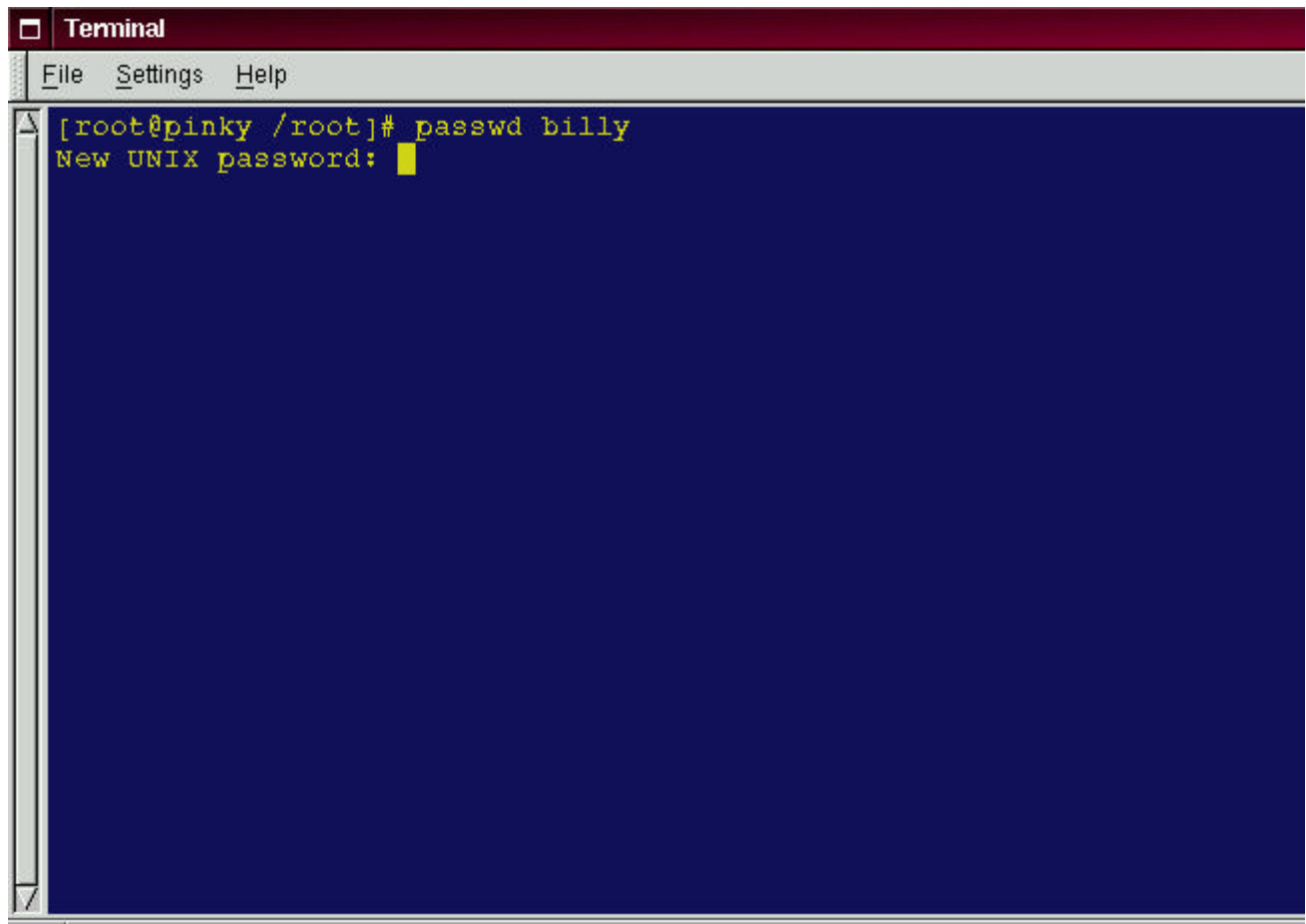


Figure 14: Adding a password at the shell prompt

From now on, whenever you want to add a user, change account information or change account passwords -- including the root password -- either `useradd` or `linuxconf` will fit the bill. Make sure you're logged in as the root user, however.

Tip: If you find yourself switching around frequently between the root account and your user account, it can become confusing to know which account you're actually logged into. You can always tell you're in the root account you see the word `[root]` at the start of the shell prompt or the hash mark (`#`) at the end. If you see (for example) `[billy]`, or the dollar sign (`$`), you're working in your user account.

Regardless of the method you choose -- from the shell prompt or from `Linuxconf` -- your new user account's "login directory" will be placed in a subdirectory of `/home`.

To finish up and try out your new account, log out from your root account.

You'll be taken back to the login screen.

Now, you can log in to your new user account.

Summary: From X -- In a terminal emulation window, type `linuxconf`. Scroll to **User Accounts** -> **Normal** -> **User Accounts**. Click on **Add**. **From the shell prompt** -- type `useradd (accountname)`; at the shell prompt again, type `passwd accountname`. Type and re-type password.

1.7 Shutting Down

Some day, computers will probably be as easy to use as televisions are today (no, we're not there yet...). Maybe we'll have remote controls to navigate easily between features and to turn off the machine.

At present, though, you can't simply turn off your computer when you're finished. You can always log out from your account, which will return you to your login screen, but if you want to completely shut off your machine, you've got a couple more steps to take.

```

2:09pm up 3 days, 3:41, 2 users, load average: 0.23, 0.12, 0.08
50 processes: 48 sleeping, 2 running, 0 zombie, 0 stopped
CPU states: 5.5% user, 2.3% system, 0.0% nice, 92.6% idle
Mem: 63016K av, 57964K used, 5052K free, 32316K shrd, 1684K buff
Swap: 72256K av, 256K used, 72000K free 34132K cached

  PID USER   PRI  NI  SIZE  RSS SHARE STAT  LIB  %CPU  %MEM  TIME COMMAND
 11992 root    19   0 11612  11M  1828 S      0   5.9  18.4  2:05 X
 12017 paulgall 6   0  2372  2372  1896 S      0   0.5   3.7   0:00 gnome-sessio
 12281 paulgall 2   0   816   816   628 R      0   0.5   1.2   0:00 top
 12284 paulgall 1   0  1808  1808  1100 S      0   0.3   2.8   0:00 xv
 12020 paulgall 0   0  1792  1792  1460 S      0   0.1   2.8   0:01 gnome-smprox
 12041 paulgall 0   0  4544  4544  2912 S      0   0.1   7.2   0:01 panel
    1 root     0   0   372   372   312 S      0   0.0   0.5   0:03 init
    2 root     0   0     0     0     0 SW     0   0.0   0.0   0:00 kflushd
    3 root    -12 -12     0     0     0 SW<    0   0.0   0.0   0:00 kswapd
    4 root     0   0     0     0     0 SW     0   0.0   0.0   0:00 md_thread
    5 root     0   0     0     0     0 SW     0   0.0   0.0   0:00 md_thread
   419 root     0   0   272   272   224 S      0   0.0   0.4   0:00 mingetty
   241 root     0   0   408   396   344 S      0   0.0   0.6   0:00 ypbind
    46 root     0   0   324   320   272 S      0   0.0   0.5   0:00 kerneld
   227 bin      0   0   356   352   280 S      0   0.0   0.5   0:00 portmap
   167 root     0   0   412   412   336 S      0   0.0   0.6   0:03 dhcpcd
   242 root     0   0   552   548   440 S      0   0.0   0.8   0:00 ypbind
  
```

Figure 15: The command `top` shows you running processes

Here's why: Even though you may not be typing, listening to music or browsing with Netscape, your machine is still working on a variety of processes in the background. (A process is a program which is being executed. Multiple processes are running all the time on your system.)

Tip: Curious to take a peek? Just go to a shell prompt and type `top`. You'll see the processes that are currently running (see Figure 15). To quit this view, type `[Q]`.

(To learn more about the `top` command, type `man top` at the shell prompt; to move forward a screen, press the `[Spacebar]`; to move back a screen, press `[B]`; to quit, press `[Q]`. You'll learn more about these "man pages" later in this chapter.)

Like a faithful assistant, your Linux system is carrying out tasks silently all the time. You can't just turn out the lights and lock the door on your assistant. Instead, you've got to give them time to put away their work and make sure everything's in its proper place before saying "good night."

To shutdown or reboot while you're in GNOME, exit from your X session (**panel** -> **Log out**). Once you're at the login screen, left-click on **Options**, and select **Halt** or **Reboot**.

You'll then be asked whether you want to stop or restart your machine. Choose `Hal`t to shutdown your machine; choose `Restart` to restart, or "reboot" your machine.

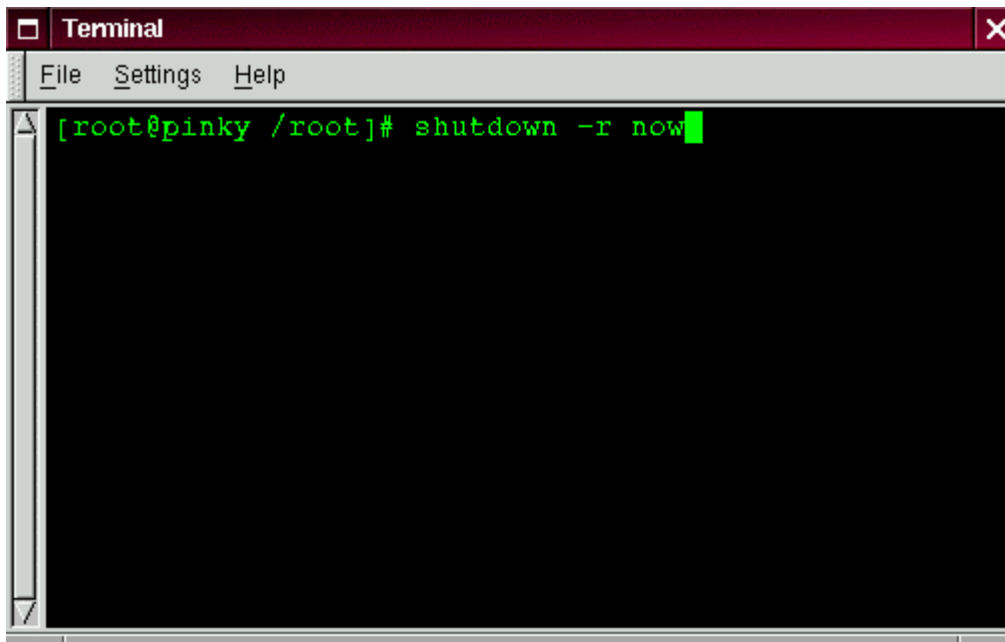


Figure 16: The shutdown command at the shell prompt

At the shell prompt, you can reboot or halt your system from your root account. To reboot from the prompt, type:

```
shutdown -r now
```

(See [Figure 16](#).)

Or, if you want to exit from your system and turn off your machine, type:

```
shutdown -h now
```

The `-r` option stands for "reboot," while the `-h` option means "halt." Stating `now` means that you want to perform this action immediately.

Please Note: Remember to save your work and exit from any applications which may be running before you perform a shutdown from the shell prompt, because you could lose work.

If you choose to halt the system, you'll see a list of messages about which services are stopping; then, you'll see:

```
The system is halted
```

Now everything's put away and it's safe to turn off your computer.

Tip: Try substituting `+5` for `now`; you'll find that you've just commanded your assistant to put everything away and stop working in five minutes.

You can learn more about the `shutdown` command by typing:

```
man shutdown
```

at a shell prompt. You'll be presented with a "man page," which will tell you about this command.

To go forward a screen, press the [Spacebar]; to go back a screen, press [B]; and to quit, press [Q].

Summary: To shutdown or reboot from GNOME, from the log in screen, go to **Options** -> **Halt** or **Reboot**. From the shell prompt: Log in as root, and type `shutdown -r now` (to reboot) or `shutdown -h now` (to halt).

1.8 Pulling Yourself Up by the Boot

When you're logged in as the root account, you might want to take a few minutes to create a fresh "boot diskette" or copy the diskette you already have.

There are a number of reasons you should make a boot diskette: it can help you recover from a system failure, it can help you test a new kernel you've downloaded and compiled and it can help you share your computer with more than one operating system.

Tip: You can always use a copy of the boot diskette to form the first half of a rescue disk set. You'll need a boot diskette and a rescue diskette to enter rescue mode. To read more about rescue mode, see later in this chapter.

You were given the opportunity to make a boot disk when you installed Red Hat Linux. If you chose not to make a boot disk at installation, here's your chance to start from scratch.

For now, we'll make boot disks from the shell prompt while we're in an X session.

Go to the shell prompt: In GNOME, for example, left-click on the **GNOME footprint** on the panel, go to **Utilities** in the menu and click on one of the items marked **xterm** or on the **GNOME terminal** item.

Now, make sure you're logged in as root. At the shell prompt, if you see something like

```
[billy@localhost billy], for example, type:
```

```
[billy@localhost billy] su
Password: yourrootpassword
[root@localhost billy]#
```

This will allow you to change from your regular user account to the root account.

Please Note: when you're in the root account, you are commander in chief of everything on your system, so be careful.

Tip: The command `su` means substitute users, which lets you log into temporarily log in as another user.

Briefly, we'll find the Linux kernel version; then, we'll use the `mkbootdisk` command to make a bootable floppy from the kernel.

Put a standard diskette in the floppy drive.

Tip: In Linux, the floppy drive is referred to as `/dev/fd0`.

If you've previously used the diskette, remember: You will lose anything that had been on that diskette!

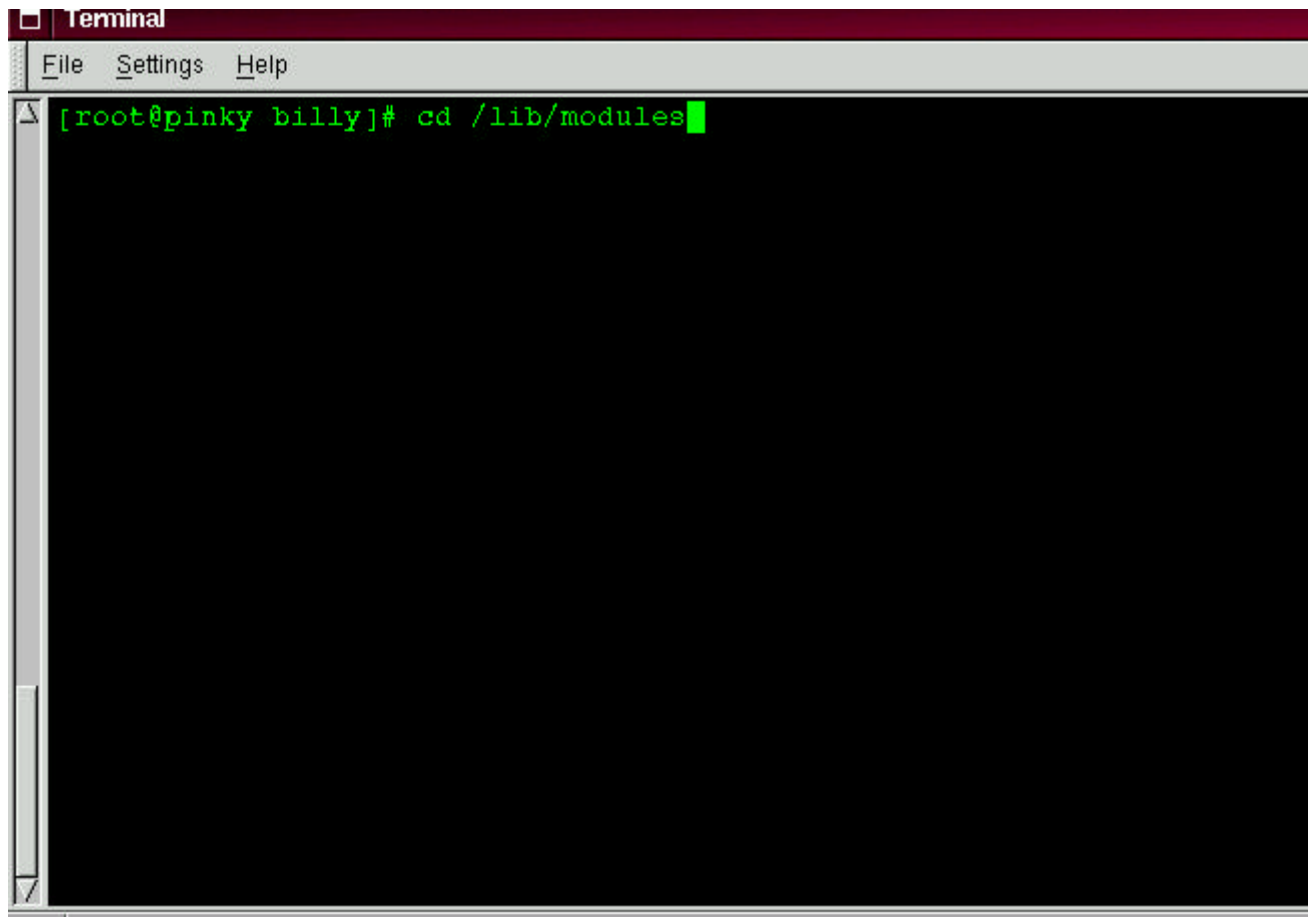


Figure 17: Changing directories to /lib/modules

Now, at the prompt, type:

```
cd /lib/modules
```

(Shown in figure [17](#).)

Now, type:

```
ls
```

The command `ls` will list the contents of a directory. (You can learn more about the `cd` and `ls` commands in Chapter [2](#).) For now, just type the commands as you see them.)

Here, you can find the kernel version of your Red Hat Linux system. The kernel is the heart of any Linux system. Your kernel version will be something similar to:

```
2.2.x-yy
```

(there will be several numbers after 2.2, as in 2.2.2-0.1 or 2.2.5-1).

Now that you've found the kernel version, you can tell the `mkbootdisk` command which kernel to copy to your floppy. (If you don't tell `mkbootdisk` where to copy the kernel, it will default to copying to the floppy in `/dev/fd0`.)

Just type:

```
mkbootdisk --device /dev/fd0 2.2.5-1
```

Then press [Enter].

Tip: If your screen becomes a little crowded with commands and ``command not founds," you can always start with a clean slate by typing `clear` at the prompt.

You're done.

Summary: As root, in a terminal window, `cd /lib/modules`; choose kernel

number; then type `mkbootdisk --device /dev/fd0 kernel.number`.

To clean the display, type `clear`.

1.9 A Good ``Man'' Is Easy to Find

As you investigate your new system, you're bound to have questions about commands and system services. One of the easiest ways to find out about how to use many commands or some applications is through the `man` command.

The word `man` stands for ``manual'', a series of online ``pages'' which can tell you the purpose of many commands. In a highly condensed format, `man` pages provide a summary of a command's purpose, the options available and the syntax which is used to issue the command.

If you count yourself among the ``newbies'' to Linux, you might not find `man` pages as useful as someone who's more accustomed to their terse delivery of information. But `man` pages can help steer you toward the proper way to use commands on your system. Even at this point, you can gain insight into your system by familiarizing yourself with the `man` pages. You'll certainly want to know how to use them eventually.

There are several ways to view the `man` pages: from GNOME's Help Browser, from an application called `xman` or from the shell prompt.

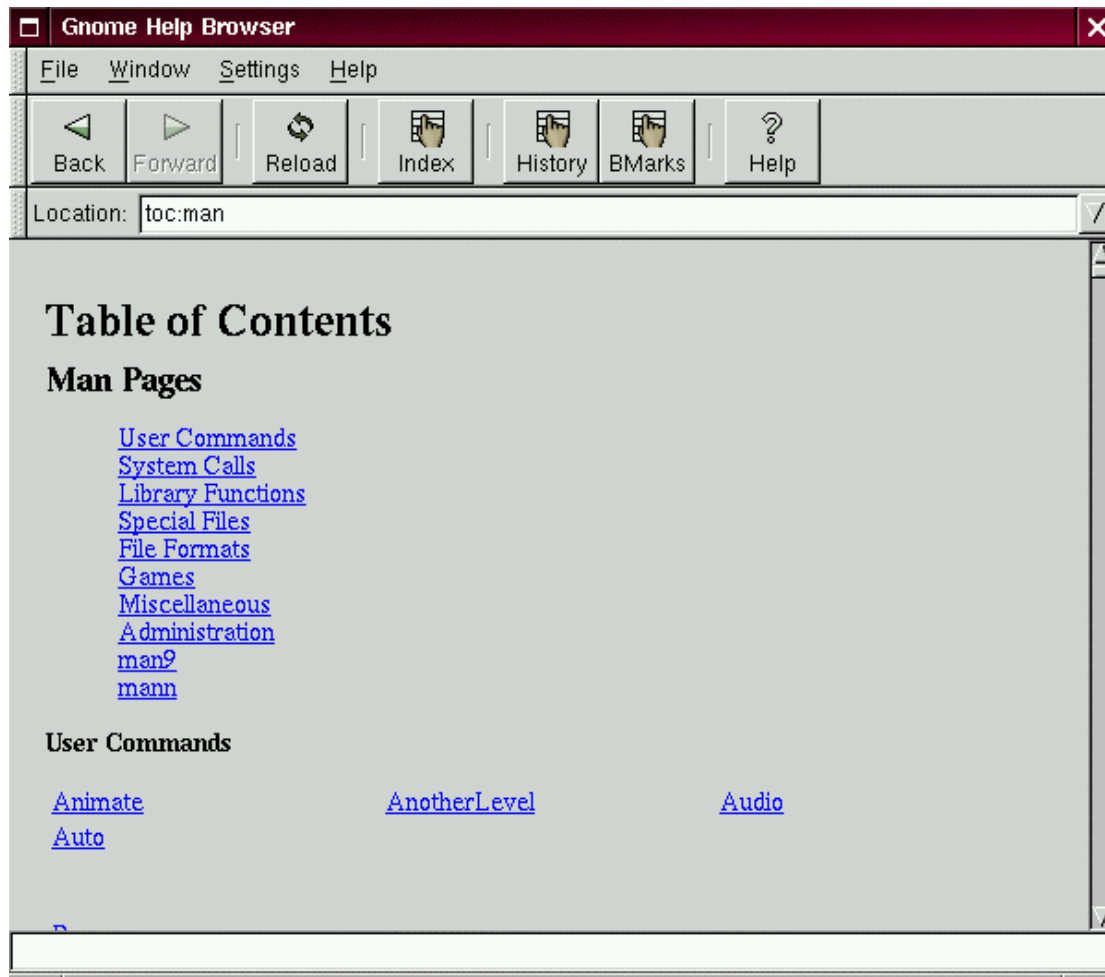


Figure 18: Man page index in GNOME's Help Browser

GNOME's Help browser:

To start GNOME's Help Browser, go to **Panel Help system**. The GNOME Help Browser will start. On the first page, you'll find links to the `man` pages and to other helpful documentation.

