Hour 3: Linux Basics

Now that you’ve installed Red Hat Linux, you might wonder what to do next. Whether you’re the kind of person who learns by jumping right in and starting to play or if you prefer to have some structure when exploring a new subject, after this hour you will have some of the most basic knowledge you need to get around your new Linux system. This chapter assumes no knowledge of Linux, so if you are already familiar with Linux or UNIX, feel free to go on to Hour 4, "Exploring the Red Hat Linux Filesystem."

In this hour you will

- Make a paradigm shift into the Linux way of thinking
- Learn what happens when you start Linux
- Log in to Linux using runlevel 3 or runlevel 5, and learn why you should not log in as root to do ordinary work
- Study the structure of the Unix command and learn some of the most basic Unix commands
- Get help with the man and info pages
- Shut down Linux properly

Most topics explored this hour pertain to all flavors of Unix-like operating systems, including Linux and BSD. When that is the case we refer to the system as "Unix" (Title case), to encompass all flavors of *nix.

The Paradigm Shift

Many people are scared away from Linux because they think it’s difficult to use. Linux is only difficult if you don’t understand that with Linux comes power and freedom. You have more power to do things the way you want to, and you have more freedom to do tasks in different ways. With power and freedom comes responsibility to learn a little bit more about the reasons why you do things. In the end, a whole universe can open up to you, and using your computer will be a lot more fun and hassle-free than when you were forced to use only Windows.

When you use Linux, you communicate with your computer in a different way from when you use Windows or Macintosh. In Windows, 99% or more of the commands you give and the messages you receive are done via graphical user interface, or GUI. You point on a button with your mouse, click it, and something happens. It is possible to open an MS-DOS prompt and give commands the good old-fashioned way in DOS, but you are strictly limited in what you can accomplish. For almost every task in Windows, you must use the graphical interface.

In Linux, this simply is not the case. There is a graphical user interface in Linux called X, and it looks
a lot like Windows. You point your mouse to a button or menu item and click to give a command. An example of X running the GNOME desktop environment is shown in Figure 3.1.

The difference with Linux is that for every task you perform with a mouse within X, *there is a direct and usually more powerful way to do the same task by typing a command on the keyboard.* When you type a command on the keyboard, you are usually in a *shell environment* (or *shell*, for short), and the place where you type the command is called the *command line*. The shell environment is pictured in Figure 3.2.

**Figure 3.1**
The X interface works on top of the operating system. It is not integrated into the operating system, like the interface is in Windows.

**Figure 3.2**
Don’t be afraid of the shell! It doesn’t look as friendly as X, but it’s not hard to learn.

You can use Linux quite successfully without ever using the shell, but it is difficult, if not impossible, to advance beyond basic proficiency if you only use the graphical interface. To explore the true power of Linux, you must know how to use the command line. For this reason, we will start our exploration of Linux in a shell, as if X didn’t even exist.

If this prospect makes you a bit nervous, maybe this analogy will help. Imagine that you are carrying on a conversation with a friend, but the only way you can communicate is by holding up little pictures to each other. Your conversation would be quite limited, wouldn’t it? Now imagine that you and your friend were having a conversation using spoken language (or sign language!). Think of how much more you can communicate with words than with a limited set of pictures. This analogy is kind of rough, but giving commands to your Linux computer using the command line is much like carrying on a conversation. There are different words you can use to communicate the same command, and there are different options you can attach to your command that mean different things, much like the tone and inflection of your voice communicates different ideas when you say the same word.

**Starting Up**

When you start Linux, it immediately looks different from when you start Windows. When you start Windows, a picture of a blue sky appears with the Windows logo on it. The logo stays there for a few seconds, and then suddenly the blue sky disappears and the Windows desktop appears. You have no idea what’s happening behind that blue sky. For all you know, a little man could be running around collecting bytes and putting them in boxes to start Windows. If the blue sky suddenly disappears and is replaced by the Blue Screen of Death, how do you know what went wrong? How do you fix it? Well, you reboot. What happens if you can’t fix the problem? Well, you reinstall Windows. What a frustrating way to use your computer!