Tip: Read about how to get more from GNOME in the GNOME User's Guide,

which you'll find at the official GNOME website: www.gnome.org. For more information about man pages, take a look at the **Finding Documentation** chapter in the Official Red Hat Linux Installation Guide.

From xman:

Depending on your window manager, there will be different ways to access the graphical presentation of the man pages through the menus. (Window managers literally manage how the windows in your X session are presented.)

A quick way to start the manual browser, however, is to go to a shell prompt and type:

```
xman
```

which will start the X Window System manual browser. When the menu appears, click on **Manual Page**. From here, you'll have a number of options from which to find a man page: You can alphabetically display all the man pages on your system; search for them by command or by the section in which they appear.

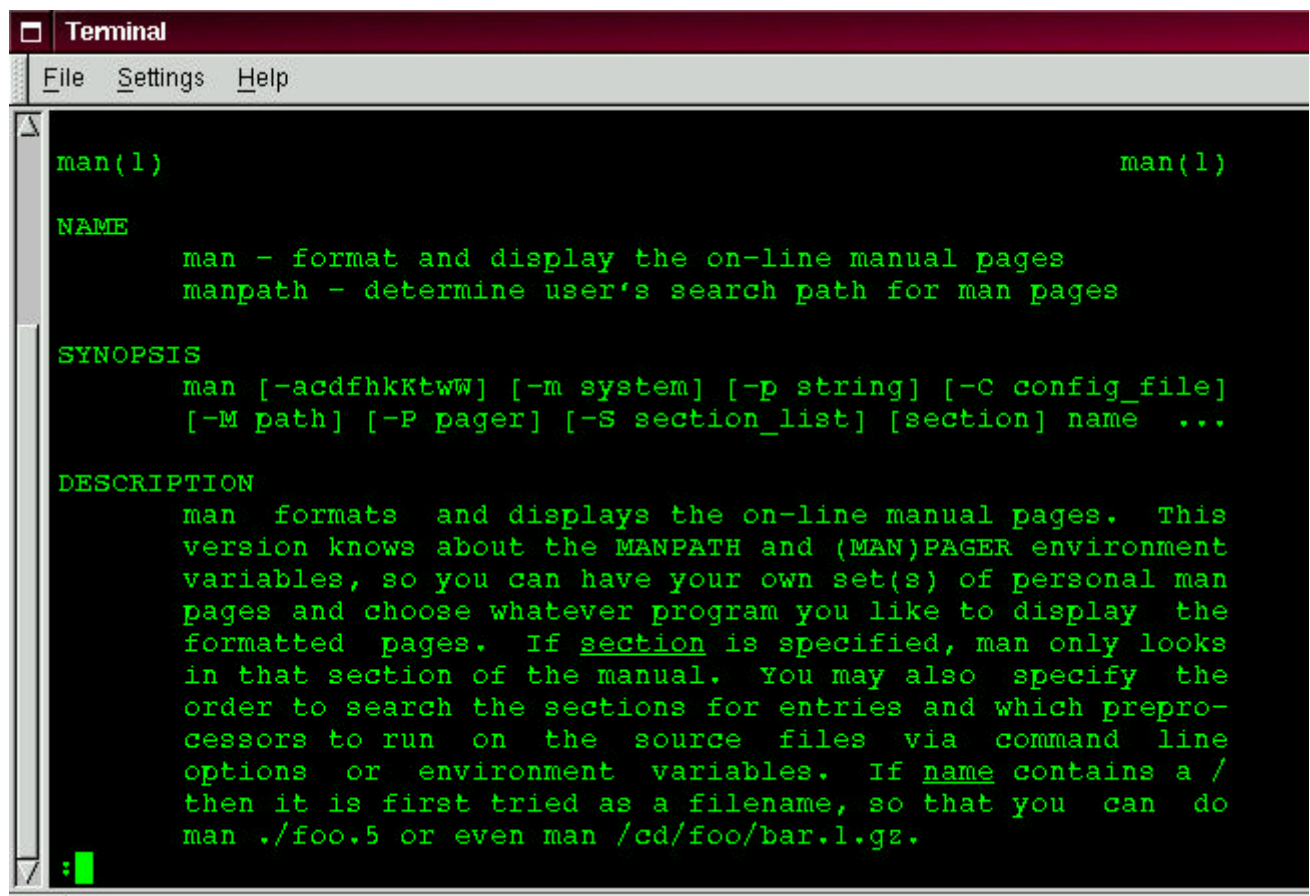
To see the scope of help available, bring up your manual browser, then click on **Manual Page** then, under **Options**, click on **Display Directory**. Here, you'll find the complete list of man pages available to you.

From the shell prompt:

If you're not in an X session, you can still read the man pages by typing

```
man pagename
```

at the shell prompt. To scroll forward through the document, press [Space]; to scroll back, press [B]. To quit the document, press [Q].



```
man(1) man(1)
NAME
  man - format and display the on-line manual pages
  manpath - determine user's search path for man pages

SYNOPSIS
  man [-acdfhkktww] [-m system] [-p string] [-C config_file]
  [-M path] [-P pager] [-S section_list] [section] name ...

DESCRIPTION
  man formats and displays the on-line manual pages. This
  version knows about the MANPATH and (MAN)PAGER environment
  variables, so you can have your own set(s) of personal man
  pages and choose whatever program you like to display the
  formatted pages. If section is specified, man only looks
  in that section of the manual. You may also specify the
  order to search the sections for entries and which prepro-
  cessors to run on the source files via command line
  options or environment variables. If name contains a /
  then it is first tried as a filename, so that you can do
  man ./foo.5 or even man /cd/foo/bar.1.gz.
```

Figure 19: Reading a man page at the shell prompt

Of course, like any good help system, the man command has its own man page. At the prompt, type

```
man man
```

to display the manual page (as shown in Figure 19).

If you want to print:

Sometimes, just viewing the man page on the screen isn't enough; you may want to have a printed copy in front of you. Although you could send the page to a printer, because of certain text formatting in the man pages, you'd likely end up with a document filled with ``garbage," symbols which didn't translate from your screen to the printer.

Before you print, then, you may have to ``strip" the formatting from the page, which you can do with the `col` command. (As you might guess, there's a man page for `col`, also.)

For example, to print a man page for the man, type:

```
man man | col -b | lpr
```

In detail, the above command ``sends" the output of the manual page entry through the `col` filter, which helps format the output for the printer. Then, the output from `col` is sent to the printer. This is called *piping*, and you can learn more about it in [Chapter 2](#).

1.10 What is Rescue Mode?

Rescue mode is a term used to describe a method of booting a small Linux environment completely from diskettes.

What follows in this section may help you recover from a problem at some point. A copy of these instructions is also available as `rescue.txt` on your Red Hat Linux 6.0 CD-ROM.

As the name implies, rescue mode is there to rescue you from something. In normal operation, your Red Hat Linux system uses files located on your system's hard drive to do everything -- run programs, store your files, and more.

However, there may be times when you are unable to get Linux running completely enough to access its files on your system's hard drive. By using rescue mode, it's possible to access the files stored on your system's hard drive, even if you can't actually run Linux from that hard drive.

Normally, you'll need to get into rescue mode for one of two reasons:

You are unable to boot Linux, and you'd like to fix it.

You are having hardware or software problems, and you want to get a few important files off your system's hard drive.

Let's take a closer look at each these scenarios.

Unable to boot Linux

-- Many times this is caused by the installation of another operating system after you've installed Red Hat Linux. Some other operating systems assume that you have no other operating systems on your computer, and overwrite the Master Boot Record (or MBR) that originally contained the LILO bootloader. If LILO is overwritten in this manner, you're out of luck -- unless you can get into rescue mode.

Hardware/software problems

-- There can be as many different situations under this category as there are systems running Linux. Things like failing hard drives and forgetting to run LILO after building a new kernel are just two things that can keep you from booting Red Hat Linux. If you can get into rescue mode, you might be able to resolve the problem -- or at least get copies of your most important files.

What do you need to get into rescue mode?

To get into rescue mode, you'll need a rescue disk set. These are two diskettes that contain the files necessary to boot into rescue mode.

If you elected to make a boot diskette while you were installing Red Hat Linux, you're halfway there!

The first diskette in a rescue disk set is this boot diskette.

Now on to the second diskette...

The second diskette is called the rescue diskette. It is produced by writing an *image file* onto a diskette.

The image file is called `rescue.img`, and is located in the `images` directory on the first Red Hat Linux CD-ROM.

To gain access to this file, you'll first need to mount your Red Hat Linux CD-ROM.

Start by inserting the CD-ROM in your system's CD-ROM drive. You'll need to do this while logged in as root.

Type the following command:

```
mkdir /mnt/cdrom
Now, type:
```

```
mount /dev/cdrom /mnt/cdrom
```

You may get an error message from the first command saying that the file exists. That's fine; we just want to make sure that there is a `/mnt/cdrom` directory on your system. The second command should issue an informational message that `/dev/cdrom` is being mounted read-only.

Please Note: Some systems may not recognize `/dev/cdrom`. If this is your case, you'll have to replace `/dev/cdrom` in the command with the appropriate device name for your CD-ROM.

Next, issue the following command (again, while logged in as root):

```
cd /mnt/cdrom/images
then type:
```

```
ls
```

to list the contents of the images directory.

You should see a file named `rescue.img`. This is the rescue diskette image file. Next, put a diskette in your first diskette drive, and enter the following command:

```
dd if=rescue.img of=/dev/fd0 bs=1440k
```

Your system's diskette drive should start writing to the diskette. After a minute or so, the `dd` command will complete, and you'll get your shell prompt back.

Wait for your diskette drive's access light to go out, and that's it!

You now have a rescue disk set. Label this diskette something like ``Red Hat Linux 6.0 rescue diskette'' and store it someplace safe.

Let's hope you never have to use it.

If you should ever need to use rescue mode, here's how.

Boot your system with the boot diskette in the first diskette drive. At the `LILLO Boot:` prompt, enter the word `rescue`. You will see the usual kernel messages as the Linux kernel starts up.

Eventually, it will ask you to insert the next diskette, and press `[Enter]`. Remove the boot diskette, insert the rescue diskette, and press `[Enter]`.

The rescue diskette will be read into memory. After a minute or so, you should see the shell prompt.

That's it -- you're in rescue mode!

Now what?

When it comes to rescue mode, that's a bit like asking, ``how long is a piece of string?'' What you require depends a great deal on what your system's problem is, your level of Linux expertise, and several variables we haven't even thought of yet. So we can't give you explicit instructions.

But we can tell you what programs you have access to while in rescue mode.

Here's the list:

<code>badblocks</code>	<code>bash</code>	<code>bzip2</code>
<code>cat</code>	<code>chmod</code>	<code>chroot</code>
<code>cp</code>	<code>cpio</code>	<code>dd</code>
<code>e2fsck</code>	<code>fdisk</code>	<code>grep</code>
<code>gunzip</code>	<code>gzip</code>	<code>head</code>
<code>ifconfig</code>	<code>init</code>	<code>ln</code>
<code>ls</code>	<code>lsmmod</code>	<code>mkdir</code>
<code>mke2fs</code>	<code>mknod</code>	<code>mount</code>
<code>mt</code>	<code>mv</code>	<code>open</code>
<code>pico</code>	<code>ping</code>	<code>ps</code>
<code>restore</code>	<code>rm</code>	<code>route</code>
<code>rpm</code>	<code>sed</code>	<code>sh</code>
<code>swapoff</code>	<code>swapon</code>	<code>sync</code>
<code>tac</code>	<code>tail</code>	<code>tar</code>
<code>traceroute</code>	<code>umount</code>	<code>vi</code>
<code>vim</code>		

You're likely to be unfamiliar with most, if not all of these commands. However, the commands do have *man pages*. Once you begin to feel more comfortable with commands, you should consider

familiarizing yourself them through the man pages. (You may not have that luxury if you have to use these commands...)

You've worked with some pretty useful commands for your Red Hat Linux system so far. You may not have known much about where those files were in the directory, however.

For more information about the Linux filesystem, including navigation and working with other useful commands to help you understand your system, turn to the next chapter.

2 You Are Here

Let's say you want to buy a pair of sneakers at a nearby shopping mall. You may not be familiar with the mall, but that shouldn't be a problem.

Why not? Because of the maps, which you can usually find near all the major entrances to the mall. The same can be said for your Red Hat Linux system: Navigation's easy once you know where you're going.

Tip: Make sure you've logged into your user account! Remember that unless you like to live on the wild - and dangerous - side, using a root account for all your activities is toying with disaster. If you haven't created your user account yet, turn to Chapter [1](#) and do it now. Really... no kidding...

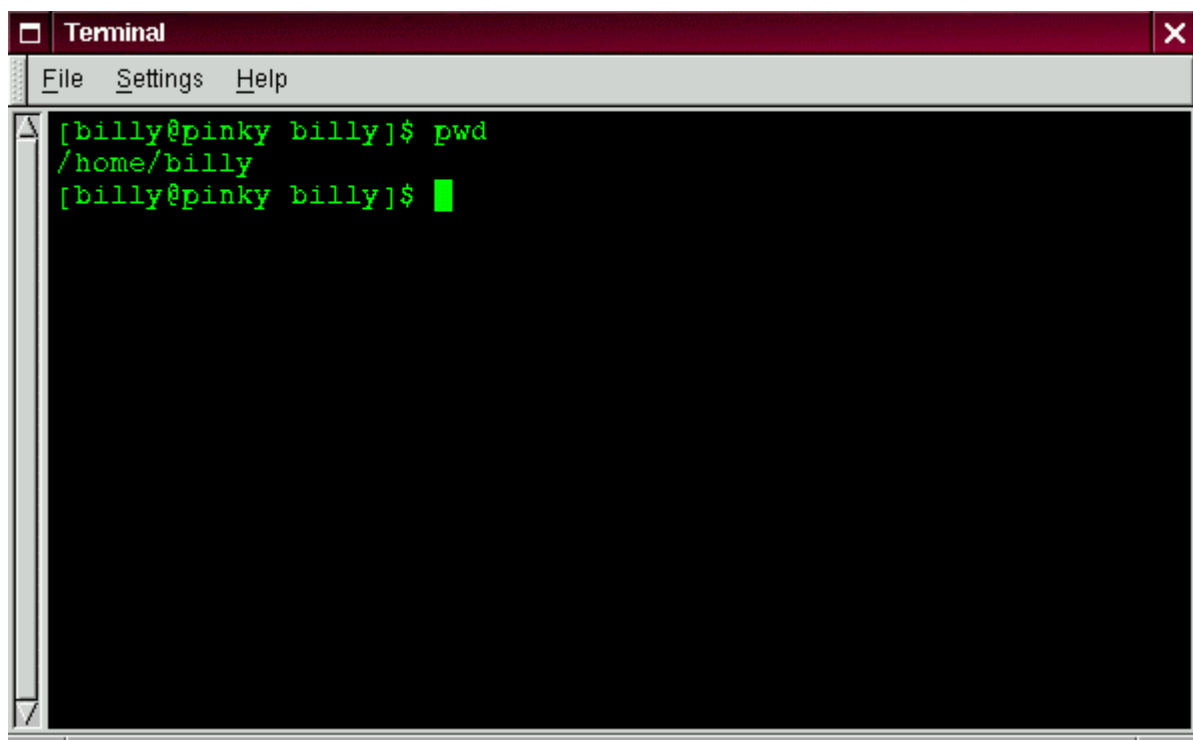
2.1 Finding Yourself With `pwd`

Sooner or later (probably sooner), when you start looking through directories, you're bound to ask, "Where the heck am I?" And you won't be speaking philosophically.

DOS can answer that question just by showing you at the prompt like:

```
C:\GAMES\Quake\ID1>
```

Your Linux system, by default, just shows your current directory.

A screenshot of a Linux terminal window titled "Terminal". The window has a menu bar with "File", "Settings", and "Help". The terminal content shows a prompt "[billy@pinky billy]\$" followed by the command "pwd". The output is "/home/billy". The prompt is followed by a green cursor block.

```
[billy@pinky billy]$ pwd
/home/billy
[billy@pinky billy]$ █
```

Figure 20: The command `pwd` shows you where you are

Try this: open an `xterm` window. You'll see something like:

```
[billy@localhost billy]
```

Now type:

```
pwd
```

What do you see? Something like

```
/home/billy
```

The command `pwd` stands for *print working directory*. When you typed `pwd`, you were asking your Linux system, "where am I?" Your system responded by "printing" the directory you're in to the monitor (as shown in Figure 20).

Seems easy, right? It ought to be; you'll be using `pwd` plenty as you look around. (Even Linux gurus depend on this little command.)

2.2 Getting From Here to There: `cd`

Whenever you want to change directories, all you've got to do is type:

```
cd
```

Go ahead, try it. In an `xterm` window, type:

```
cd
```

That didn't do much, did it? That's because you didn't tell your system where you wanted to go.

Whether you're going to a store in a mall or to visit relatives across the country, you've got to know how to get from one point to another. That is, you'll need to know the path to follow.

As with anything in life, the path -- or, *pathname* -- is basically the set of directions that takes you from one point to another. In the case of your Linux system (and in the DOS/Windows world, as well), you state a path to take you from one directory or file to another.

Let's try it again. Open an `xterm` window. Find yourself first with the `pwd` command. When you type your commands, your window will look like:

```
[billy@localhost billy] pwd
/home/billy
[billy@localhost billy]
```

Now that you see where you are, you can start to give your system the path to follow.

Well, almost...

Try typing:

```
cd home
```

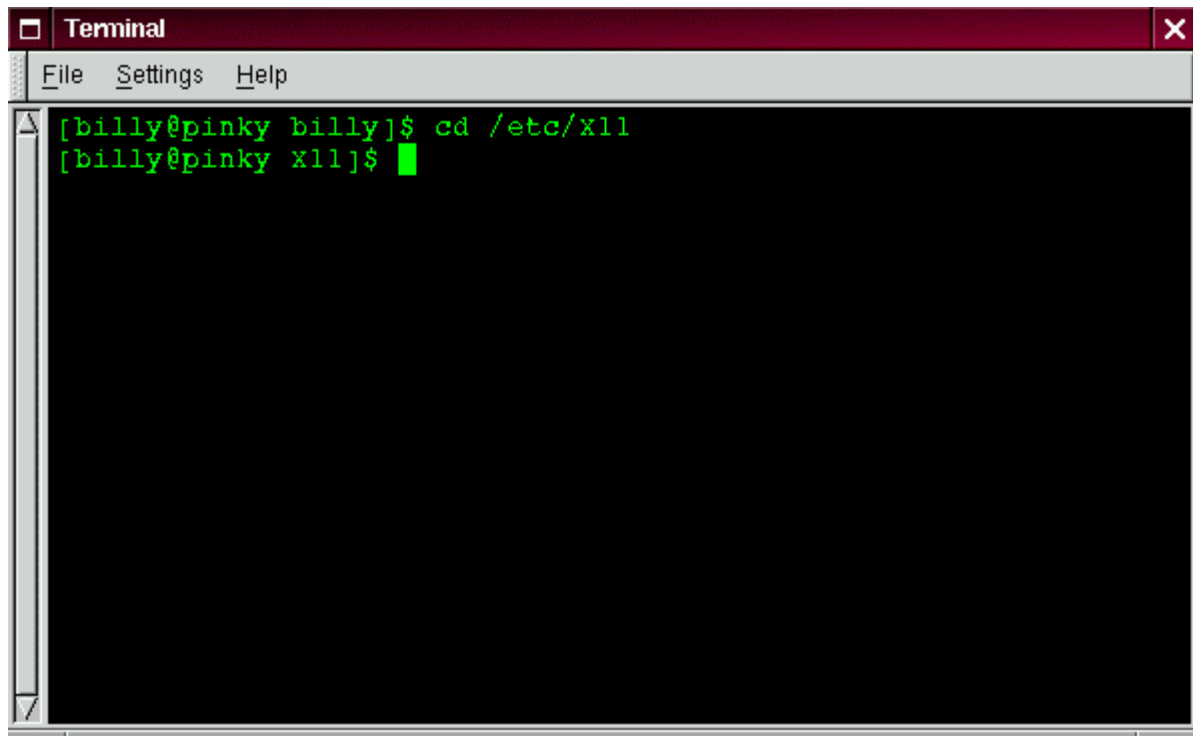
What happened? You know there's a directory called `home`, and you typed in the path. So what does this "no such file or directory" mean?