When you start Linux, an entirely different thing happens. Words and phrases appear on the screen. Linux tells you exactly what it is doing, and whether the particular process it was loading was successful or not. At first, all those messages might tell you about as much as the Windows picture of the blue sky. Eventually, however, you will be able to understand and use those messages to help you track down and solve problems, or to make your system run faster and more efficiently.

**A Quick and Dirty Explanation of the Boot Sequence**
When you boot your system in Linux, this is what happens:

Every computer has a chip called the BIOS (for Basic Input and Output). When you turn on your computer, the BIOS wakes up and looks for a program called the boot loader. As you recall from Hour 2, LILO is the Linux boot loader. The boot loader is copied into memory, and then it starts the operating system. The LILO boot loader is special because it can boot not just Linux, but also other operating systems that are installed on your computer.

If you specified that you would start Linux from a boot disk when you installed, then the original (Windows) boot loader is not changed. If your BIOS is set up the way we recommended in Hour 2, it has instructions to look for a boot loader first on floppy, then on CD-ROM, and finally on the hard drive. With a boot disk, the BIOS will load the boot loader from the boot disk into memory, and Linux is started. If no floppy disks or CD-ROMs are found, the normal boot loader on the hard drive is loaded into memory.

After the boot loader is started, it looks for the kernel of the operating system. Once the kernel is found, it is loaded into memory. The kernel’s first job is to find all your hardware and to prepare it for use. It then starts the system processes, such as logging, administration services, email, and network services. Finally, the kernel starts the login program (called getty) so that you can log in and use your system.

The entire boot process is logged in the /var/log/boot.log file. Just skimming this file can help you get an idea of what happens during startup (also called system initialization).

Logging In

Before you can use Linux, you must enter a user name and password to log in, even if you are the only user. The reason for this is that Linux (like other Unices), is designed to be a multiuser system. Windows/MS-DOS was designed to be a single-user system (except for Windows NT).

A simple analogy to explain the difference between single-user and multi-user systems is the difference between an apartment building and a single-family house. In the house there is usually one front door, and one family lives in the building. In Windows you turn on the computer and start using it, just as to enter a house you unlock the front door and walk inside. In Linux there can be many users, just as in an apartment building there are many people living in the same building. There is a front door that you walk through, but there is also the door to your individual apartment, to which only you have access. The individual apartments are like different user accounts on a Linux system.

There are two ways to start the login screen in Linux. One way, called runlevel 5, starts the X graphical interface automatically when you start up Linux. A graphical login box appears in a
window on the screen (just like in Windows NT). When you enter your username and password, the
desktop appears, just like in Windows or Macintosh.

When you installed Red Hat 7.0, you were asked if you wanted X to start automatically
whenever you booted your computer. If you answered Yes, then your Linux system starts
in runlevel 5.

Another way to start Linux is the non-graphical way called runlevel 3. At runlevel 3, when startup is
complete, the login: line appears on a blank screen (some distributions display a simple text-based
graphic as well). After you enter your user name and password, a new line appears with nothing but
your username, the name of your computer, and a $ sign, like this:

judith@localhost$

Rather than being in a graphical environment, you are in a shell environment. The $ sign is called the
shell prompt, or sometimes the command line. There are different shell environments, such as bash,
csh, ksh, sh, and zsh, but Red Hat Linux uses bash by default. In fact, most Linux users use bash.

You can start a graphical environment from the command line, but we won’t talk about
that until Hour 6, "XFree86—The Linux Window System."

For the rest of this hour and for Hours 4 and 5, we will assume that you are in a shell environment. If
your Linux system starts at runlevel 5 (in a graphical environment), there are two ways to drop into a
shell:

- You can press Ctrl+Alt+F2 (or any other function key, except for F7) to open a virtual
terminal.

- You can open an xterm, which is a shell environment within the X environment. To open an
xterm, click the computer icon on the panel, as shown in Figure 3.3, or (assuming you're in
GNOME), click Main Menu, System, GNOME terminal.

**Figure 3.3**
It would be better not to use X until you've learned the shell, but if you must use X, you can open a
virtual shell inside X.

A virtual terminal is one of the niftier tools you can use with Unix. Unix is not only multi-user in the
sense that multiple users can use the same computer, it is multi-user in the sense that multiple users
can use the same computer at the same time. You can hook up multiple terminals to the computer, or
you can use virtual terminals. To use a virtual terminal, press Alt+F[1-12], which means press the Alt
key plus any of the function keys (you can go up to as many virtual terminals as your system
supports). You are presented with a completely new login screen and you log in as a new user. You
can log in as yourself multiple times, or you can log in as a completely different user—it doesn't
matter. If you are in X, you can open a new virtual terminal by pressing Ctrl+Alt+Fx. To return to X,
press Alt+F7. (If you are running an X session, F7 is reserved for X.) Virtual terminals are particularly useful if you want to run several processes at one time in the foreground (we’ll talk about running processes in the foreground and the background later).

**Why You Should Not Log in As Root**

Now that you know about multiple logins and virtual terminals, it is time for the advice that every new Unix user gets. *The root account is intended for administrative use only. Do not login as root unless you must do so to perform an administrative task.*

Some new users get into the bad habit of using root for everything. This is bad for several reasons: You can inadvertently destroy important files. You can do irreparable damage to your filesystem or wreak havoc with system processes. Also, if you are connected to a network, being logged in as root is a potential security hole for a cracker to get into your system.

If you log in as root all the time, you increase your chances of making a simple error that can cost you hours of recovery time. If you are the administrator of your system, there is a simple command called `su`, which enables you to assume the identity of root or any other user (if you know the password). It is much better to log in as an ordinary user, then use the `su` command to give yourself administrative privileges to perform a task. This is also called becoming the "superuser." If you simply type

```
$ su
```

followed by the root password, you will assume the root identity. If you type

```
$ su other_username
```

followed by other user’s password, you will assume that user’s identity.

When you become the superuser, look twice at every command you enter to make sure that it is what you intended. When you are finished with the administrative task, type

```
# exit
```

You will re-assume your ordinary user identity.

---

Do not log in as root, unless you must do so to administer the system. Instead, use the `su` command to assume root status. Then, carefully review each command you type before pressing Enter. Carelessness can cause a lot of damage!

---

**The Unix Command**

Now we are ready to explore our environment by issuing commands to the shell. Unix has thousands of commands, which you can string together to make more complex commands. You can also create new commands in Unix. All this flexibility makes Unix an extremely powerful operating system.
The reason you have all this power is that commands are treated like any other executable file. When you type a command such as `ls`, you run the `ls` executable file, which is located in `/bin/ls`. When you create a new command, you simply add it to the `/bin` directory, then use it like any other Unix command. (If that doesn’t make sense, you will understand by the end of the hour.)

Unix commands come with switches and options, which can extend or slightly change the action that the command performs. Switches are composed of a dash followed by a letter or series of letters that alter or enhance the command’s output. Options consist of two dashes, followed by a word, that perform the same way as switches. In many cases, there is a switch and an option to perform the same task. A switch looks like this:

```
$ ls -a
```

An option looks like this:

```
$ ls --all
```

### Some Basic Unix Commands

After you have logged in for the first time (using your user account, not root!), you should get to know your surroundings. Directories in Linux are based on a tree structure, just like in DOS and Windows. The very base of the directory tree is called the root directory. Below the root directory are more directories, called subdirectories, and below the subdirectories are the files.

Directories and subdirectories are the same thing as folders in Windows.

---

### Getting Your Bearings with the `pwd` Command

If you lose track of the directory you’re in, you can type the `pwd` command to show you the full pathname of your current location in the filesystem:

```
$ pwd
/home/judith
```

`pwd` stands for *present working directory*. `pwd` prints the full pathname of your current directory, starting from the root directory and ending at your current directory.

Notice that the slashes that separate directories in Unix are forward slashes (`/`), not backward slashes (`\`). Backward slashes also have a meaning in Unix, which we will discuss later.

The root directory is the very lowest level in the directory tree. You might remember the root directory from Hour 2, when you had to assign a mount point to `/`. Every other directory in your
filesystem is a subdirectory of ./.

If you enter `pwd` immediately after logging in, you should see something like `/home/ username` print to the screen (where `username` is your user name). When you first log in to Linux, you enter the filesystem at your `home directory`. Your home directory is where you keep all your work, as well as your personal configuration and other system files.