It means your path is incomplete.

Try typing:

```
cd /home
```

Now you've successfully changed directories and moved from your login directory into the subdirectory called `home`.

A terminal window titled "Terminal" with a menu bar containing "File", "Settings", and "Help". The terminal shows a user named "billy" at a host named "pinky". The user enters the command "cd /etc/x11" and the prompt changes from "[billy@pinky billy]" to "[billy@pinky x11]".

```
[billy@pinky billy]$ cd /etc/x11
[billy@pinky x11]$
```

Figure 21: Absolute pathnames state out the full path

The difference, of course, was adding the forward slash.

Let's take a second to look at the reason adding a slash made all the difference.

When you saw you were in `/home/billy`, you were looking at the full path -- or the absolute path from the root directory. You can think of the `billy` directory as being located two directories "down" from the root, which is the topmost level of your system.

So when you typed:

```
cd /home
```

you were actually saying "go to the root directory, then go to the directory called home, which is one directory below the root." You specified an absolute path to get to the home directory (see, for example, [Figure 21](#)).

Now, if you type:

```
cd /
```

you'll end up with a prompt that looks like:

```
[billy@localhost /]
```

That single forward slash means you're at the root. When you're at the root, you can't go any higher on your system (the same is true in DOS/Windows).

To get back to your login directory from the root directory by using the absolute path, just type:

```
cd /home/billy
```

You're home.

Using the absolute path is just one way to move around. Another method of getting from one point to another is by using the relative path (as in [Figure 1](#)).

Let's go back to the root directory:

```
cd /
```

Now, let's move *back* to your login directory using relative pathnames:

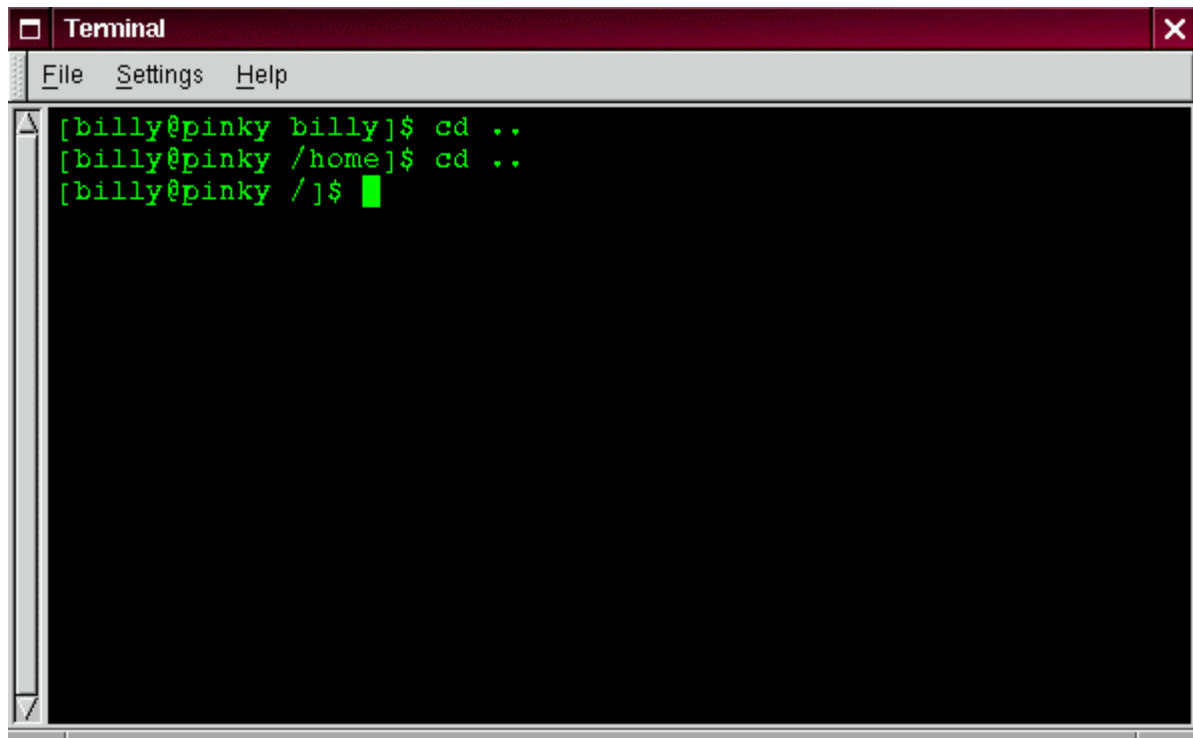
```
cd home/billy
```

Notice that the `/` is missing? That's because the root directory is the *parent* of the home directory, which means that the home directory is one step down from the root directory. Since home is the parent of the directory called `billy`, these two directories are separated with a `/`.

If you're in your login directory, you can move up one directory, to home, just by typing:

```
cd ..
```

The relative path describes the directory you want to `cd` to in terms which are relative to your current directory.

A terminal window titled "Terminal" with a menu bar containing "File", "Settings", and "Help". The terminal output shows three lines of commands and their results: 1. "[billy@pinky billy]\$ cd .." 2. "[billy@pinky /home]\$ cd .." 3. "[billy@pinky /]\$ █" The prompt changes from "billy" to "/" after the second command, indicating the root directory.

```
[billy@pinky billy]$ cd ..
[billy@pinky /home]$ cd ..
[billy@pinky /]$ █
```

Figure 22: Relative pathnames are 'relative' to your current position

When you typed `cd ..`, you were saying "go up one directory." The next directory up, from your login directory, was `home`.

Tip: When speaking of directories which hold other directories, you can refer to them as *parent directories*. In our case, `home` is the parent directory of `billy`.

Using two dots (`..`) when you `cd` is the same as stating you want to go to the parent of your current working directory. Try using a single dot. Type:

```
cd .
```

What happened? Not much. That's because using a single dot (`.`) is the same as specifying your current working directory.

The differences between absolute and relative paths can sometimes be pretty striking.

Getting back to our shopping mall analogy, if you were to give directions by using an absolute path, you might say something like:

```
"Get your car keys. Get in the car. Start the car. Pull out of the driveway. Drive to the corner..."
```

```
...And so on, until you're finally standing inside your favorite shoe store in the shopping mall.
```

When you're using a relative path, you're saying something like:

```
"The store's a couple miles from here, in the shopping mall."
```

That's quite an exaggeration, but you get the idea: As long as you know where you want to go in relation to where you are, you can use relative paths.

Tip: A path is absolute if the first character is a `/`; otherwise, it's a relative path.

You're now in the `home` directory, the parent of your login directory. Type:

```
cd ..
```

and you'll find yourself at the root directory.

Using relative paths, get yourself back to your login directory by typing:


```
cd home/billy
```

Doesn't look much different from absolute paths, does it? Notice, though, that there's no forward slash in front of home. In essence, you were saying, "go one directory down, to home, then go to billy, in the home directory."

Tip: Whenever you want to quickly jump back to your login directory just type `cd ~` anywhere in the system.

That wasn't much of a demonstration.

Try this: from your login directory, type:

```
cd ../../etc/X11
```

Now, you're in the directory X11, which is where you'll find configuration files and directories related to the X Window System.

Please Note: You can always type `pwd` to find out where you are in the directory tree. And you can get back to your login directory with the `cd ~` command.

Take a look at your last `cd` command. What you were really telling your system was, "go up to the parent directory, then up to that directory's parent directory (which is the root directory), then go to the `etc` directory and from there, to the X11 directory."

Using an absolute path would also get you to the X11 quickly. Type:

```
cd /etc/X11
```

and you're there.

Tip: Always make sure you know which working directory you're in before you state the relative path to the directory or file you want to get to. You don't have to worry about your position in the filesystem, though, when you state the absolute path to another directory or file.

Now that you're starting to get the hang of changing directories, see what happens when you change to root's login directory.

```
cd /root
```

Oops... You're not logged in as root, so you're "denied permission" to access that directory.

Denying access to the root and other users' accounts (or *login directories*) is one way your Linux system prevents accidental or malicious tampering. You'll find out more about file "ownership" and permissions later in this chapter.

Really want to change to the root login? Then you've got to use the `su` command. Type this series of commands:

```
[billy@localhost billy] su root
Password: (your root password)
[root@localhost billy]# cd /root
[root@localhost /root]#
```

As soon as you give the root password, you'll see the changes in your command prompt to show your new, superuser status: the `root` account designation at the front of the prompt and `#` at the end (as shown in [Figure 23](#)).

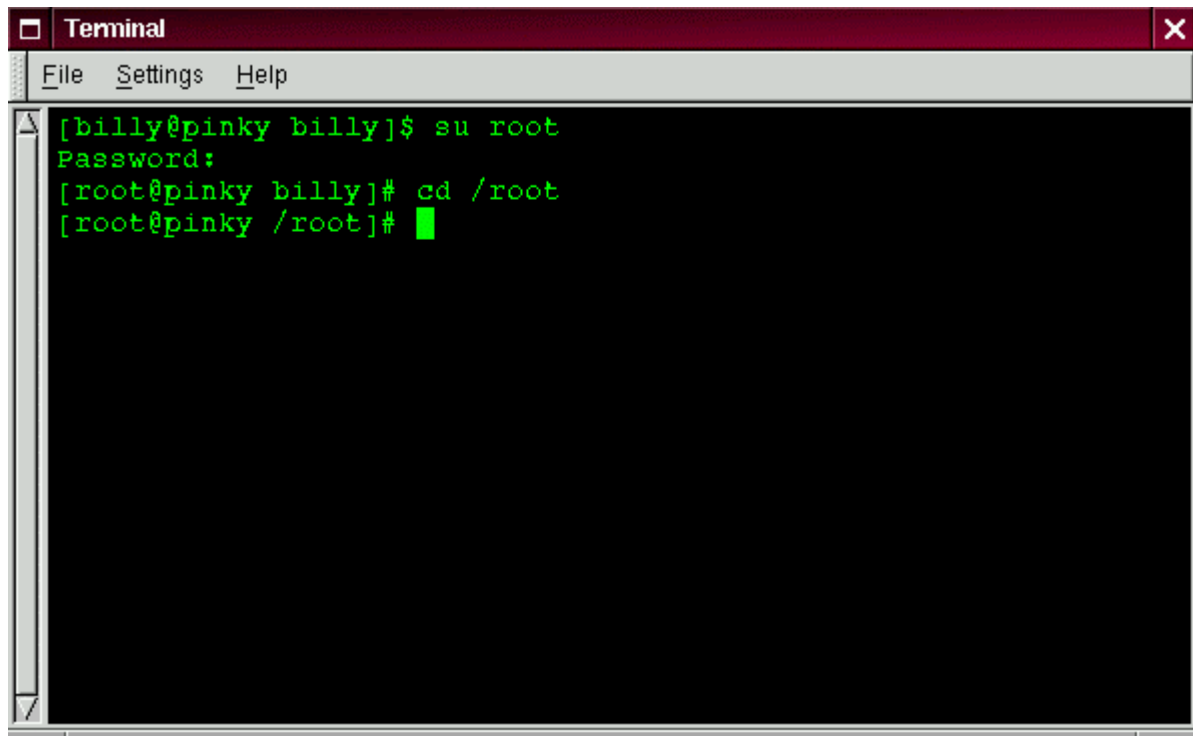


Figure 23: Becoming root

Now, when you `cd` to root's login directory, you'll be granted access. When you're done being root, just type `exit` at the prompt.

```
[root@localhost /root]# exit
exit
[billy@localhost billy]
```

Summary: To change directories using absolute pathnames, type `cd/directory/directory`; to change directories using relative pathnames, type `cd directory` to move one directory below, `cd directory/directory` to move two directories below, etc.; to jump from anywhere on the filesystem to your login directory, type `cd ~`; to change to the parent of the directory you're in, type `cd ..`. Use `.` to refer to your current directory.

2.3 Looking Around With `ls`

Now that you know how to move around, it's time to take a look at what's in the directories. But first, let's make sure you've got something you can look for in your login directory before we go any further.

You can start by creating an empty file. To do so, you can use a utility called `touch` at the shell prompt. Try it; type:

```
touch foo.bar
```

Now, in your login directory, you've got an empty file called `foo.bar`. You'll see it in a couple minutes.

Let's also create a new directory, using the `mkdir` command. At the prompt, type:

```
mkdir tigger
```

Now, you've created a directory called `tigger` in your login directory. From root, your new directory's absolute pathname would be `/home/yourlogin/tigger`, and your directory is the parent of `tigger`. (You can learn more about creating -- and removing -- files and directories in Chapter 3.)

Now, you're ready to go.

In the DOS world, using the `dir` command will display the contents of a directory.

The same can be said of Linux -- with some notable exceptions.

In Linux, `dir` won't fully display the contents of directories, and doesn't have the power or flexibility of the list command -- `ls`.

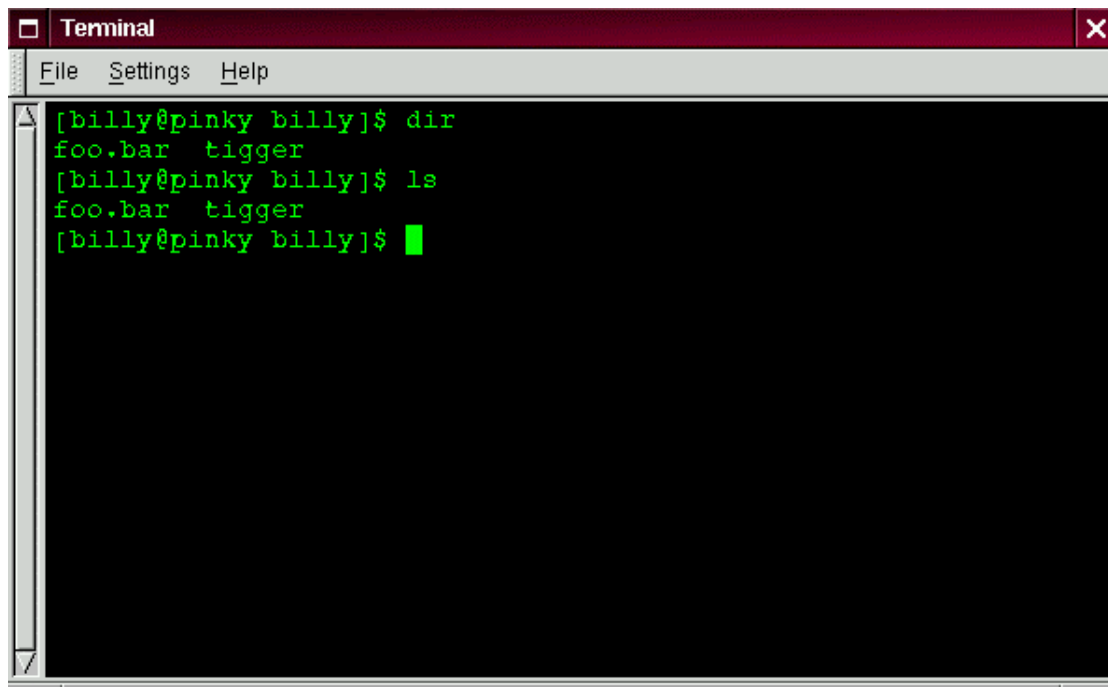
In your login directory, for example, type:

```
dir
```

Now, in the same xterm window, type:

```
ls
```

Looks the same (see Figure 24). You see, among other contents, your new file, `foo.bar` and the new directory, `tigger`.

A terminal window titled "Terminal" with a menu bar containing "File", "Settings", and "Help". The terminal shows the following commands and their outputs:

```
[billy@pinky billy]$ dir
foo.bar  tigger
[billy@pinky billy]$ ls
foo.bar  tigger
[billy@pinky billy]$ █
```

Figure 24: The `dir` and `ls` commands seem similar

But here the similarities end. Where `dir` shows you the contents of your directory, it doesn't actually show you everything. Even using the `ls` command, by itself, won't show you all the files in your directory. To see everything, you've got to call upon an option or two.

For example, in the same window that you'd used to type the `dir` and `ls` commands, now type:

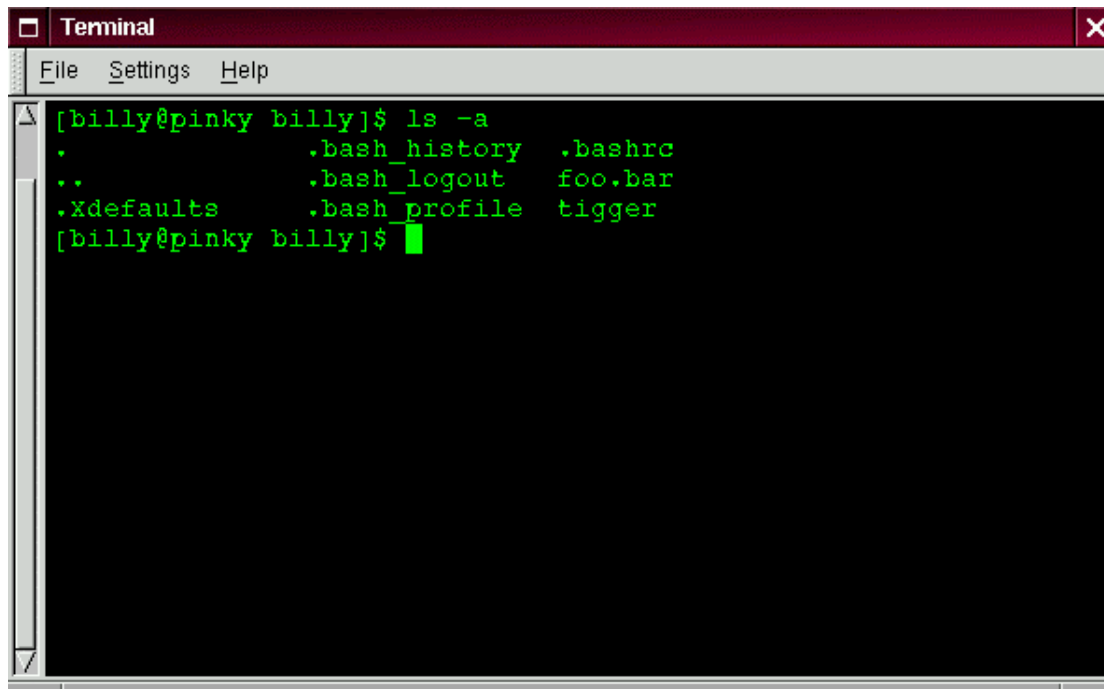
```
ls -a
```

Quite a difference. When you added the `-a` option, you were specifying that you wanted to list *all* the files in the directory (see Figure 25).

In fact, there are a multitude of options available with the `ls` command.

Tip: If you want to see all the options of the `ls` command, you can read the man page by typing `man ls` at a shell prompt. If you want to print the man page, type `man ls | col -b | lpr` at the prompt.

Why so many options? Because they can help you sort information according to your needs. For example, you can specify how files are displayed, see their *permissions* and much more.



```
[billy@pinky billy]$ ls -a
.          .bash_history  .bashrc
..         .bash_logout  foo.bar
.Xdefaults .bash_profile  tigger
[billy@pinky billy]$
```

Figure 25: The ls command with the -a option

When you typed `ls -a`, you probably noticed the files that begin with dots. These are called *hidden files* or, appropriately enough, *dot files*.

Hidden files are mostly configuration files which set preferences in programs, window managers, shells and more. The reason they're "hidden" is to help prevent any accidental tampering by the user.

Whenever a filename starts with a dot (`.`), it's a hidden file, and `ls` won't list it.

Viewing all the files can give you plenty of detail, but there's more you can uncover, simply by adding more than one option.

If we want to see the size of a file or directory, when it was created and more, we can just add the "long" option (`-l`) to our `ls -a` command.

Try it. Type:

```
ls -al
```

There's quite a bit more detail now. You can see the file creation date, its size, ownership, permissions and more.

You don't have to be in the directory whose contents you want to view, either.

Let's see what's in the `/etc` directory by typing:

```
ls -al /etc
```

Here, you'll get plenty of information about the contents of the `/etc` directory.

If you want to add color to your listing, just include the `--color` option.