**Listing Directory Contents with the `ls` Command**

So now you know where you are. The next step is to find out what’s in your home directory. To list the contents of any directory in Linux, type the following command:

```bash
$ ls
```

If you’re a new user, there probably won’t be anything in your directory, unless your system administrator added files. There actually are files in your directory; you just need to add a switch to the `ls` command to see them:

```bash
$ ls -a
...
   .bash_logout  .bash_profile  .bashrc  .emacs  .screenrc
```

The `-a` switch to the `ls` command means ”all.” That means to show all the files, including the hidden files that are displayed above. Hidden files are usually configuration files that are used by programs, but anyone can make a hidden file. Notice that no information is given about the files. The `ls -a` command doesn’t tell you if an item is a file or directory, who owns it, who has permission to use it, when it was created, or anything. To get more detailed information about the contents of a directory, you need yet another switch:

```bash
$ ls -la
```

```
total 28
  drwx------  2 judith  users  4096 Jul 18 14:29 .
  drwxr-xr-x 10 root   root   4096 Jul 18 14:29 ..
  -rw-r--r--  1 judith  users   24 Jul 18 14:29 .bash_logout
  -rw-r--r--  1 judith  users   230 Jul 18 14:29 .bash_profile
  -rw-r--r--  1 judith  users   124 Jul 18 14:29 .bashrc
  -rw-rw-r--  1 judith  users   688 Jul 18 14:29 .emacs
  -rw-r--r--  1 judith  users  3394 Jul 18 14:29 .screenrc
```

First, `ls -la` tells you the total size of your directory in blocks (1024 bytes) on the hard disk, which is 28 in our example. Each file and directory is listed with a lot of information.

Notice the first two entries in the `ls -la` output after the total size are simply a dot and a double-dot. If you are familiar with DOS, you have probably seen these before. The . stands for the current directory. The .. stands for the next-highest directory closer to root. Notice that both the . and the .. directories have a d at the beginning of the first string in the output. The d stands for directory. On some systems, that might be your only way of determining whether an item in a directory listing is a file or a subdirectory.

In Unix, commands are space and case-sensitive. If you forget the space between the command and the argument, or if you capitalize a file that should be lowercase, the
command will not work the way you intended. For example, if you enter cd.., you will get an error message.

If you have a color monitor, directories appear in blue in the output of an ls command on a default Red Hat 7.0 system. This behavior is arbitrary, however, and is easily changed.

The letters and dashes that come after the d list the permissions for the directory (and for the files, as well). Permissions specify who has access to the file, as well as what kind of access each kind of user has. The name judith means that the user judith owns the file or directory (notice that root owns the .. directory, which is /home). The date and time, July 18, 14:29, is the date and time that the file was last modified. Finally, the name of the file appears at the far right.

Since you haven’t created any files yet, your home directory is kind of boring. Let’s use another command to explore the system a bit further.

**Changing Directories with the cd Command**

The command to change directories is cd. At the shell prompt, type cd followed by the pathname of the directory where you want to go. For example, if you type

```bash
$ cd /
```
you will go to the root directory. If we type ls in this new directory, we see a new set of directories:

```bash
$ cd /
bin  dev home lost+found opt  root tmp var
boot etc lib  mnt     proc sbin usr
```

If we type pwd, we see:

```bash
$ pwd
/
```

That way, we know for sure that we are in the root directory. The symbol for the root directory is /. No matter where you are in the directory tree, you can always return to the root directory by typing cd /
and then pressing Enter.

**Copying Files with the cp Command**

Now we are at the second highest level in the directory tree. You will learn about all these directories in Hour 4, but for now let’s just explore a bit to help you learn some more commands. Enter the following commands:

```bash
$cd /usr/X11R6/lib
$ls
```
The output of the `ls` command is shown in **Figure 3.4**. We are going to copy the `XF86Config.eg` file from this directory to our home directory.

**Figure 3.4**
The `/usr/X11R6/lib/X11` contains files needed to run X. We’re just using a convenient sample configuration file from this directory as an example.

If you are not the administrator of your Linux system, you might not have access to any directories besides your home directory. If you don’t have access, ask your system administrator for some sample files to play with.