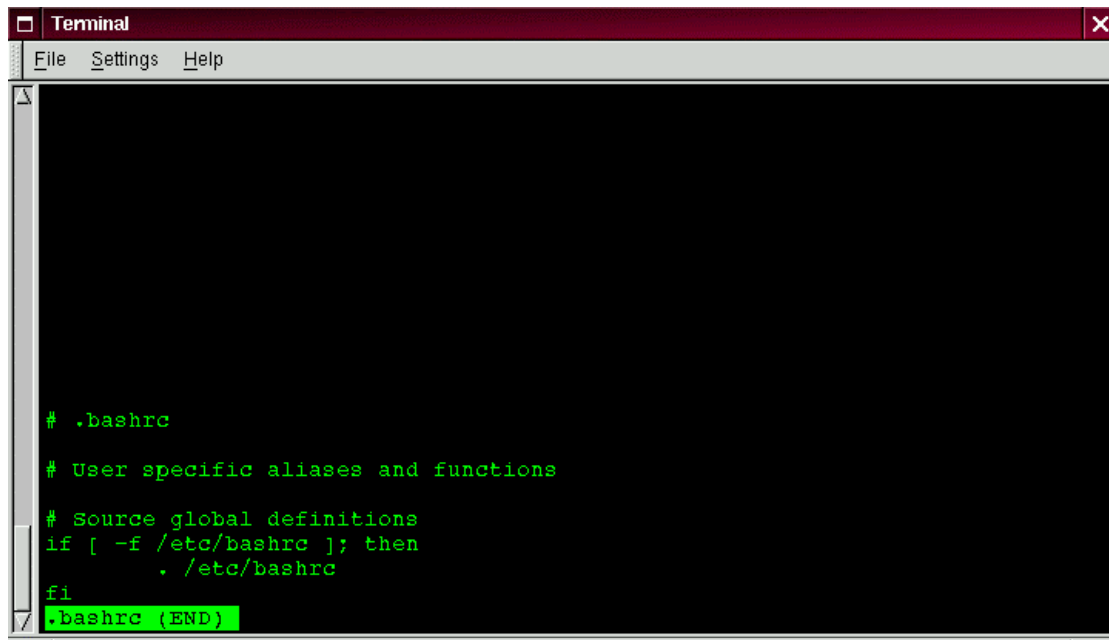
```
ls -al --color /etc
```

To some, adding `--color` does more than add a splash of color; it gives a clue about the types of files in a directory. For example, directories might all be a royal blue, program files would be green, and so on.

If you like what you see, here's how you can display the listing in color all the time. Briefly, we'll be adding one line to the `.bashrc` file in our login directory.

The `.bashrc` file is used by your shell when you login (an example `.bashrc` file is shown in [Figure 26](#)).

Now before we go any further...

A terminal window titled "Terminal" with a menu bar containing "File", "Settings", and "Help". The terminal displays the contents of the .bashrc file in green text on a black background. The text includes comments and code for sourcing global definitions. The last line, ".bashrc (END)", is highlighted in a green box.

```
# .bashrc
# User specific aliases and functions
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
.bashrc (END)
```

Figure 26: The .bashrc file

Remember that any changes you make to configuration files can cause you a world of grief if you've made a mistake and you don't have a backup copy of that file.

To make a backup copy, make sure you're in your login directory and in an xterm window, type:

```
cd
```

to get to your login directory. Then copy the .bashrc file, keeping it in the same directory, but with a name like .bashrc2.

```
cp .bashrc .bashrc2
```

When you type the above command, what you're saying is, "make a copy of the .bashrc file and name that copy .bashrc2."

Now, you have a backup copy of the unmodified .bashrc file in your login directory. If you make a mistake or have trouble, you can replace your .bashrc file by typing:

```
cp .bashrc2 .bashrc
```

at the shell prompt.

If you need to type this command, you'll be saying, "make a copy of the file .bashrc2 and name that copy .bashrc." The copy command here will overwrite the original .bashrc file -- and you'll still keep a copy of the original (and untouched) .bashrc file with the name of .bashrc2.

Now that we're prepared, we'll open .bashrc with Pico, a simplified *text editor*. (A text editor is a utility program that can create or modify files.) From an xterm window, type:

```
pico .bashrc
```

You should see something like this:

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
```

fi

It's a pretty short file. Those hash marks (#) are *comments*. Any text after them is ignored by the shell, but they are put there to help guide anyone who's editing or modifying files.

Bring your cursor under the line #User specific aliases and functions and type:

```
alias ls="ls -al --color"
```

So the full file ought to look something like this:

```
# .bashrc
```

```
# User specific aliases and functions
```

```
alias ls="ls -al --color"
```

```
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
```

fi

See Figure 27 for an example in Pico.

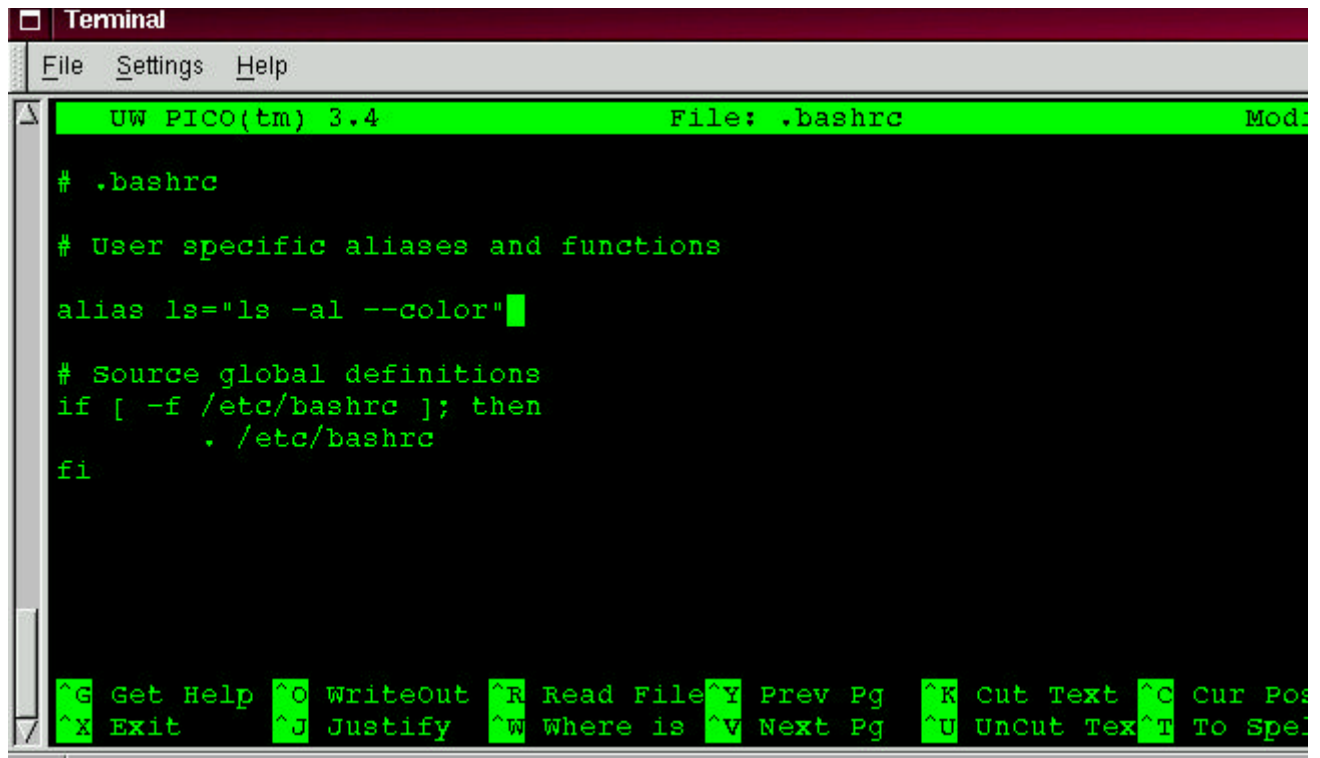


Figure 27: Adding an alias for the ls command to the .bashrc file

Double-check for any typos then, when you're satisfied with the changes, exit by pressing the [Ctrl] and [X] keys. You'll see, at the bottom of your editor screen, a message reading:

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES)?

Press [Y] for "yes." Now, another message will appear at the bottom:

File Name to write: .bashrc

Simply pressing [Enter] will save your changes to your .bashrc file.

You won't be able to see your changes take effect until you close your xterm window and open a new xterm. Once you do that, you'll see your modifications take effect.

Here's a short list of some popular options with `ls`. Remember, you can view the full list by reading the `ls` man page (`man ls`).

- **-a** -- all. Lists all the files in the directory, including the hidden files (`.filename`). The `..` and `.` at the top of your list refer to the parent directory and the current directory, respectively.
- **-l** -- long. Lists details about contents, including permissions (modes), owner, group, size, creation date, whether the file is linked to somewhere else on the system and where its link points.
- **-F** -- file type. Adds a symbol to the end of each listing. These symbols include `/` to indicate a directory; `@` to indicate a symbolic link to another file; `*` to indicate an executable file.
- **-r** -- reverse. Lists the contents of the directory from back to front.
- **-R** -- recursive. This recursive option lists the contents of all directories (below the current directory) recursively.
- **-S** -- size. Sorts files by their size.

A little later in this chapter, when we introduce you to *pipes* and *I/O redirection*, you'll discover that there are other ways to view the contents of a directory.

Summary: To see the contents of a directory, type `ls` at a shell prompt; typing `ls -a` will display all the contents of a directory; typing `ls -a --color` will display all the contents categorized by color.

2.4 A Larger Picture of the Filesystem

Every operating system has a method of storing its files and directories so that it can keep track of additions, modifications and other changes.

In Linux, every file is stored on the system with a unique name, in directories which can also hold other files and directories -- or, subdirectories.

You might think of the system as a tree-like structure, in which directories "branch off." Those directories may contain -- or be the "parent" of -- other directories which may hold files or directories of their own.

There wouldn't be a tree without a root, and the same is true for the Linux filesystem. No matter how far away the branches, everything is connected to the root, which is represented as a single forward slash (`/`).

It might seem confusing to have several references to "root" - the root account, the root account's login directory and the root directory (`/`), but think of it this way: The root login, who is the system administrator, is just as important to keeping things together in the system as the system's root (`/`).

Tip: Even though there are other Linux distributions, your Red Hat Linux system is likely to be compatible with them. The reason is because of the Filesystem Hierarchy Standard (also known as FHS). These guidelines help to standardize the way system programs and files are stored on all Linux systems. You can read more about the FHS at its website:

<http://www.pathname.com/fhs/>

As long as we're logged into our user account -- which will help prevent disastrous mistakes -- let's take a look around.

The first stop on this tour ought to be the root directory, which will give us a larger picture of where things are.

At the shell prompt, then, let's type:

```
cd /
```

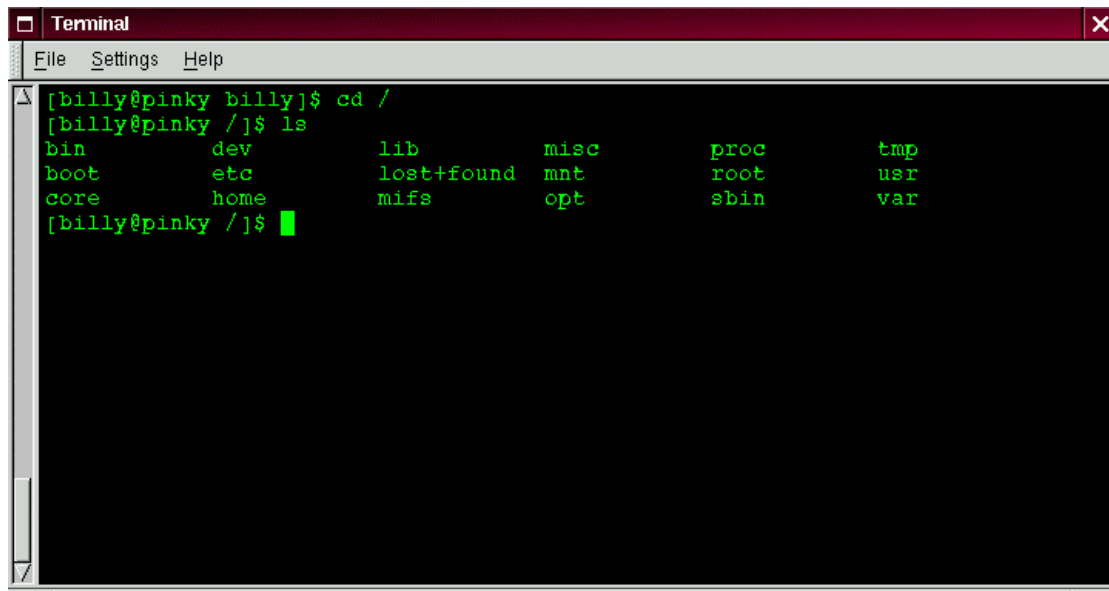
We'll see a prompt which looks like:

```
[billy@localhost ~]$
```

Now, let's see which directories "branch off" root by typing:

```
ls
```

Doesn't look like much, does it? Well, it's a little like viewing the tip of an iceberg. These are the parent directories of other directories, in which there may be other directories... and so on.

A terminal window titled "Terminal" with a menu bar containing "File", "Settings", and "Help". The terminal shows the following commands and output:

```
[billy@pinky billy]$ cd /  
[billy@pinky /]$ ls  
bin          dev          lib          misc         proc         tmp  
boot        etc          lost+found  mnt         root        usr  
core        home        mifs        opt         sbin        var  
[billy@pinky /]$
```

Figure 28: Getting of view of the directories from root

Here are just a few of the directories we're likely to find:

```
etc      lib      sbin
```

```
usr      var
```

There are more, but let's take a look in the `/etc` directory.

```
[billy@localhost ~]$ cd /etc
```

```
[billy@localhost /etc]$ ls
```

Here, among other type of files and directories, we'll find *configuration files*, which are files that help make programs work for our system, store our program and system settings and more.

Among the directories in here, you'll see `/etc/X11`, which also contains directories and configuration files for the X Window System.

In the directory `/etc/skel`, you'll find *skeleton* user files, which are used to populate newly created user accounts with standard, commonly used files.

That sounds a little gothic, perhaps, but here's what it means. When we were logged in as root, one of our first tasks was to create an account for ourselves.

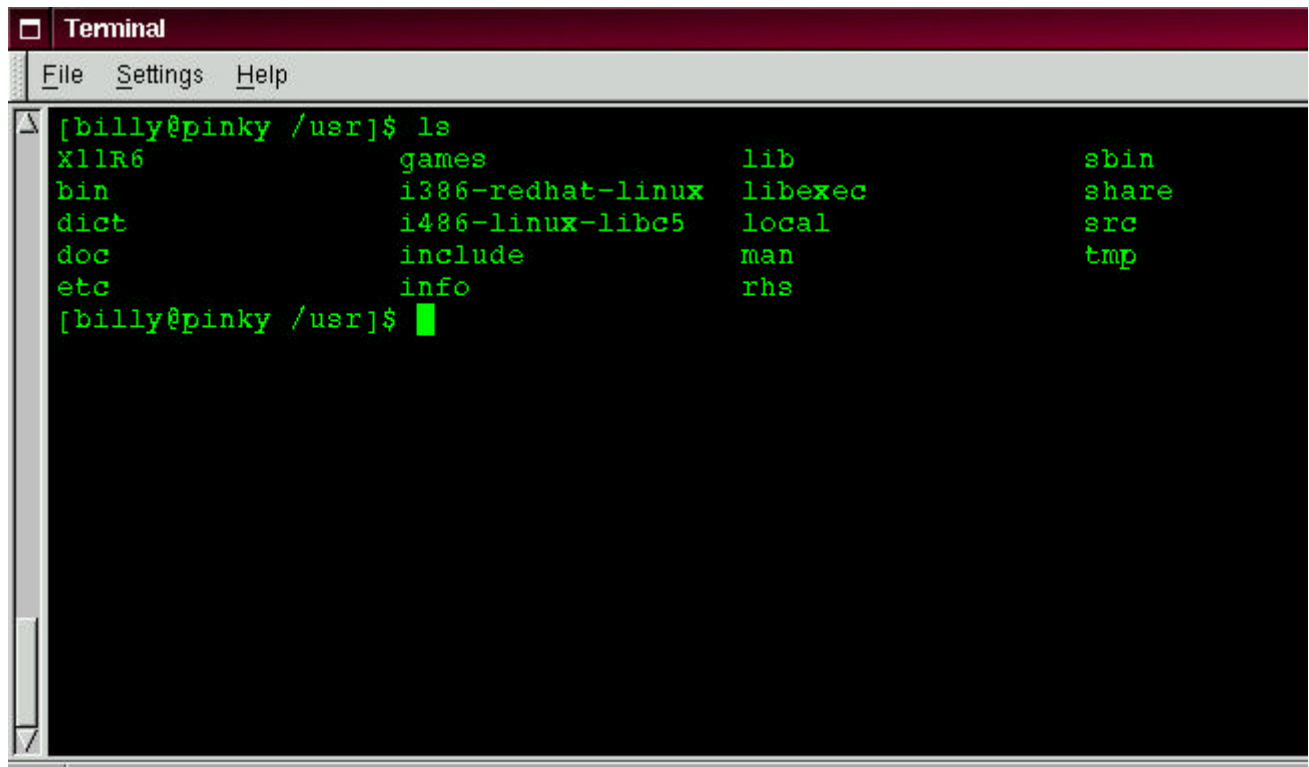
When our user account was created, files were taken from `/etc/skel` and placed into the new account. These files helped to "flesh out" the account (skeleton... flesh... get it?).

Let's look around a little in `/usr`. From our current location in `/etc/skel`, we can type:

```
[billy@localhost /skel]$ cd /usr
```

```
[billy@localhost /usr]$ ls
```

In `/usr`, we're going to find a number of directories which hold some of our system's most important programs, and files (see Figure 29).

A terminal window titled "Terminal" with a menu bar containing "File", "Settings", and "Help". The terminal shows the command `[billy@pinky /usr]$ ls` and its output: `x11R6`, `bin`, `dict`, `doc`, `etc`, `games`, `i386-redhat-linux`, `i486-linux-libc5`, `include`, `info`, `lib`, `libexec`, `local`, `man`, `rhs`, `sbin`, `share`, `src`, and `tmp`. The prompt `[billy@pinky /usr]$` is shown again at the end of the output.

```
[billy@pinky /usr]$ ls
x11R6      games      lib        sbin
bin        i386-redhat-linux  libexec    share
dict       i486-linux-libc5  local      src
doc        include     man        tmp
etc        info        rhs
```

Figure 29: The `ls` command in `/usr`

In `/usr/man` we'll find the system manual pages; other documentation which isn't covered by man pages will be found in `/usr/doc`.

In `/usr/X11R6`, we'll find files related to the X Window System, including configuration and documentation files.

Although we may think of something more literary when we hear the word "libraries," in `/usr/lib` we'll find files which are considered libraries for our system. In this context, libraries are files containing commonly-used snippets of code which can be shared by many programs.

Red Hat Linux uses the RPM (the **R**ed **H**at **P**ackage **M**anager) technology of software installation and upgrades. Using RPM, either from the shell prompt or through GnoRPM, is both a safe and convenient way to upgrade or install software.

(For more on using GnoRPM, see its chapter in the Red Hat Linux Installation Guide.)

However, once you become more comfortable with your system, there may be times when you'll want to install software that may not be available in RPM format. To minimize collisions with RPM-managed files, the best place to put such software is in `/usr/local`.

2.5 "Washing" the Window

After even one `ls` command in an `xterm` window, things might start feeling a little crowded. We can always exit from the terminal window and open a new one, but here's a quicker way to wipe the slate clean.

Just type:

```
clear
```

at the shell prompt. The `clear` command does just as advertised: it clears the terminal screen.

Sometimes, you may accidentally open a program file or some other non-text file in a terminal window. Once you close the file, you could find that the text you're typing doesn't match with the output on the monitor.

In such cases, you simply have to type:

```
reset
```

to return the window to its default values.

Summary: To clear clutter in a console or `xterm` window type `clear`; to return an `xterm` window to its default display properties, type `reset`.

2.6 Using `cat`

There's a handy utility which can help you keep short lists, gather those lists together and, at the same time, show you a little of the power behind your Red Hat Linux system.

The utility is called `cat`, short for "concatenate," which means that it strings files together.

But `cat` can also perform a quick demonstration of two important terms: *standard input* and *standard output*.

Standard input and standard output direct input and output (often referred to as *I/O*) to the user. If a program reads from standard input, it will, by default be reading from the keyboard. If a program writes to standard output, by default it will be writing to the screen.

Let's start `cat` to see what this means. At the shell prompt, type:

```
cat
```

The cursor moves to a blank line. Now, in that blank line, let's type:

```
stop at sneaker store  
and press the [Enter] key. Suddenly, your screen looks like:
```

```
[billy@localhost billy] cat  
stop by sneaker store  
stop by sneaker store
```

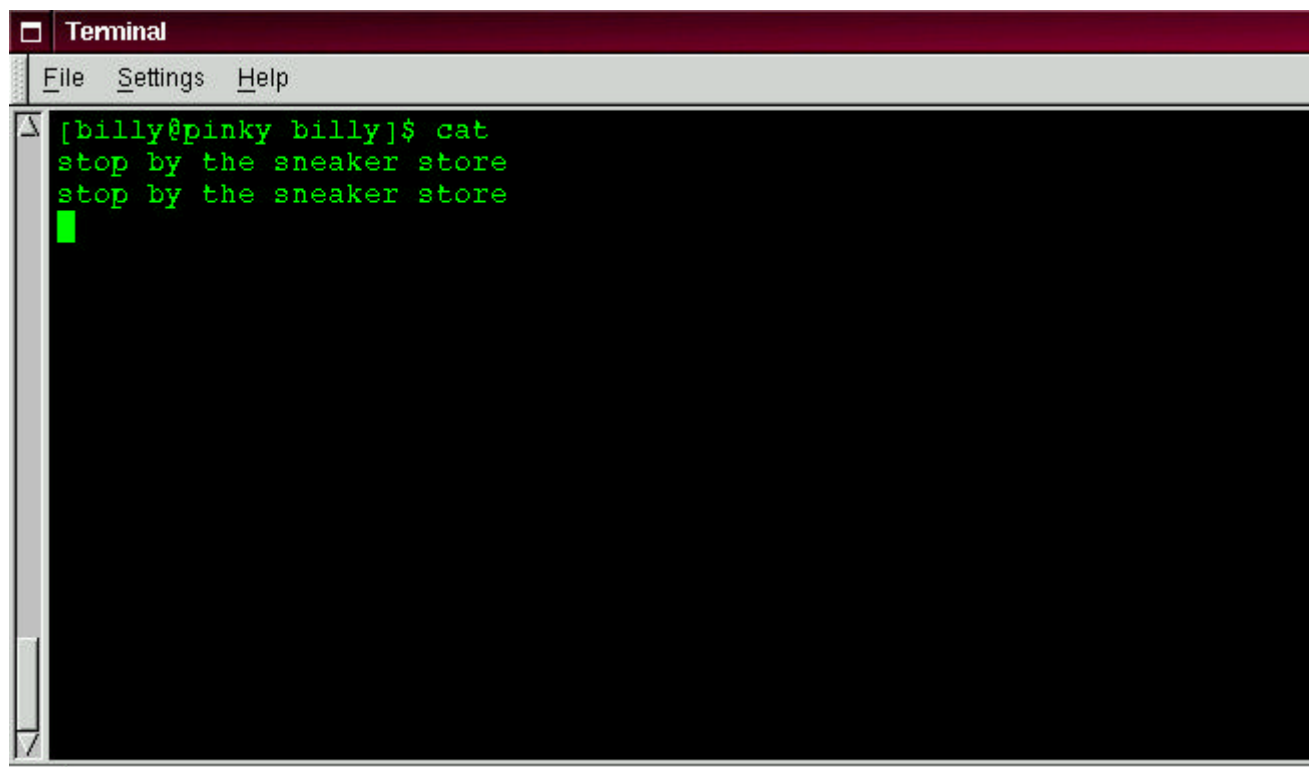


Figure 30: `cat` demonstrates standard input and standard output

To quit `cat` now, just move the cursor to a blank line by pressing [Enter] then press the [Ctrl] and [D] keys at the same time.

So it's not too exciting. But `cat` has just demonstrated the definition of standard input and standard output. Your input was read from the keyboard (standard input), and that input was then directed to your terminal (standard output).

Summary: Standard input is often text which is entered from the keyboard. Standard output is the place where information is sent, such as your terminal (as shown in Figure 30).

2.7 Using Redirection

Now that we have a handle on what standard input and standard output are, it's time to expand a little. Redirection means causing the shell to change what it considers standard input or where the standard output is going.

We used `cat` before to demonstrate the idea behind standard input and standard output. Now, let's use `cat` to see how standard output can be redirected.

To redirect standard output, we'll use the `>` symbol. Placing `>` after the `cat` command (or after any utility or application that writes to standard output) will direct its output to the filename following the symbol.

Let's try it. In an xterm window type:

```
[billy@localhost billy] cat > sneakers.txt
buy some sneakers
then go to the coffee shop
then buy some coffee
```

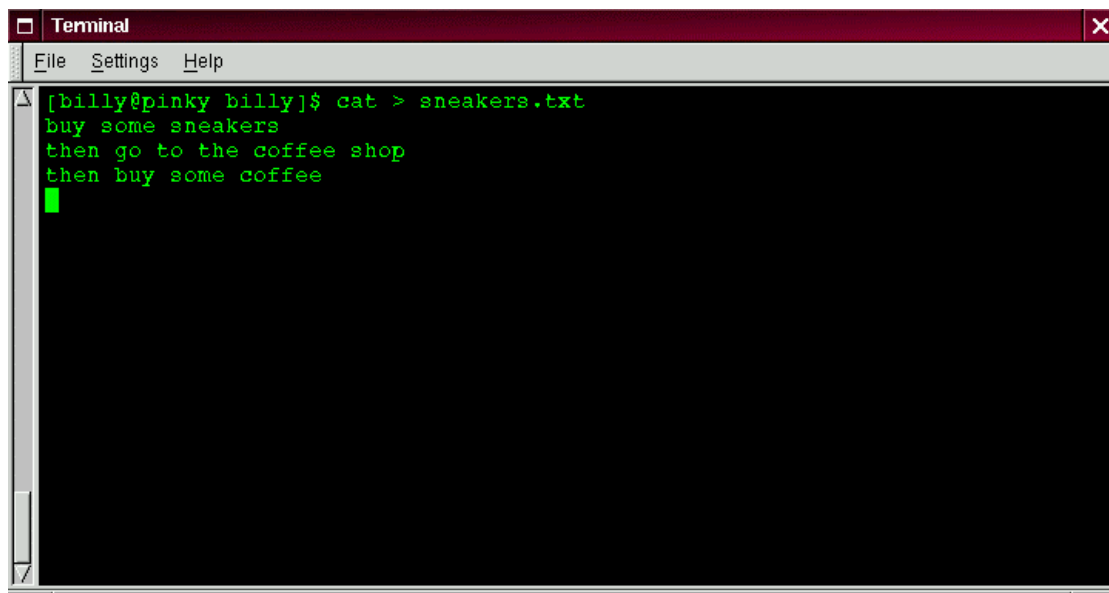


Figure 31: Redirecting the output to a file

Now press [Enter] to go to an empty line, and use the [Ctrl] and [D] keys to quit `cat`. Notice the difference (see Figure 31)? For one thing, there are no double entries. That's because the standard output from `cat` was redirected. That redirection was to a brand new file you made called `sneakers.txt`.

You can find the file in your login directory (may we suggest using `ls` if you want to see it listed?). You can even use `cat` to read the file, by typing:

```
cat sneakers.txt
at the prompt.
```

Tip: Be careful when you redirect the output to a file, because you can easily overwrite an existing file! Make sure the name of the file you're creating doesn't match the name of a pre-existing file, unless you want to replace it.

Let's use output redirection for another file and call it `home.txt`.

```
[billy@localhost billy] cat > home.txt
bring the coffee home
take off shoes
put on sneakers
make some coffee
relax!
```

Now, on an empty line, use the [Ctrl] and [D] keys again to quit cat.
We can check the file again by typing:

```
cat home.txt
at the prompt.
```

Let's use cat again to join home.txt with sneakers.txt and redirect the output of both files to a brand new file we'll call saturday (you'll find an example in [Figure 32](#)).

```
[billy@localhost billy] cat sneakers.txt home.txt > saturday
That's it.
```

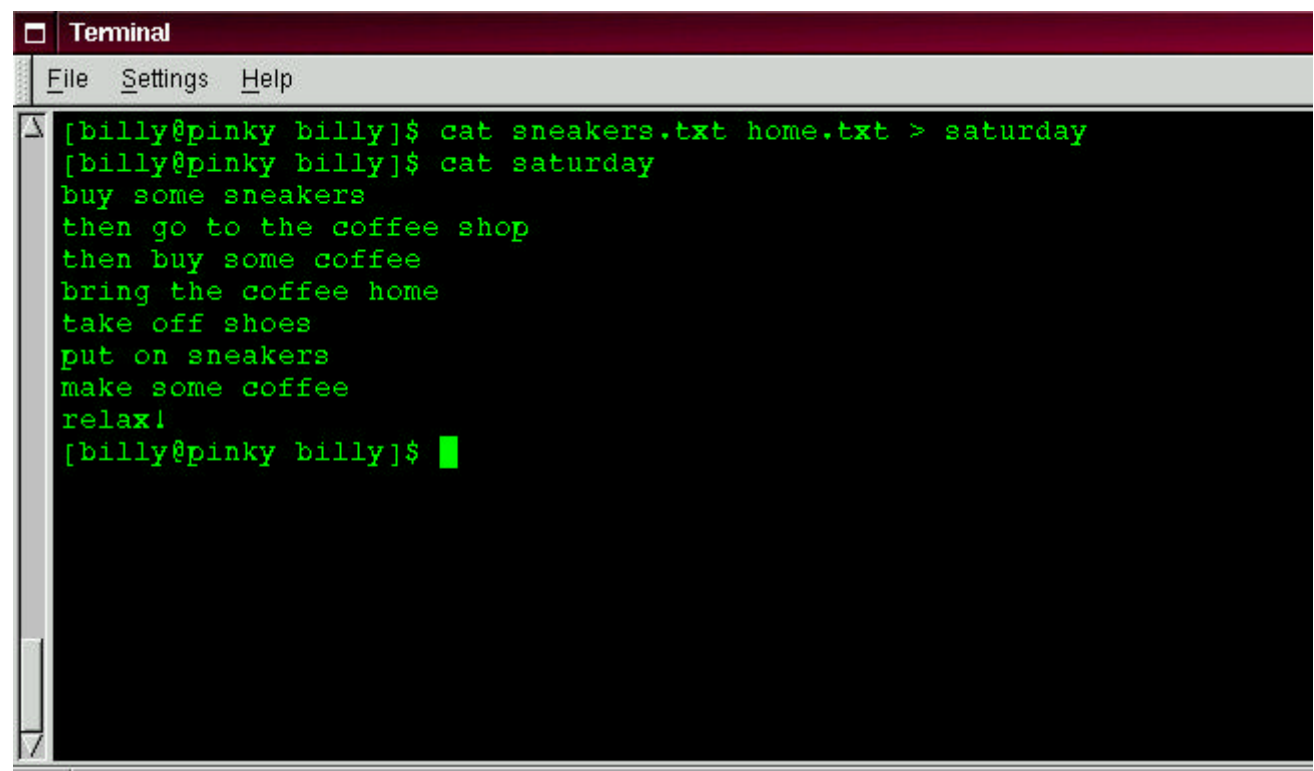
A terminal window titled "Terminal" with a menu bar containing "File", "Settings", and "Help". The terminal shows a shell prompt [billy@pinky billy]\$ where the command 'cat sneakers.txt home.txt > saturday' is entered. The next prompt shows the command 'cat saturday' which outputs the contents of both files concatenated: 'buy some sneakers', 'then go to the coffee shop', 'then buy some coffee', 'bring the coffee home', 'take off shoes', 'put on sneakers', 'make some coffee', and 'relax!'. The prompt returns to [billy@pinky billy]\$ with a green cursor.

Figure 32: Joining files and redirecting the output

Now it's time to check our handiwork. Type:

```
[billy@localhost billy] cat saturday
and you should see something like this:
```

```
[billy @localhost billy] cat saturday
buy some sneakers
then go to the coffee shop
then buy some coffee
bring the coffee home
take off shoes
put on sneakers
make some coffee
relax!
[billy @localhost billy]
```

You can see that cat has added home.txt where sneakers.txt left off.

Tip: Creating and combining short files with cat can be a convenient

alternative to using a text editor like `pico`.

Summary: By using the output redirection symbol (`>`) you can send the output to a file instead of the terminal. The `cat` utility can be used along with output redirection to join files together into a single, unified file with one filename.

2.8 Appending Standard Output

There's a neat twist to output redirection which allows you to add new information to the end of an existing file. Similar to when you used the `>` symbol, you tell your shell to send the information somewhere other than standard output.

However, when you use `>>`, you're *adding* information, rather than replacing it.

The best explanation is a demonstration, so let's take two files which have already been created -- `sneakers.txt` and `home.txt` -- and join them by using the append output symbol. We want to add the information in `home.txt` to the information already in `sneakers.txt`, so we type:

```
cat home.txt >> sneakers.txt
```

Now let's check the file by typing:

```
cat sneakers.txt
```

And there it is -- with the contents of `home.txt` at the end.

What we were saying when we typed that command was, "append the output from the file `home.txt` to the file `sneakers.txt`."

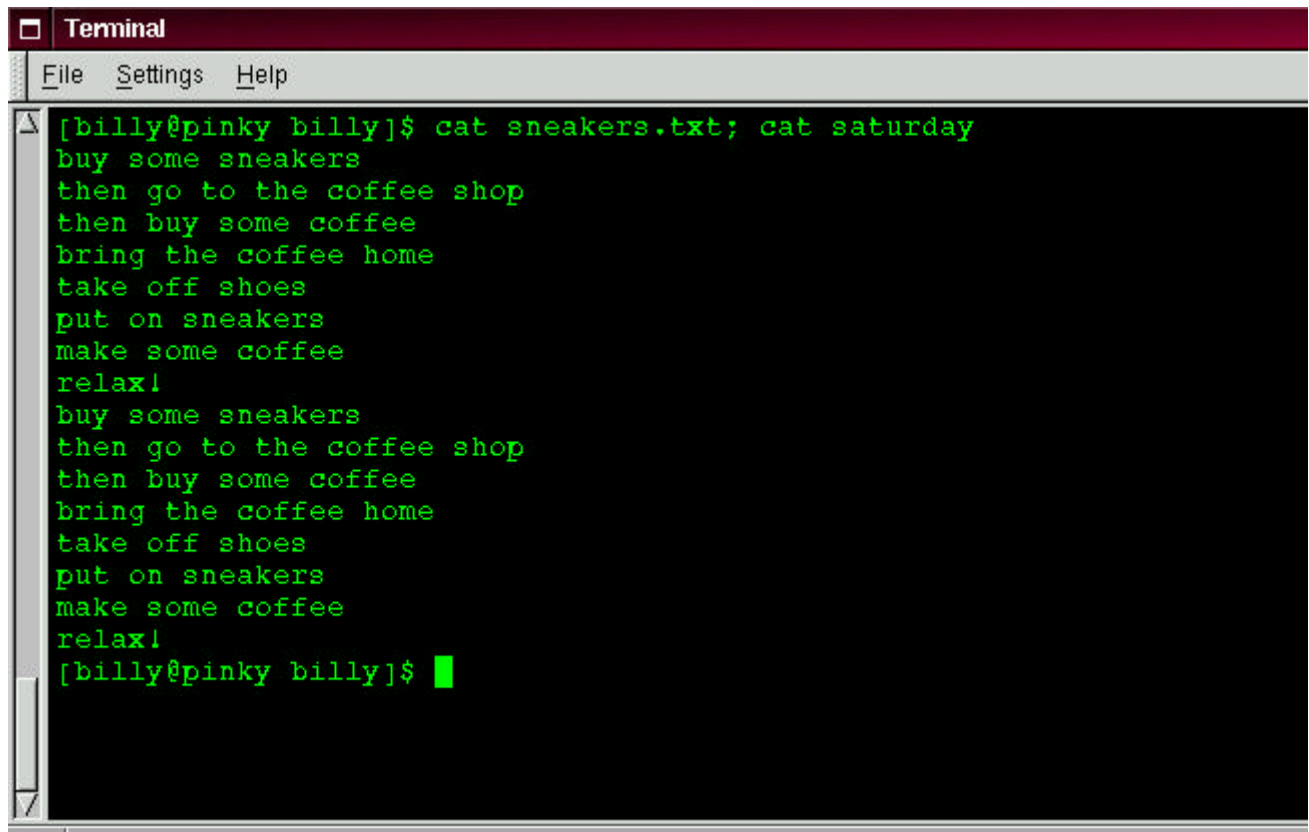
By appending the output, we've saved ourselves a step or two (and a bit of disk clutter) by using existing files, rather than creating a new file.

Compare the results of the files `sneakers.txt` and `saturday` now, and you'll see that they're identical. To make your comparison, just type:

```
cat sneakers.txt; cat saturday
```

The contents of both files will be displayed - first `sneakers.txt`, then `saturday` (as shown in [Figure 33](#)).

Tip: Remember that when you append output, you've got to include the double greater-than symbols (`>>`). Otherwise, you'll end up replacing the very file to which you want to append information!



```
Terminal
File Settings Help
[billy@pinky billy]$ cat sneakers.txt; cat saturday
buy some sneakers
then go to the coffee shop
then buy some coffee
bring the coffee home
take off shoes
put on sneakers
make some coffee
relax!
buy some sneakers
then go to the coffee shop
then buy some coffee
bring the coffee home
take off shoes
put on sneakers
make some coffee
relax!
[billy@pinky billy]$
```

Figure 33: Stringing commands and comparing files

(By the way, if you're curious about the use of the semi-colon in that last command, read on. We'll cover that later in this chapter.)

Summary: To append output, use two greater-than symbols (>>). For example: `cat addthisfile >> tothisfile.`

2.9 Redirecting Standard Input

Not only can you redirect standard output, you can perform the same type of redirection with standard input.

Here's how it works:

When you use the redirect standard input symbol <, you're telling the shell that you want a file to be read as input for a command.

We can use a file we've already created to demonstrate this idea. Just type:

```
cat < sneakers.txt
```

Because we used the less-than symbol (<) to separate the `cat` command from the file, the output of `sneakers.txt` was read by `cat`.

2.10 Pipes

No, we're not going to start talking about plumbing here. In Linux, "pipes" connect the standard output of one command to the standard input of another command.

Let's take a step back, to the `ls` command. There are plenty of options available with `ls`, but what if the contents of a directory stream by too quickly for you to view them?

Let's view the contents of the `/etc` directory.

```
ls -al /etc
```

How do we take a closer look at the output before it races off the screen?

One answer is to pipe the output to a utility called `less`. Known as a pager, `less`, (like `more`) allows us to view information one page (or screen) at a time.

We use the vertical bar (`|`) to pipe the commands (as shown in Figure 34).

```
ls -al /etc | less
```

Now we can view the contents one screen at a time. To move forward a screen, just press `[Space]`; to move back a screen, press `[B]`; to quit, just press `[Q]`.

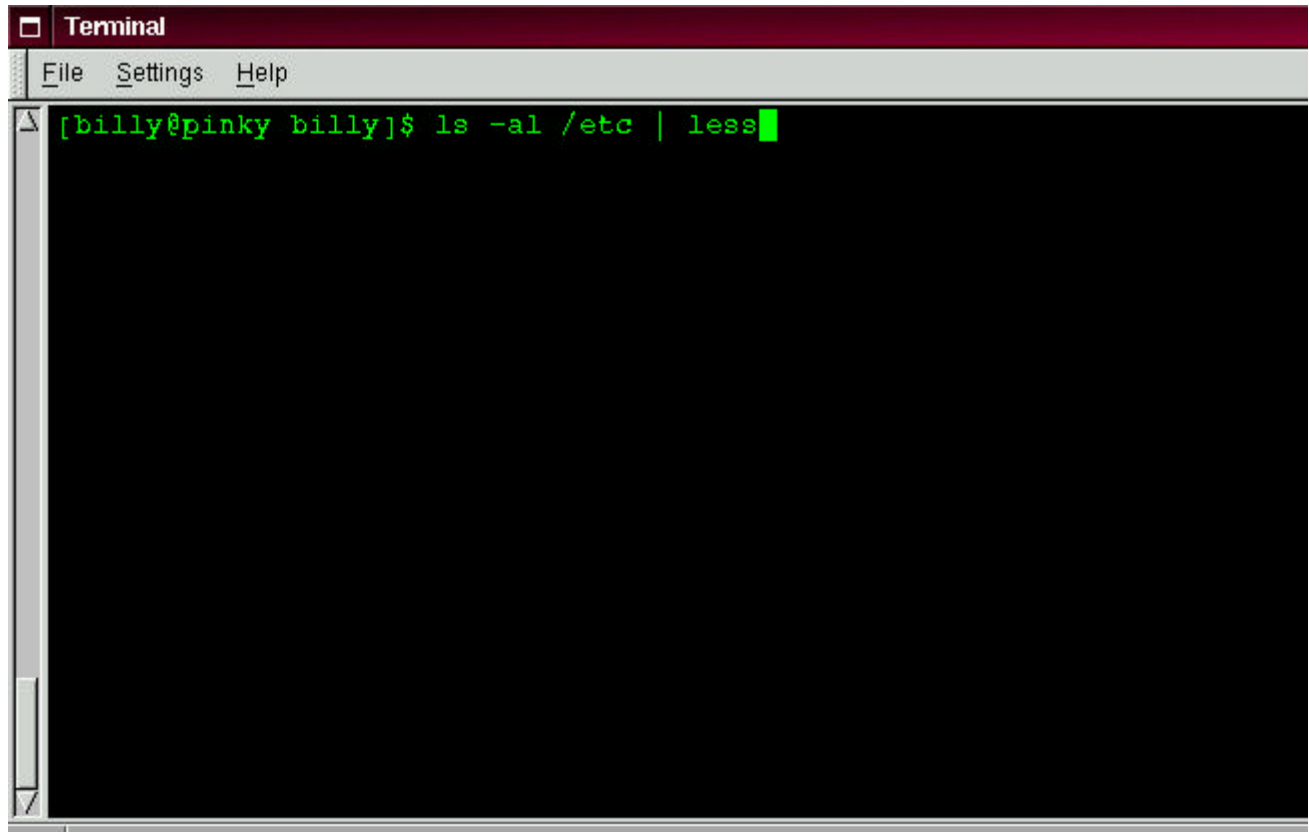


Figure 34: Piping the output of `ls` to the `less` pager

Actually, we've already been using pipes, before we even discussed what they were.

In previous references to `man` pages, we used the following to print out the `man` page entry:

```
man ls | col -b | lpr
```

Here, we're sending the output of `man ls` to a filter called `col` with an option of `-b` to help format the text for the printer, then we sent the output of that to the printer using the `lpr` command.

Summary: Piping allows you to send the output of one command as the input of another command. For example: `ls -al /etc | more` pipes the output of the `ls` command to the `more` utility for easy viewing.

2.11 Stringing Commands Together

Linux allows you to enter multiple commands at one time. The only requirement is that you separate the commands with a semicolon.

Want to see how long you've been online using Netscape? Just combine the `date` command with Netscape's command.