The command to copy a file in Linux is `cp`. To copy a file, enter `cp filename desired_location`. For example:

$ `cp XF86Config.eg /home/judith/`

This tells the system to copy the file `XF86Config.eg` from the current directory to the `/home/judith` directory.

You don’t have to be in a directory to copy a file from it. We could have stayed in `/home/judith` and written the following command instead:

$ `cp /usr/X11R6/lib/X11/XF86Config.eg /home/judith/`

or simply:


As you may recall, the . at the end of the last command stands for "the current directory."

Now let’s see what that file we just copied looks like. To return to our home directory from anywhere in the directory tree, we can simply type a tilde (~) at the shell prompt. Type `cd ~`, then `pwd` and see what happens:

$ `cd ~`
$ `pwd`
/home/judith

Now type `ls` and verify that the file is there:

$ `ls`
`XF86Config.eg`

**Reading Output with the `cat` and `more` Commands and Using Ctrl+C**

To view the contents of the file, we can use the `cat` command. Cat is short for concatenate, because you can also use it to combine the contents of two or more files.
$cat XF86Config.eg

This is a big file! This file is so big that it streams by too fast for you to read it. To slow down the `cat` command so that you can read one screenful of a long file at a time, you can pipe the `cat` command to the `more` command:

$cat XF86Config.eg | more

When you `pipe` a command in Unix, that means to execute one command, then apply the results of the first command to the second command. In this way, you can build some pretty powerful commands using smaller commands strung, or piped, together. The `more` command simply means display the output one screenful at a time. To scroll to the next screenful of text, press the spacebar. Continue pressing the spacebar until you reach the end of the file.

If you get tired of reading this very long file before you reach the end, you can issue the abort command, Ctrl+C.

Ctrl+C means to press Ctrl, then press C while holding down Ctrl. In this particular case, the shell isn’t case sensitive, so pressing Ctrl+c is the same as pressing Ctrl+C.

Ctrl+C stops the currently running command and displays a shell prompt, ready for the next command.

**Deleting Files with the `rm` Command**

After you have finished using the `XF86Config.eg` file, you should delete it from your home directory, since it doesn’t really belong there. To remove a file or directory, use the `rm` command:

$ rm /home/judith/XF86Config.eg

remove file ’XF86Config.eg’? y/n

y

Red Hat Linux gives you a prompt to make sure that you really want to delete the file. Other Unices and Linux distributions might simply delete the file with no warning.

There is no Undo button in Unix! Once you have deleted a file, it’s gone.

The commands you have learned so far are shown in Table 3.1, along with a few new commands that you can experiment with on your own.

**Table 3.1 A (Very) Few Common Unix Commands**
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pwd</code></td>
<td>Displays the present working directory</td>
</tr>
<tr>
<td><code>ls</code></td>
<td>Lists the contents of a directory</td>
</tr>
<tr>
<td><code>cd directory</code></td>
<td>Changes to <code>directory</code></td>
</tr>
<tr>
<td><code>cp file new_location</code></td>
<td>Copies <code>file</code> to a new location or <code>filename</code></td>
</tr>
<tr>
<td><code>cat file</code></td>
<td>Views <code>file</code></td>
</tr>
<tr>
<td><code>more</code></td>
<td>Displays the output one screenful at a time</td>
</tr>
<tr>
<td><code>Ctrl+C</code></td>
<td>Aborts the command (only works while the command is still running)</td>
</tr>
<tr>
<td><code>rm file</code></td>
<td>Removes (deletes) <code>file</code></td>
</tr>
<tr>
<td><code>mv file new_location</code></td>
<td>Moves a file to a new location, also used to rename a file</td>
</tr>
<tr>
<td><code>mkdir dir_name</code></td>
<td>Creates a new directory</td>
</tr>
<tr>
<td><code>rmdir dir_name</code></td>
<td>Removes an empty directory</td>
</tr>
<tr>
<td><code>rm -r dir_name</code></td>
<td>Removes a directory and all its contents</td>
</tr>
</tbody>
</table>

You can display the last few commands you entered by repeatedly pressing the up and down arrows. This way, you can avoid typing in long commands or filenames over again. You can edit the command on the command line, then press Enter to issue it again.