```
date; netscape; date
```

Remember that commands are case sensitive, so the command to start Netscape must be in lower-case to start the browser.

In the `xterm` window, we'll see something like this:

```
[billy@localhost billy] date; netscape; date
Saturday Mar 27 21:26:27 EST 1999
```

We'll see the second date entry when we close out of Netscape. Then, our screen will look like this:

```
[billy@localhost billy] date; netscape; date
Fri Mar 26 13:26:27 EST 1999
Fri Mar 26 14:28:32 EST 1999
[billy@localhost billy]
```

And the prompt will return. The discrepancy between the two results from the `date` command shows that we were using Netscape for just over an hour.

2.12 Ownership and Permissions

Earlier in this chapter, when we tried to `cd` to root's login directory, we received the following friendly message:

```
[billy@localhost billy] cd /root
bash: /root: Permission denied
[billy@localhost billy]
```

That was one demonstration of Linux's security features. Linux, like UNIX, is a multi-user system, and file *permissions* are one way the system uses to protect against any type of tampering -- malicious or accidental.

One way to gain entry when we see we're denied permission is to `su` to root, as we learned earlier. That's because whoever knows the root password has complete access.

```
[billy@localhost billy] su root
Password: (your root password)
[root@localhost billy]# cd /root
[root@localhost /root]#
```

But switching to superuser isn't always convenient -- or smart, since it's so easy to mistakenly mess up important configuration files.

All files and directories are "owned" by the person who created them. We created the file `sneakers.txt` in our login directory, so `sneakers.txt` "belongs" to us.

That means, we can specify who's allowed to read the file, write to the file or, if it were an application instead of a text file, who can execute the file.

Reading, writing and executing are the three main settings in permissions.

Since every user on the system is placed into a *group* when that user is created, then we can also specify whether certain groups can read, write to, or execute our file.

Let's take a closer look at `sneakers.txt` with the `ls` command using the `-l` (long) option (see Figure 35).

```
[billy@localhost billy] ls -l sneakers.txt
-rw-rw-r-- 1 billy billy 150 Mar 19 08:08 sneakers.txt
```

There's quite a bit of detail here. We can see who can read (`r`) and write to (`w`) the file, as well as who created the file (`billy`) and to which group the owner belongs (`billy`).

Tip: Remember that, by default, your group was the login name you chose.


```

Terminal
File Settings Help
[billy@pinky billy]# ls -l
total 1
-rw-rw-r-- 1 billy billy 66 Apr  3 18:16 sneakers.txt
[billy@pinky billy]#

```

Figure 35: Permissions for sneakers.txt

Other information to the right of the group includes the file name, date and time of its creation as well as size.

How do all those dashes and letters fit together? It's not as hard to read as it might seem. Let's take a look:

`-rw-rw-r--`

There are 10 slots in this column. The first slot represents the type of file. The remaining nine slots are actually three sets of permissions for three different categories of users.

Those three sets are: the owner of the file, the group in which the file belongs and "others," meaning users and groups other than owner of the file (billy) and those in billy's group (which is also billy).

Let's stretch out these file settings a bit:

```

-   (-rw)  (-rw)  (r--)   1 billy billy
|       |       |       |
type  owner  group  others

```

The first item, which specifies the file type, can show one of the following:

- `d` -- a directory
- `-` -- a regular file (rather than directory or link)
- `l` -- a symbolic link to another program or file elsewhere on the system

Beyond the first item, in the following three sets, we'll see one of the following:

- `r` -- file can be read
- `w` -- file can be written to
- `x` -- file can be executed (if it's a program)

When we see a dash in owner, group or others, it means that particular permission hasn't been granted.

Let's look again at first column of `sneakers.txt` and identify its permissions. (See Figure 36)

```

[billy@localhost billy] ls -l sneakers.txt
-rw-rw-r-- 1 billy billy 150 Mar 19 08:08 sneakers.txt
[billy@localhost billy]

```

```
total 1
-rw-rw-r--  1 billy  billy
[billy@pinky billy]$
```

Figure 36: A closer view of permissions

The file's owner, `billy`, has permission to read and write to the file; it's not a program, so `billy` doesn't have permission to execute it. The group, `billy`, has permission to read and write to `sneakers.txt`, as well. Similar to the program notation for owner `billy`, there's no execute permission for group `billy`.

In the last set, we can see that those who aren't either the user `billy` or in the group called `billy` can read the file, but can't write to it or execute it.

We can use the `chmod` command to change a file's permissions.

Let's work a bit more on `sneakers.txt` to change the permissions with the `chmod` command.

The original file looks like this, with its initial permissions settings:

```
-rw-rw-r--  1 billy billy      150 Mar 19 08:08 sneakers.txt
```

As long as we're the owner of the file -- or we're logged into the root account -- we can change permissions in any combination of settings for the owner, group and others.

Right now, the owner (that's us) and our group (which is `billy`) can read and write to the file.

Anyone outside of our group -- for example, anyone in the `adm` group - can only read the file (`r--`).

Tip: Remember that file permissions are a security feature. Whenever you allow everyone to read, write to and execute files, you may be increasing your risk of tampering. As a rule, then, you should shy away from allowing everyone to read and write to a file.

In this case, however, let's say that we want to allow everyone to write to the file, so they can read it, write notes in it and save it. That means we'll have to change the change the ``others'' section of the file permissions.

Since we're the owner of the file, we don't have to `su` to root to do it. Let's take a look at the file first.

At the shell prompt, type:

```
ls -l sneakers.txt
```

which gives us this file information:

```
-rw-rw-r--  1 billy billy      150 Mar 19 08:08 sneakers.txt
```

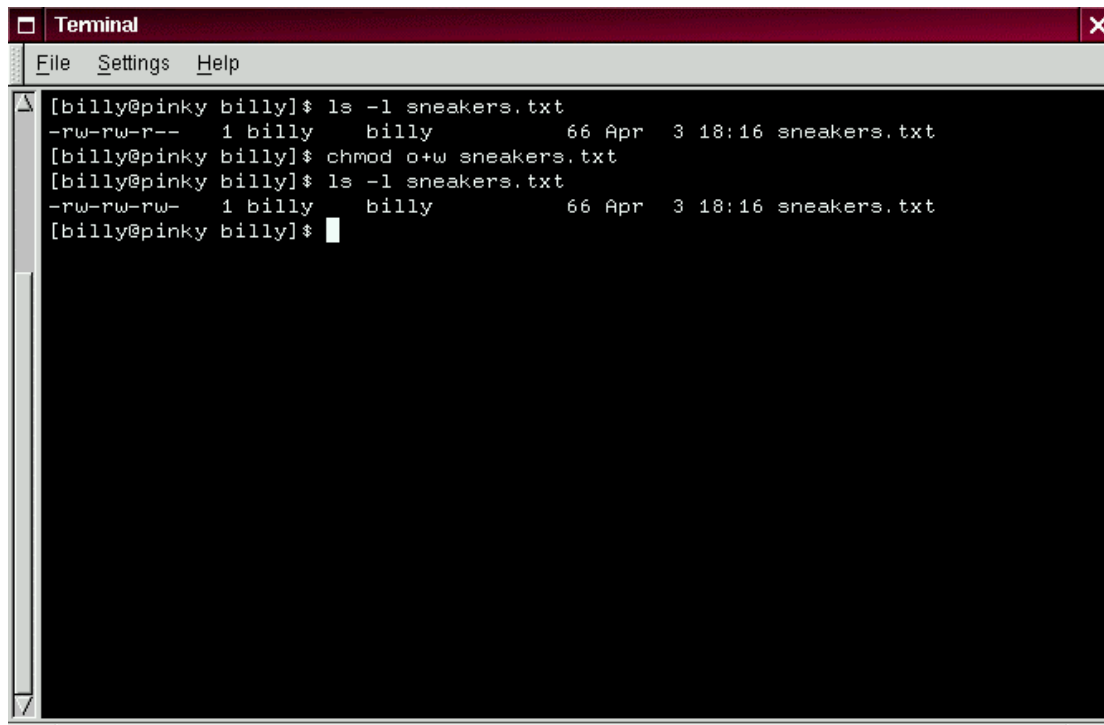
Now, type the following:

```
chmod o+w sneakers.txt
```

To check our results, we can list the file's details again. Now, the file looks like this:

```
-rw-rw-rw-  1 billy billy      150 Mar 19 08:08 sneakers.txt
```

There's our result: Now, everyone can read and write to the file (Figure [37](#)).



```
[billy@pinky billy]$ ls -l sneakers.txt
-rw-rw-r-- 1 billy billy 66 Apr 3 18:16 sneakers.txt
[billy@pinky billy]$ chmod o+w sneakers.txt
[billy@pinky billy]$ ls -l sneakers.txt
-rw-rw-rw- 1 billy billy 66 Apr 3 18:16 sneakers.txt
[billy@pinky billy]$
```

Figure 37: Changing permissions for sneakers.txt

When we typed `o+w`, we were saying, "for others, add write permission to the file sneakers.txt." If we want to remove all access permission from sneakers.txt (even though it's only a sketchy shopping list), we could use the `chmod` command to take away both the read and write permissions like so:

```
chmod go-rw sneakers.txt
```

and the result will look like this:

```
-rw----- 1 billy billy 150 Mar 19 08:08 sneakers.txt
```

By typing `go-rw`, then, we were saying "for the group and others, remove read and write permission to the file sneakers.txt."

You might think of these settings as a kind of shorthand when you want to change permissions with `chmod`, because all you really have to do is remember a few symbols and letters with the `chmod` command.

Here a list of what the shorthand represents:

Identities

- `u` -- the user who owns the file (that is, the owner)
- `g` -- the group to which the user belongs
- `o` -- others (not the owner or the owner's group)
- `a` -- everyone (`u`, `g`, and `o`)

Permissions

- `r` -- read access
- `w` -- write access
- `x` -- execute access

Actions

- `+` -- adds the permission
- `-` -- removes the permission
- `=` -- makes it the only permission

Want to test it out? Let's remove all permission from sneakers.txt -- for everyone.

```
chmod a-rw sneakers.txt
```

Now, let's see if we can read the file:

```
[billy@localhost billy] cat sneakers.txt
```

```
cat: sneakers.txt: Permission denied
[billy@localhost billy]
```

Guess it worked; even we can't get into the file. But since the file belongs to us, we can always change permission to allow us read and write access. (See Figure 38)

```
[billy@localhost billy] chmod u+rw sneakers.txt
[billy@localhost billy] cat sneakers.txt
buy some sneakers
then go to the coffee shop
then buy some coffee
bring the coffee home
take off shoes
put on sneakers
make some coffee
relax!
[billy@localhost billy]
```

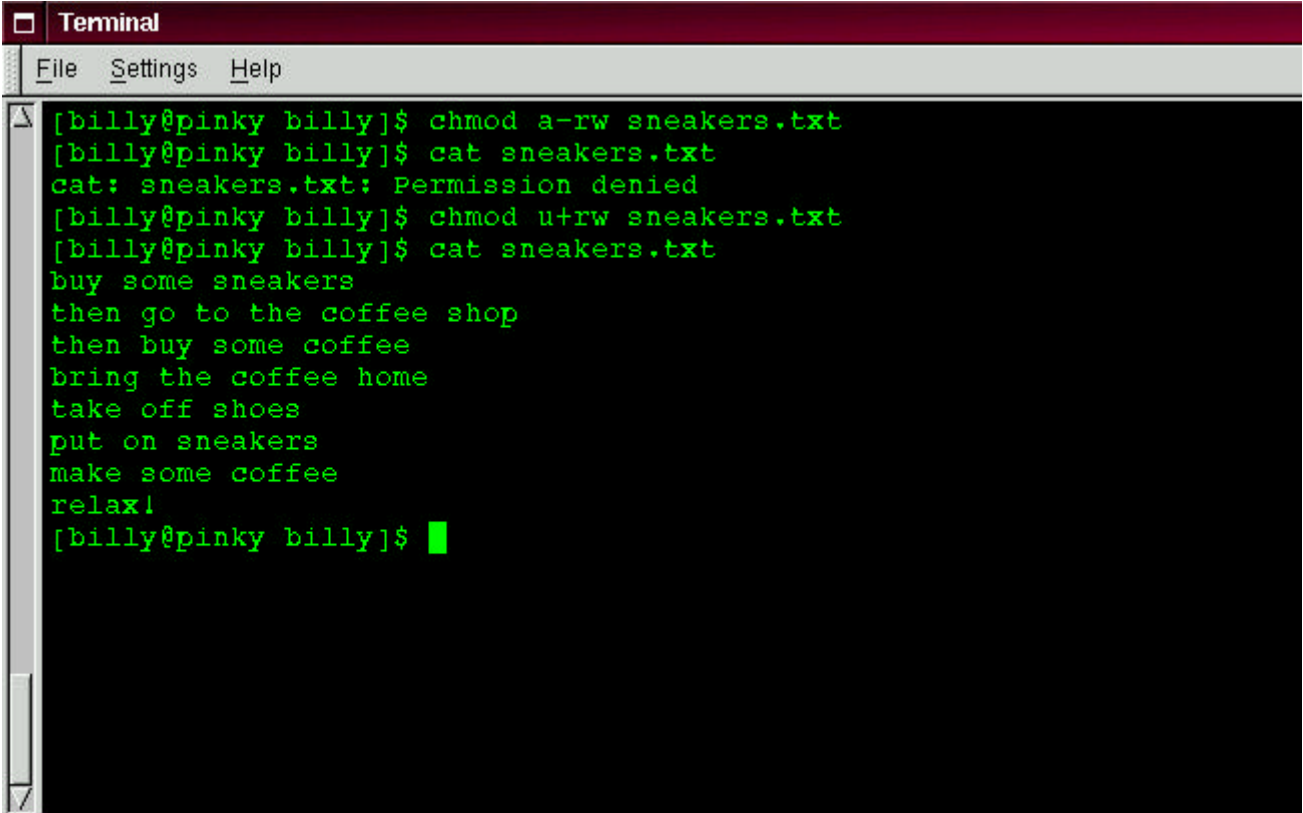
A terminal window titled "Terminal" with a menu bar containing "File", "Settings", and "Help". The terminal shows a user named "billy" on a host named "pinky". The user runs the command `chmod a-rw sneakers.txt`, then `cat sneakers.txt`, which results in a "Permission denied" error. The user then runs `chmod u+rw sneakers.txt`, followed by `cat sneakers.txt`, which successfully displays the contents of the file: "buy some sneakers", "then go to the coffee shop", "then buy some coffee", "bring the coffee home", "take off shoes", "put on sneakers", "make some coffee", and "relax!". The terminal ends with the prompt `[billy@pinky billy]$` and a cursor.

Figure 38: Removing, then restoring permissions

Here are some common examples of settings that can be used with `chmod`:

- `g+w` -- adds write access for the group
- `o-rwx` -- removes all permissions for others
- `u+x` -- allows the file owner to execute the file
- `a+rw` -- allows everyone to read and write to the file
- `ug+r` -- allows the owner and group to read the file
- `g=rx` -- lets the group only read and execute (not write)

By adding the `-R` option, we can change permissions for entire directory trees.

There's a slight twist, however, because we can't really "execute" a directory as we would an application. Instead, when we add or remove execute permission for a directory, we're really allowing (or denying) permission to search through that directory.

To allow everyone read and write access to the `tigger` directory in our login directory, we just type:

```
chmod -R a+rw tigger
```

But... If we don't allow others to have execute permission to tigger, it doesn't matter who has read or write access, because no one will be able to get into the directory -- unless they know the exact filename they want.

For example, let's type:

```
chmod a-x tigger
```

to remove execute access to all.

Here's what happens now when we try to cd to into tigger:

```
[billy@localhost billy] cd tigger
```

```
bash: tigger: Permission denied
```

```
[billy@localhost billy]
```

Let's restore ours and our group's access.

```
chmod ug+x tigger
```

Now, if we check our work with `ls -dl` we'll see that only others will be denied access to tigger.

2.13 Fun with Numbers in `chmod`

Remember when we made a reference to the "shorthand" method of `chmod`? Here's another way to change permissions; it may seem a little complex at first - especially if math isn't your strong suit.

Let's go back to the original permissions for `sneakers.txt`.

```
-rw-rw-r-- 1 billy billy 150 Mar 19 08:08 sneakers.txt
```

Each permission setting can be represented by a numerical value:

- $r = 4$
- $w = 2$
- $x = 1$
- $- = 0$

When these values are added together, the total is used to set specific permissions - more specific than changing permissions with the alphabetical "shorthand."

In `sneakers.txt`, then, here are the numerical permissions settings:

```
- (-rw) (-rw) (r--)  
  |      |      |  
  0+4+2  0+4+2  4+0+0
```

The total for the user is six, the total for the group is six and the total for others is four. The permissions setting, then, is read as 664.

If we want to change `sneakers.txt` so those in our group didn't have write access, but could still read the file (as shown in [Figure 39](#)), we'll have to remove the access by subtracting 2 from that set of numbers.

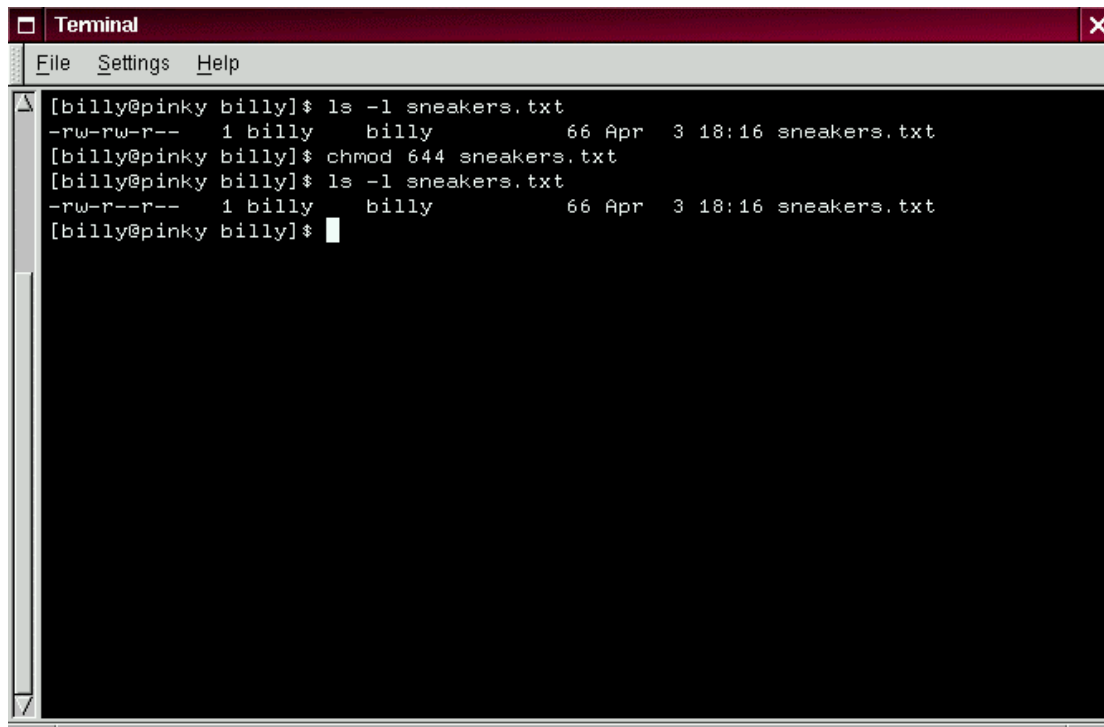
The numerical values, then, would become six, four and four -- or 644.

So we can type:

```
chmod 644 sneakers.txt
```

Let's check our changes by listing the file (`ls -l sneakers.txt`):

```
-rw-r--r-- 1 billy billy 150 Mar 19 08:08 sneakers.txt
```

A terminal window titled "Terminal" with a menu bar containing "File", "Settings", and "Help". The terminal shows the following commands and output:

```
[billy@pinky billy]$ ls -l sneakers.txt
-rw-rw-r-- 1 billy billy 66 Apr 3 18:16 sneakers.txt
[billy@pinky billy]$ chmod 644 sneakers.txt
[billy@pinky billy]$ ls -l sneakers.txt
-rw-r--r-- 1 billy billy 66 Apr 3 18:16 sneakers.txt
[billy@pinky billy]$
```

Figure 39: Removing group write permissions

And there it is; now, neither the group nor others have write permission to `sneakers.txt`. To return the group's write access for the file, we can just add the value of `w` (2) to the second set of permissions.

```
chmod 664 sneakers.txt
```

Tip: Beware 666 and 777. Biblical implications aside, either of these settings will allow everyone to read and write to a file or directory. Such settings as these could allow tampering with sensitive files, so in general, it's not a good idea to allow these settings.

Here's a list of some common settings, numerical values and their meanings:

- `-rw-----` (600) -- Only the user has read and write permissions.
- `-rw-r--r--` (644) -- Only user has read and write permissions; the group and others can read only.
- `-rwx-----` (700) -- Only the user has read, write and execute permissions.
- `-rwxr-xr-x` (755) -- The user has read, write and execute permissions; the group and others can only read and execute.
- `-rwx--x--x` (711) -- The user has read, write and execute permissions; the group and others can only execute.
- `-rw-rw-rw-` (666) -- Everyone can read and write to the file. Bad idea.
- `-rwxrwxrwx` (777) -- Everyone can read, write and execute. Another bad idea.

Here are a couple common settings for directories:

- `drwx-----` (700) -- Only the user can read, write in this directory.
- `drwxr-xr-x` (755) -- Everyone can read the directory, but its contents can only be changed by the user.

Summary: You can change permissions with the `chmod` command by using letters or numbers. Type `chmod (permissions) file` to change permissions of a file or directory.

You've already come quite a distance in learning about your Red Hat Linux system -- from navigation to setting and changing permissions.

Now, it's time to learn a little more about managing what you have on your system. The following chapter will help you to understand a little more about file types and how to work with a variety of commands.

3 Managing Files and Directories

If you're a Linux newbie -- and there are many -- you may feel a little disoriented when you want to accomplish your first tasks.

Relax. If you've had any experience with other operating systems, learning Linux is a bit like learning to drive in a new country: The ideas are the same, but some of the particulars are a bit different. We'll go over several of those "rules of the road" in this chapter.

But there's one component of your new operating system you just can't do without: the *shell*. We've made numerous references to the shell -- as in "shell prompt," or "bash."

Now, it's time to learn a little more about this indispensable tool. But first, a little history...

3.1 Shell Collecting

In the olden days (we're talking '60s here), when AT&T's Dennis Ritchie and Ken Thompson were designing UNIX, they wanted to create a way that humans could interact with their new system. Operating systems at the time did come with "command interpreters," which could take commands from the user and interpret them for the computer to understand.

But Ritchie and Thompson wanted something more, something which could offer better features than the command interpreters of the day.

Enter the Bourne shell (known simply as `sh`), created by S.R. Bourne, which fulfilled the goals of UNIX's creators.

Since the creation of the Bourne shell, other shells have been developed, such as the C shell (`csh`) and the Korn shell (`ksh`).

When the Free Software Foundation sought a royalty-free shell, developers began to work on the language behind the Bourne shell as well as some of the popular features from other shells available at the time.

The result was the Bourne Again Shell -- or `bash`.

By now, of course, you've probably seen the word `bash` when you've mistyped commands at the shell prompt (as in `bash: oops: command not found`).

In Chapter [1](#), when we covered redirection and piping, we were also demonstrating the power of `bash`.

Tip: You can learn more about `bash` by reading the `bash` man page. At the shell prompt, type `man bash` (or you can save the file as a text file by typing `man bash | col -b > bash.txt`, which you can then open to read with an editor like `pico` or a pager like `less`. You can also print the file with `man bash | col -b | lpr`, but be warned: it's a large file. If you want more information, O'Reilly & Associates publishes *Learning the bash Shell*, by Cameron Newham and Bill Rosenblatt.

Although your system came with several different shells, `bash` is the default shell for Red Hat Linux. You might think of `bash` as a fleet-footed office assistant who has made a habit of keeping notes on ways to fulfill commands quickly. This assistant also keeps pointers on how you like to customize the way you work.

These "pointers" `bash` keeps are referred to as *environment variables*.

The shell uses an "environment" in the same way we use an environment, like a kitchen. We work in our kitchen, arrange pots, pans, and spices. We know where the dishes are, how things operate.

The same can be said for `bash` and its environment. There's a basic arrangement to `bash` as there would be to just about any kitchen. For example, we'd expect to see pots in a kitchen in the same way that we would expect to see certain commands in `bash`.

That's the idea behind environment variables.

As long as your assistant has the right pointers, he'll fulfill your commands quickly.

Let's take a look at our environment variables. At the shell prompt, type:

```
env
```

Quite a few "shortcuts" `bash` uses, aren't there?

Each one of these helps `bash` customize the environment for you.

Among the most important environment variables is the `PATH` environment variable -- which defines what is known as the *default path*.

The `PATH` environment variable for our account `billy` might look something like this:

```
PATH=/usr/local/bin:/usr/X11R6/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/billy/bin
```

It looks crowded, but the `PATH` statement is a great signpost, which points to where programs can be found.

Tip: Remember the reference in the previous chapter to the FHS (Filesystem Hierarchy Standard)? The `PATH` statement is set according to that standard, and programs are installed in directories in accordance with the FHS as well. The end result is that the `PATH` statement will enable `bash` to automatically find nearly any program, assuming it has been installed in accordance with the FHS.

3.2 Locating Files and Directories

There will be times when we know a file or directory exists but we won't know where to find it.

Searching for a file or directory can be made much easier with the `locate` command.

With `locate`, we'll see every related file or directory which matches our search criterion. Let's say we want to search for all files related to the `finger` command.

```
locate finger
```

The `locate` command uses a database to check for files and directories which match the string `finger`.

Tip: To learn more about `locate`, read the `locate` man page by typing `man locate` at a shell prompt.

It's a handy command which works very quickly -- as long as the database is up to date. That database is automatically updated on a nightly basis, from `cron`. What's `cron`? It's a small program that runs in the background, performing various tasks -- such as updating the `locate` database -- at regularly scheduled intervals.

Tip: `cron` is a *daemon*. Daemons handle tasks in the background. To read the `cron` man page, type `man cron` at the shell prompt.

So what happens if we:

- Have more than one operating system on our machine, and switch between them -- causing us to halt and restart our Red Hat Linux system;
- Shutdown and turn off our machine at the end of the day.

This might mean that `cron` never has a chance to update the `locate` database. No problem. We can just update the database manually. Let's give it a try.

First, `su` to root.

Now, at the shell prompt, type:

```
/etc/cron.daily/updatedb.cron
```

After a few minutes, the `locate` database will be current.

3.3 Command History and Tab Completion

It doesn't take long before the thought of typing the same command over and over becomes unappealing, at best.

Linux users don't feel any differently about that, either. But in Linux, since you can string together commands at the shell prompt, one minor typo in a couple lines of a command could mean that all that typing was in vain.

So there's a solution: It's called *command-line history*. By scrolling with the up and down arrow keys, we can find plenty of our previously typed commands -- including the ones with typos.

Let's try it by taking a look again at `sneakers.txt`. The first time, however, at the shell prompt, we'll type:

```
cat sneakers.txt
```

Oops! Nothing happens, of course, because there is no `sneakrs.txt` file. No problem. We'll just use the up-arrow key to bring back the command, then use the left-arrow key to get to the point where we missed the ``e`." Insert the letter and press [Enter] again.

Voila! We now see the contents of `sneakers.txt`.

The `bash` shell can store up to 1,000 commands.

Tip: By typing the `env` command at a shell prompt, we can see the environment variable that controls the size of the command-line history. The line which reads, "`HISTSIZE=1000`" tells us that `bash` will store 1,000 commands in its history.

The command-line history is actually kept in a file, called `.bash_history` in our login directory.

We can read it in a number of ways: by using `pico`, `cat`, `less`, `more`, and others.

Be prepared, though: the file can be pretty long.

Let's read it with `more`:

```
more .bash_history
```

To move forward a screen, press [Space]; to move back a screen, press [B]; to quit, press [Q].

Another time-saving tool is known as *tab completion*. If you type part of a file or pathname then hit the [Tab] key, `bash` will present you with either the remaining portion of a the file/path, or a beep. If we get a beep, we can press [Tab] again to obtain a list of the files/paths that match what we've typed so far.

So even if we do turn off the machine at the end of the day, we probably won't have to work too hard in order to remember the command to update locate's database: The chances are good that the command will be stored in the command-line history or can be completed with tab completion (as long as we remember the start of the pathname for the command).

Both tab completion and command-line history can be useful to help you use a command you've forgotten, as well.

Let's try tab completion to update locate's database. First, `su` to root.

Now, at a shell prompt, type the beginning of the path:

```
/etc/cr
```

Now, press the [Tab] key and tab completion will complete the path to `/etc/cron` (and you'll hear a beep). Press [Tab] again, and you'll be presented with a list of possible completions:

```
cron.daily cron.hourly cron.monthly cron.weekly crontab
```

So add the `.daily` to the command (or simply type the `.d` and press [Tab] again). Press [Tab] again, and the result will be:

```
logrotate tetex.cron tmpwatch updatedb.cron
```

And there's our `updatedb.cron` command. We can simply add it to the end of the path we've got so far (or just type `u` and press [Tab] yet again) and press [Enter] to run `updatedb.cron`.

3.4 Identifying and Working with File Types

If you're new to Linux, it won't take long before you begin seeing files with extensions that may seem foreign. A file's extension is the last part of a file's name, after the final dot (in the file `sneakers.txt`, `txt` was that file's extension).

Here's a brief listing of extensions and their meanings:

3.4.1 Compressed/Archived Files

- `.Z` -- a compressed file
- `.tar` -- an archive file (short for *tape archive*)
- `.gz` -- a compressed file (gzipped)
- `.tgz` -- a tarred and gzipped file

3.4.2 File Formats

- `.txt` -- a plain ASCII text file
- `.html/.htm` -- an HTML file
- `.ps` -- a PostScript file; formatted for printing
- `.au` -- an audio file
- `.wav` -- an audio file
- `.xpm` -- an image file
- `.jpg` -- a graphical or image file, such as a photo or artwork
- `.gif` -- a graphical or image file
- `.png` -- a graphical or image file

3.4.3 System Files

- `.rpm` -- a Red Hat Package Manager file
- `.conf` -- a configuration file
- `.a` -- an archive file
- `.lock` -- a "lock" file; determines whether a program is in use

3.4.4 Programming and Scripting Files

- `.h` -- a C and C++ program language header file
- `.c` -- a C program language source code file
- `.cpp` -- a C++ program language source code file
- `.o` -- a program object file
- `.pl` -- a Perl script
- `.tcl` -- a TCL script
- `.so` -- a library file

But file extensions are not always used, or used consistently. So what happens when a file doesn't have an extension, or the file doesn't seem to be what the extension says it's supposed to be?

That's when the `file` command can come in handy.

In [1](#), we created a file called `saturday` -- without an extension. Using the `file` command, we can tell what the file is by typing:

```
file saturday
```

and we'll see it's a text file. Any file that's designated a text file should be readable using `cat`, `more`, or `less`.

Tip: To learn more about the `file` command, read the `file` man page by typing `man file`.

And speaking of reading files...

There are plenty of ways to read files in Linux. In the previous chapter, for example, we covered the pagers `more` and `less` -- they're called pagers because you can "page" through documents one screen at a time. We also learned how we can not only view but manipulate files with the `cat` command.

But there are even more options when it comes time to take a look at README files, man pages or documents you've created.

You have a number of tools to help you read text files, among them, the text editors `pico`, `emacs`, and `vim`, the pagers `more` and `less`, and the viewers `head`, `tail`, `cat`, and `grep`.

Let's take a look at some of the features in these tools.

3.4.4.1 The `less` Command

In [Chapter 1](#), we were introduced to the pager `less`. `Less` is the pager that's used to display man pages. Let's view the man page for `less` to see `less` in action.

```
man less
```

To move forward a screen, press `[Space]`; to move back a screen, press `[B]`, and to quit, press `[Q]`.

There are other powerful features to `less`, as well, including the ability to scroll horizontally and specify the number of lines to scroll.

3.4.4.2 The `more` Command

Odd as it may seem, `more` offers less than `less` (actually, `less` was inspired by `more`). Let's take a look at the man page for `more`, but this time, we'll open the page using `more --` by piping man's output to `more`.

```
man more | more
```

It may not look too different at first, but there are fewer enhancements to `more` than to `less`. Probably the most striking difference at first is the lack of a way to go backwards in a document -- although moving forward by pressing [Space] and quitting by pressing [Q] are the same.

3.4.4.3 The `head` Command

You can use the `head` command if you just want to look at the beginning of a file. The command is:

```
head <filename>
```

`Head` can be useful, but because it's limited to the first several lines, you won't know how long the file actually is. By default, you can only read the first 10 lines of a file, although we can specify the number to see more by typing:

```
head -20 <filename>
```

Read `head`'s man page (`man head`) for more information. You'll probably find that `less` or `more` are more helpful, because you can page through the file if you find that the information you're looking for is further into the file than you originally thought.

3.4.4.4 The `tail` Command

The reverse of `head` (obviously), is `tail`. With (`tail`), you can review the last 10 lines of a file.

3.4.4.5 The `cat` Command

The command `cat`, short for concatenation, will dump the contents of the entire file on the screen. Using `cat` can be handy if the file is fairly short, such as when we created `sneakers.txt`. But if a file is fairly long, it will easily scroll past you on the screen, since `cat` displays the whole file.

3.4.4.6 The `grep` Command

The `grep` command is pretty nifty for finding specific character strings in a file. Let's say we want to find every reference we made to ``coffee'' in the file `sneakers.txt`, which we created in our login directory. We could type:

```
grep coffee sneakers.txt
```

and we would see every line in which the word ``coffee'' could be found.

Tip: Unless otherwise specified, `grep` searches are case sensitive. That means that searching for *Coffee* is different than searching for *coffee*. So among `grep`'s options is `-i`, which allows you to make a case-insensitive search through a file. Read the `grep` man page for more about this command.

3.4.4.7 I/O Redirection and Pipes

And don't forget about using pipes and output redirection when you want to store and/or print information to read at a later time.

You can, for example, use `grep` to search for particular contents of a file, then have those results either saved as a file or sent to a printer.

To print the information about references to ``coffee'' in `sneakers.txt`, for example, we just type:

```
grep coffee sneakers.txt | lpr
```

This command behaves similar to the command `ls -al /etc | more`, which you may have used in Chapter 1 to list the contents of the `/etc` directory then send the results through the `more` command for viewing on the screen.

Tip: Remember the distinction of using `>` and `>>`: using `>` will overwrite a file,

while `>>` appends the information to a file. Usually, unless you're certain you want to, it's safer to use `>>`, because you won't lose potentially valuable information (though you may have to edit the file if you didn't want to append information to it).

3.4.4.8 Wildcards and Regular Expressions

What if you forget the name of the file you're looking for? You can't say to your computer, "Find a file called 'sneak' or 'sneak-something'."

Well, yes you can, in a way. Using *wildcards* or *regular expressions*, you can perform actions on a file or files without knowing the complete filename. Just fill out what you know, then substitute the remainder with a wildcard.

Tip: To read more about wildcards and regular expressions, take a look at the `bash` man page (`man bash`). Remember that you can save the file to a text file by typing `man bash | col -b > bash.txt`. Then, you can open and read the file with `less` or `pico` (`pico bash.txt`). If you want to print the file, be prepared: It's quite long.

We know the file's called "sneak-something.txt", so just type:

```
ls sneak*.txt
```

and there's the name of the file:

```
sneakers.txt
```

You'll probably use the asterisk (*) most frequently when you're searching. The asterisk will search out everything that matches the pattern you're looking for. So even by typing:

```
ls *.txt
```

or:

```
ls sn*
```

you'd find `sneakers.txt` -- except that as time goes on, there will be more text files, and they'll all show up because they match the pattern you're searching for.

It helps, then, to narrow your search as much as possible.

One way to narrow the search might be to use the question mark symbol (?). Like the asterisk, using ? can help locate a file matching a search pattern.

In this case, though, ? is useful for matching a single character -- so if you were searching for `sneaker?.txt`, you'd get `sneakers.txt` as a result -- and/or `sneakerz.txt`, if there were such a filename.

When an asterisk, for example, just happens to be part of a filename, such as might be the case if the file `sneakers.txt` was called `sneak*.txt`, that's when regular expressions can come in handy.

Regular expressions are more complex than the straightforward asterisk or question mark.

Using the backslash (\), you can specify that you don't want to search out *everything* by using the asterisk, but you're instead looking for a file with an asterisk in the name.

If the file is called `sneak*.txt`, then, type:

```
sneak\*.txt
```

Here is a brief list of wildcards and regular expressions:

- * -- Matches all characters
- ? -- Matches one character in a string (such as `sneaker?.txt`)
- * -- Matches the * character
- \? -- Matches the ? character
- \) -- Matches the) character

You can also use wildcards for more than searching: they can come in handy when you want to move and rename files. And regular expressions can help you rename files with characters like * and ? in them.

For more on that, read on.

3.5 Copying, Moving and Renaming Files and Directories

By now, you've learned a little about the structure of the filesystem; and you've learned how to create files and directories.

But just because you know how to create files and directories doesn't mean that you're stuck with the changes you've made. What if you want to rename and/or move files and directories?

Let's start with the copy command.

3.5.1 Copying Files

Like so many Linux features, you have a variety of options from which to choose when you want to manipulate files and directories. You can also use wildcards when you're copying, moving, or deleting files and directories.

Basically, the copy command is not much more complex than typing:

```
cp <source> <destination>
```

so to copy the file `sneakers.txt` to the directory `tigger` in your login directory, just type:

```
cp sneakers.txt tigger
```

Notice that you also used relative pathnames to copy the file. You can use both relative and absolute pathnames with `cp`. Our login directory is the parent of the directory `tigger`; meaning that `tigger` is one directory down from ours.

Read the `cp` man page (`man cp`) for a full list of the options available with `cp`. But among the options you can use with `cp` are:

- `-i` -- interactive. Prompts you to confirm if the file is going to overwrite a file in your destination. This is a handy option because it can help prevent you from making mistakes.
- `-r` -- recursive. Rather than just copying all the files and directories, copies the whole directory tree, subdirectories and all, to another location.
- `-f` -- force. Copies without prompting you for confirmation that the file should be overwritten. Unless you're sure you want to force the copy, you probably don't want to make friends with this option right now.
- `-v` - verbose. Will show the progress of the files being copied.

Just by using `cp` alone, you won't see much when the command is executed. Using an option, such as `-i`, can make the process a little more useful, because if you want to copy a file to a location that already has a file with the same name, you'll be asked first if you really want to overwrite -- meaning replace -- the file that's already there.

Tip: Remember that among your options is `-f` (force), which can overwrite files without asking you if you're certain. Make sure, when you use the force option, that you *really* want to overwrite a file.

Now that we have the file `sneakers.txt` in the `tigger` directory, let's use `cp -i` to copy the file again to the same location.

```
[billy@localhost billy] cp -i sneakers.txt tigger
```

```
cp: overwrite 'tigger/sneakers.txt'?
```

If we want to overwrite the file that's already there, we can press `[Y]` and then `[Enter]`. If we think we don't want to overwrite the file, now's the time to press `[N]` and `[Enter]`.

3.5.2 Moving Files

To move files, use the `mv` command (`man mv`), which is similar to the `cp` command, except that with `mv` the file is physically moved from one place to another, instead of being duplicated, as with `cp`.

Common options available with `mv` include:

- `-i` -- interactive. Will prompt you if the file you've selected will overwrite an existing file in the destination directory. This is a good option, because like the `-i` option in `cp`, you'll be given the chance to make sure you want to replace an existing file.

- `-f -- force`. Overrides the interactive mode and moves without prompting. Unless you know what you're doing, this option doesn't play nice; be very careful about using it until you become more comfortable with your system.
- `-v -- verbose`. Shows a list of the files being moved.

If you want to move a file out of your home directory and into another directory, you would type:

```
mv sneakers.txt tigger
or, mv sneakers.txt /home/billy /home/billy/tigger using absolute pathnames.
```

3.5.3 Renaming Files

Actually, we've already covered half of renaming, because when you copy or move files, you can also rename.

To copy the file `sneakers.txt` from our login directory to our `tigger` subdirectory, just type:

```
cp sneakers.txt tigger
```

To copy and rename that file from `sneakers.txt` to `piglet.txt`, type:

```
cp sneakers.txt tigger/piglet.txt
```

To *move* and rename the file, just substitute `mv` for `cp` in the above example.

If you `cd` to `tigger` and use `ls`, you'll see the file `piglet.txt`.

If you just want to rename the file and keep its location, just `mv` in your current directory:

```
mv sneakers.txt piglet.txt
```

3.5.4 Deleting Files and Directories

We talked about creating files with the `touch` command and by using redirection in Chapter [2](#). And we created the directory `tigger` using `mkdir`.

But we haven't discussed how to delete files and directories.

Deleting files and directories with the `rm` command (`man rm`) is a straightforward process.

Let's take our new file `piglet.txt`, and delete it from the `tigger` directory with the `rm` command:

```
rm piglet.txt
```

What happens if we didn't really want to get rid of it? Too late! Again, that's where the `-i` (interactive) option comes in handy, because with it, we have the chance to think about whether we really want to toss the file.

```
[billy@localhost billy] rm -i piglet.txt
```

```
rm: remove 'piglet.txt'?
```

You can also delete files using the wildcard `*`, but be careful, because you can easily delete files you didn't intend to throw away.

To remove a file using a wildcard, you would type:

```
rm pig*
```

You can also remove more than one file in one command, as in:

```
rm piglet.txt sneakers.txt
```

Options for removing files `--` and directories `--` include:

- `-i -- interactive`. Prompts you to confirm the deletion. This is good.
- `-f -- force`. Overrides interactive mode and removes the file(s) without prompting. This might not be good, unless you know exactly what you're doing.
- `-v -- verbose`. Shows a listing of files as they're being removed.
- `-r -- recursive`. When removing directories, will remove all of the files and the subdirectories of the specified directory. This can also get rid of an empty directory.

To remove directories with `rm`, you must specify the `-r` option.

For example, if you want to recursively remove the directory `tigger` you would type:

```
rm -r tigger
```

And if you want to combine options, such as forcing a recursive deletion, you can type:

```
rm -rf tigger
```

Tip: *Be careful!* `rm` is a powerful command, and can delete your entire system! If you're root and you type the simple command `rm -rf /` you're sunk -- like a snake eating its tail, the command will recursively remove everything on your system.

The safer alternative to using `rm` for removing directories is the `rmdir` command. With this command, you won't be allowed to use recursive deletions, so a directory which has files in it won't be deleted.

Read the `rmdir` man page by typing `man rmdir` to find out more about the command.

3.5.5 Time to Learn More

So far, you've learned to become familiar with how to create accounts, using passwords, your filesystem and more.

But what we've covered is just the tip of the iceberg. There are books, websites, and newsgroups (Linux users are second to none when it comes to helping new users) to help you find out more about your new system.

You can begin with the next chapter, in which you'll find pointers to other documentation, as well as additional post-installation configuration.

4 What Do I Do Now?

Now that you're becoming a little more familiar with your Red Hat Linux system, you might be wondering, "What do I do now?"

If so, this chapter is for you. We'll start with some basic places you can go to find more documentation and help from the Linux user community. Then we'll show you how to do some post-installation configuration so you can set up things just the way *you* want.

But before we do any of that, let's talk about documentation...

4.2 The X Window System

While there are people that will use the character-cell interface present when you first log in, many people prefer a graphically-oriented user interface. For Linux systems, the graphical user interface of choice is the X Window System.

In order to run X, you need to have the necessary packages installed. If you selected the "X Window System" component to be installed when you originally installed Red Hat Linux, everything should be ready to go. In that case, please refer to section [4.2.2](#).

4.2.1 If You Haven't Installed X

If you didn't select the "X Window System" component when you installed Red Hat Linux, your Red Hat Linux system won't have the necessary software installed. While it is possible to manually install the required packages, you'll probably find it easier to re-do the installation, particularly if you're new to Linux. Another possibility is to perform an upgrade of the software and select the X Window System components that you need from the package selection installation process.

4.2.1.1 XFree86 Configuration

There are three methods for configuring XFree86 on your machine:

- `Xconfigurator`
- `xf86config`
- by hand

`Xconfigurator` and `xf86config` are functional equivalents and should work equally well. If you are unsure of anything in this process, a good source of additional documentation is:

<http://www.xfree86.org>

`Xconfigurator` is a full-screen menu-driven program that walks you through setting up your X server. `xf86config` is a line-oriented program distributed with XFree86. It isn't as easy to use as `Xconfigurator`, but it is included for completeness. If these utilities fail to provide a working `XF86Config` file, you may have an unsupported card or you may need to write the config file by hand. Usually the former is the case, so check and make sure your card is supported before attempting to write the config file yourself. If your card is not supported by XFree86 you may wish to consider using a commercial X server. If you have questions about whether or not your video card is supported you can check out <http://www.xfree86.org> for information on XFree86.

The X Server

Provided you selected the proper video card at install time, you should have the proper X server installed. When later running `Xconfigurator` or `xf86config`, you need to make sure you select the same video card or the autoprobe will fail.

If you think you installed the wrong X server for your video card, you will have to install the correct one before it can be configured. For instance, if the CD is mounted on `/mnt/cdrom`, and you need to install the S3 server, enter the following commands:

```
cd /mnt/cdrom/RedHat/RPMS
rpm -ivh XFree86-S3-3.1.2-1.i386.rpm
```

```
ln -sf ../../usr/X11R6/bin/XF86s3 /etc/X11/X
```

This will install the S3 server and make the proper symbolic link.

Xconfigurator

To configure the X Window System you must first select your video card. Scroll down the list of supported cards until you locate the card in your machine. Figure 1 may help you determine the video server that matches your hardware. If your card is not listed it may not be supported by XFree86. In this case you can try the last card entry on the list (Unlisted Card) or a commercial X Windows server. The next step is to select your monitor. If your monitor is not listed you can select one of the generic monitor entries or "Custom" and enter your own parameters. Custom monitor configuration is recommended only for those who have a sound understanding of the inner workings of CRT displays. The average user should probably use one of the generic selections from the list. After selecting a monitor you need to tell Xconfigurator how much video memory you have. Move the highlight to the appropriate list entry and then press [Enter] or [F12] to continue. For the next step it is recommended that you select the default (No Clockchip Setting) entry, but experienced users may want to select a specific clockchip.

Selecting your Server

If you are unsure what chipset you have, the best way to find out is usually to look at the card. Figure 1 lists which chipsets and boards require which servers. Pick the one that best matches your hardware.

Server	Chipset
8514	IBM 8514/A Boards and true clones
AGX	All XGA graphics boards
I128	#9 Imagine 128 (including Series II) boards
Mach32	ATI boards using the Mach32 chipset
Mach64	ATI boards using the Mach64 chipset
Mach8	ATI boards using the Mach8 chipset
Mono	VGA boards in monochrome
P9000	Diamond Viper (but not the 9100) and Others
S3	#9 Boards, most Diamonds, some Orchids, Others
S3V	Boards using the S3 ViRGE (including DX, GX, VX) chipset
SVGA	Trident 8900 & 9400, Cirrus Logic, C & T, ET4000, S3 ViRGE, Others
VGA16	All VGA boards (16 color only)
W32	All ET4000/W32 cards, but not standard ET4000's

Figure 40: XFree86 X Servers

Finishing Up

If later you want to increase your refresh rate for your monitor, you can edit the config file by hand or you can run Xconfigurator again and pick a monitor from our list that more closely matches the specs of your monitor.

The final configuration step consists of selecting the video modes that you want to include in your XF86Config file. Use the arrow keys to move the cursor up and down the list under each color depth (8, 16 and 24 bit). Use the [Spacebar] to select individual resolutions and the [Tab] key to move between color depth fields. When you have selected the video modes you want to use move the cursor to the "OK" button and press [Enter], or use the [F12] shortcut. An information screen will give you the most current information on selecting video modes, starting and stopping the X server.

4.2.2 If You've Already Installed X

If you selected the "X Window System" component when you installed Red Hat Linux, but didn't choose to start X automatically when the system boots, you're all set. All you'll need to do is to get X running. As it turns out, there are two ways to do this. You can:

- Start X manually after you log in.
- Start X automatically whenever the system boots.

Let's start with the manual procedure.

4.2.2.1 Starting X Manually

Red Hat Linux, during the installation, gives you the option of starting X automatically. If you didn't choose this option, you'll see the character-cell login prompt you saw when you first booted your Red Hat Linux system.

In order to get X started, you'll first need to log in. Do so (using your non-root account), and then enter the `startx` command. The screen should go blank, and (after a short delay) you should see a graphical desktop with one or more windows. The appearance of the desktop you'll see will vary, depending on the packages you installed and other variables. (See [Chapter 1](#) for more info.)

4.2.2.2 Starting X Automatically

Please Note: Make sure you verify that your X configuration works properly before making X start automatically. Failure to do so can make it difficult to log into your Red Hat Linux system. If you haven't done so already, review the previous section before continuing.

It is possible to configure your Red Hat Linux system such that X will start automatically whenever the system is booted. When configured in this manner, `xm` will run, which will present a graphically-oriented login screen. After logging in, you will have a regular X session running, just as if you had issued a `startx` command manually.

Here's a quick overview of how it's done:

- Test `xm` using `telinit`.
- Edit `/etc/inittab`.
- Reboot.

Let's look at each step in more detail.

Testing `xm` Using `telinit`

-- The `telinit` command is used to change your Red Hat Linux system's "run level." It is the run level that controls various aspects of system operation, including whether `xm` should be started or not. Since `xm` is started at run level 5, you'll need to issue the command:

```
/sbin/telinit 5
```

Please Note: You will need to be logged in as root in order to use `telinit`. Also note that you should *not* be running anything else on your Red Hat Linux system when you change run levels, as any running programs may be killed by the run level change.

If everything is configured properly, after a short delay you should see an `xm` login screen. Log in, verifying that an X desktop appears. Then log out to make sure that `xm` reappears. If it does, your system is configured properly to automatically start X. If there are problems, you can go back to run level 3 using `telinit` (ie, "`/sbin/telinit 3`"), or by rebooting.

Editing `/etc/inittab`

-- The file `/etc/inittab` is used to, among other things, determine the system's default run level. We need to change the default run level from 3 to 5; therefore, we'll need to edit `/etc/inittab`. Using the text editor of your choice, change this line in `/etc/inittab`:

```
id:3:initdefault:
```

When you're done, it should look like this:

```
id:5:initdefault:
```

Please Note: Make sure you change *only* the number 3 to be 5! Do not change anything else, otherwise your Red Hat Linux system may not boot at all! When you've made the change, exit the editor, and use this command to review your handiwork:

```
less /etc/inittab
```

(Press the [Space] to page through the file; [Q] will exit.) If everything looks OK, it's time to reboot. Use the shutdown command to properly shut down your system, and you're done!

4.2.2.3 Exiting X

When you're done, and you'd like to leave X, select the **GNOME foot** on the panel bar, choose **Log out** and answer **Yes** to confirm your decision. You will then be logged out of your system. **Please Note:** If you're running GNOME as your desktop environment, all programs that were currently running will be restarted when you log back in.

4.2.2.4 Changing Your Desktop

You can use the **Switchdesk** feature to change out your desktop environment. **Switchdesk** will present a screen which allows you to switch between any desktop environments that you may have installed on your system. You will then be asked to exit and restart X. You will see your new desktop of choice after X has restarted.

To use the **Switchdesk** feature you can type `switchdesk` at the command line of an Xterm. If you are using GNOME, click on the **GNOME foot**, choose **Run Programs** and type `switchdesk`.

4.2.2.5 Virtual Consoles and X

Note that even if you're running X, you still have access to the regular character-cell user interface. That's because Red Hat Linux uses *virtual consoles* while X is running. To switch to a virtual console, press `[Ctrl]-[Alt]-[Fn]`, where `[Fn]` is any one of the first six function keys. When switching virtual consoles, you should see a standard login prompt; at this point you can login and use the system normally on any (or all) of the virtual consoles.

When you'd like to go back to your X session, simply press `[Ctrl]-[Alt]-[F7]`.

Please Note: Some people remap keys under X; if you do this, be aware that your X keyboard mappings will only be active when in X. This can be confusing if, for example, you've swapped the `[Ctrl]` and `[Caps Lock]` keys under X, as you will have to use two different keystrokes to switch between X and non-X virtual consoles.

4.2.2.6 Handy X-Based Tools

There are several tools that can make life easier for the new Red Hat Linux user. They perform tasks that either require root access, or can only be done by memorizing arcane commands. They all require X to run, so you'll need to get that set up first. These tools are:

- **User Information Tool** -- Makes it easy to update your ``gecos," or basic account information. Run `/usr/bin/userinfo` to start it.
- **User Password Tool** -- Changing passwords is simple with this tool. It's started by running `/usr/bin/userpasswd`
- **Filesystem Mounting Tool** -- Makes mounting and unmounting filesystems simple. Every user-mountable filesystem must have the `user` option present in `/etc/fstab` (see the `mount` man page for more information on the `user` option). Run `(1) /usr/bin/usermount` to start it.
- **Network Device Tool** -- Starting and stopping network interfaces becomes a point-and-click operation with this tool. Run `/usr/bin/usernet` to start this tool. Requires that every interface to be controlled by `usernet` is configured to be ``user-controllable." This can be done by using `netcfg`, and selecting the interface's **Allow any user to (de)activate interface** checkbox.

4.3 Configuring Your Red Hat Linux System For Sound

By default, the only sound you'll hear out of your newly installed Red Hat Linux system is the ordinary, boring, default beep. If your computer system has sound hardware, chances are you can make it work under Red Hat Linux. In some cases successfully getting sound support to work requires a kernel rebuild. However, most of the time it's possible to use the modular sound drivers.

4.3.1 Modular Sound Drivers

Red Hat Linux 6.0 includes the standard OSS/Free sound drivers. This makes it possible to load and unload the various sound drivers without recompiling the kernel or rebooting.

For additional information, please consult the README files in the `rhsound` documentation directory (`/usr/doc/rhsound*`), and also the files in the kernel documentation directory (`/usr/doc/kernel-doc-*/sound`).

There is a mailing list associated with the modular sound drivers (`sound-list@redhat.com`). To subscribe, send mail to `sound-list-request@redhat.com`, with `subscribe` as the subject line.

4.3.1.1 Recognized Sound Cards

At this point, most sound cards should be recognized by the modular sound drivers; however, drivers for the following sound cards were among the first to be developed, and as such, have received the most testing:

- Sound Blaster 1.0
- Sound Blaster 2.0
- Sound Blaster Pro
- Sound Blaster 16
- Sound Blaster 16 PnP
- Sound Blaster AWE32/AWE64

4.3.2 Sound Card Configuration Tool

Also included in Red Hat Linux 6.0 is `sndconfig`, a screen-oriented utility that can properly configure modular sound card drivers.

There are a few things that you should know about `sndconfig`:

Plug and Play Aware -- `sndconfig` is able to detect and automatically configure Plug and Play sound cards such as the Sound Blaster 16 PnP. The configuration information is stored in the `/etc/isapnp.conf` file, along with the configuration information for any other Plug and Play devices. In order to ensure that no configuration will be lost, `sndconfig` saves your original `/etc/isapnp.conf` file as `/etc/isapnp.conf.bak`.

Modifies `/etc/conf.modules` -- `sndconfig` modifies the module configuration file (2) `/etc/conf.modules` by adding information about the module options required for your sound card. Note that `sndconfig` saves your original `/etc/conf.modules` file as (3) `/etc/conf.modules.bak`.

To set up your sound card, run `/usr/sbin/sndconfig`. Note that you must be root in order to run `sndconfig`. If your system contains a Plug and Play sound card, `sndconfig` will identify it, and configure it appropriately.

If you do not want `sndconfig` to probe for Plug and Play sound cards, run `sndconfig` with the `--noprobe` option. It is also possible to manually specify the settings for your sound card; to do so, run `sndconfig` with the `--noautoconfig` option.

If `sndconfig` cannot automatically identify your system's sound card (or you ran `sndconfig` with the `--noprobe` option), you'll be asked to select the type of sound card you have (See Figure 41).

Use the `[]` and `[]` keys to scroll through the different cards listed, and position the highlight on the entry that matches your system's sound card.

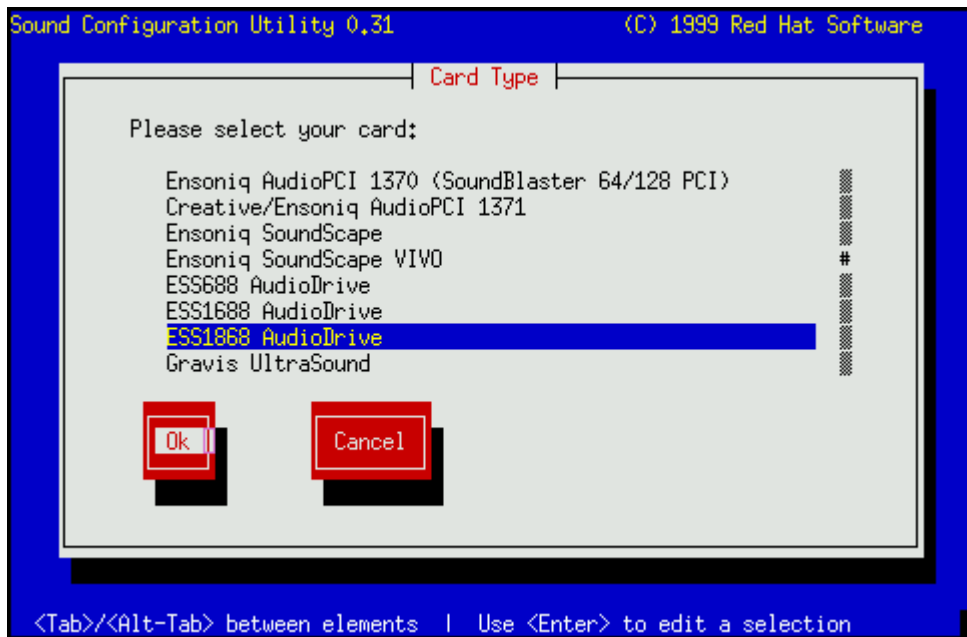


Figure 41: Selecting Sound Card Type

If you've run `sndconfig` with the `--noautoconfig` option, you'll see a screen similar to the one in Figure 42. Here is where you can specify the settings for your sound card. Using the [Tab] key, select a field. Then use the arrow keys to select the desired setting for that field. When finished, select **Ok**, and press [Space].

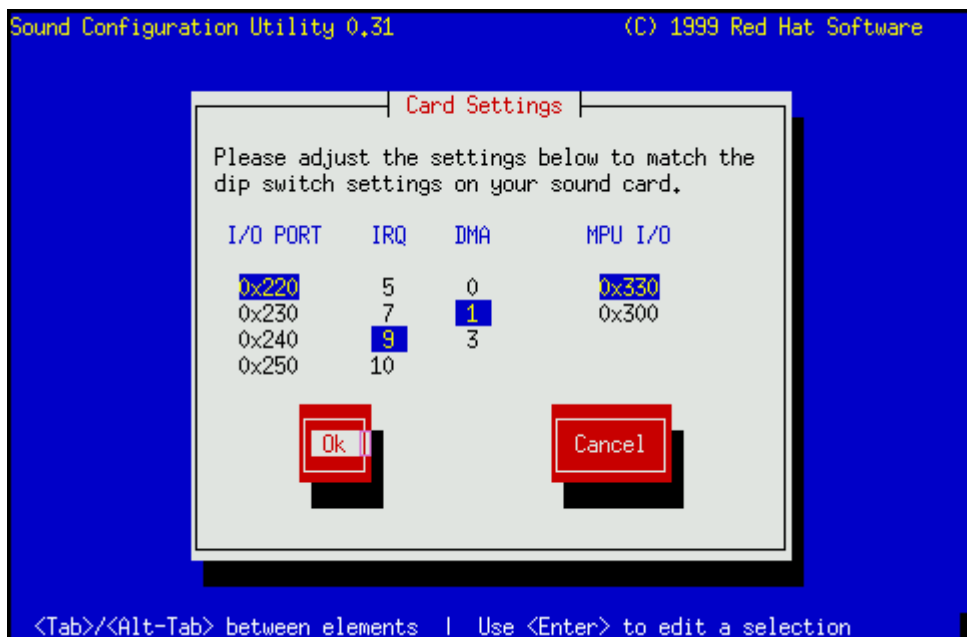


Figure 42: Configuring Sound Card

After this screen, you may see an informational dialog box saying that `/etc/conf.modules` already exists. select **Ok** and press [Space] to continue.

Finally, `sndconfig` will attempt to play a sound sample to verify proper configuration of your sound card. If you can hear the sound sample (make sure the speaker volume is turned up), you're done!

Please Note: On cards that have a recognized MIDI synthesizer, `sndconfig` will attempt to play a MIDI sample as well.

4.4 World Wide Web

The World Wide Web is one of the hottest aspects of the Internet today. Red Hat Linux lets you get in on the action in two ways -- as a Web browser, and as a Web server. Let's look at both.

4.4.1 World Wide Web Browsers

A variety of Web browsers are available for Linux, including freely distributable browsers such as Arena, Lynx, and Grail. The most popular commercial browsers are those from Netscape Communications Corporation. And now they're available with Red Hat Linux 6.0! If you selected the `netscape-communicator` or `netscape-navigator` packages, you're ready to surf. Enjoy!

4.4.2 World Wide Web Server

If you installed the Apache Web server (from the `apache` package), then your Web service is already up and running! Just point your Web browser at <http://localhost>.

The default page shown is `/home/httpd/html/index.html`. You can edit this file (or completely replace it) to your liking. All the CGI programs, icons, and html pages are stored in `/home/httpd`, but this can be changed in the apache configuration files, all of which are stored in `/etc/httpd/conf/`. Logs of all httpd activity are kept in `/var/log/httpd/`. Setting up your Web site is as easy as adding your own HTML pages and links to the `/home/httpd/html/` directory. For more information on customizing your Web server we recommend a reference such as *HTML: The Definitive Guide* by Chuck Musciano & Bill Kennedy, published by O'Reilly & Associates.

4.5 Good luck and enjoy!

As we said before, there is a wealth of information available which can help you get the most out of your your new Red Hat Linux system. We hope this guide has assisted you in your journey.