**Getting Help with the man and info Pages**

As you explore Linux, you will see many commands, system files, library functions, and other special files that you don’t understand. Your best and first reference when learning a new aspect of Linux is the set of man pages (short for manual pages). The man pages are the best way of discovering every possible variation and option that you can use with a command. They tend to be long and complicated, because they serve as a complete reference. However, even beginners can benefit greatly from them. Usually, if you read the first couple of paragraphs of the description, you can get a good idea of the topic.

Another reference is the info pages, which are somewhat newer than the man pages. Info pages also tend to provide more description and explanation than do the man pages.

**Task: Look Up Information on a Topic**

1. To look up info on a topic, type `info topic`, for example:

   ```
   $ info ls
   ```
2. To invoke a man page about a command or system file, type `man command` at the shell prompt. For example, to get help on the command `ls`, type

```
$ man ls
```

The manual page for that command opens, as shown in Figure 3.5. The first line of the man page is the name of the command, a synopsis of all the different switches and parameters that you can attach to the command, and a detailed description of the command usage.

3. To scroll through the man page, use the arrow down and arrow up buttons.

4. To quit the man page, enter `q`.

In addition to the man and info pages, many commands come with an option that prints out a description of the command, such as `command --help`. Also, all the documentation that comes with application packages that you install is stored in `/usr/doc`. Many of the files are simple README files or other notes, but you can sometimes find a wealth of information (the X documents are an example).

**Figure 3.5**
As a beginner, you can usually get the information you need by reading the first few paragraphs of a man page.

### Shutting Down

Unlike Windows systems, it is unnecessary and usually undesirable to continually shut down and reboot, unless you have a dual-boot computer. You can run your Linux system for as long as you like. Even after you install new software, there is no need to reboot.

If you do want to reboot, *never* just turn off your computer or hit the reset button. This can cause data corruption. If you are forced to turn off the computer or you lose power, you might have to run a program called `fsck` to restore your filesystem. You will learn more about `fsck` in Hour 16, "Administering the System."

- To shut down and power down the computer, enter the following command:

```
# halt
```

- To shutdown and reboot, enter:

```
# shutdown -r now
```

- In Red Hat Linux, you can also press the Ctrl+Alt+Del combination, which invokes the `shutdown -r now` command.
By default in Red Hat Linux 7.0, you do not have to be root to shut down. There are steps you can take, however, to prevent ordinary users from shutting down (see Hour 16).

Summary

In this hour, you learned how to start thinking like a Linux user. You learned how the Linux boots and initializes, and how to log in. You learned the basics of the Unix command, and you used the `ls`, `pwd`, `cd`, `cp`, `cat`, `more`, and `rm` commands. You also learned how to use a pipe to link commands and how to use Ctrl+C to abort a command. You discovered the man and info pages and how to safely shut down a Linux system.

Workshop

The Workshop contains quiz questions and exercises to help reinforce what you’ve learned in this hour.

Q&A

Q. I’m still really confused. Where can I get more information about Linux basics?

A. Learning how to use Linux takes more than one hour of study. Here are some resources for you to continue studying the basics:


- [http://www.linuxnewbie.org](http://www.linuxnewbie.org) for general help for Linux newbies.

Q. I don’t see my floppy drive or CD-ROM drive in the root directory. Where are they?

A. Linux treats different drives and other devices differently than MS-DOS or Windows. Other drive devices, including floppy drives, ZIP drives, tape drives, and CD-ROMs are usually found in the `/mnt` directory. You will learn more in the next hour.

Quiz

1. What is X?

2. In which environment do you use the command line?

3. What is the first thing that happens when you turn on your computer?

4. What is the login program called, and what starts it?
5. Which login sequence is runlevel 3, and which is runlevel 5?

6. What is a virtual terminal?

7. When is it okay to log in as root?

8. How do you leave superuser status?

Quiz Answers

1. What is X?
   The Linux (actually also the Unix) graphical interface.

2. In which environment do you use the command line?
   The shell environment.

3. What is the first thing that happens when you turn on your computer?
   The BIOS copies the boot loader into memory.

4. What is the login program called, and what starts it?
   The kernel starts the getty program, which makes it possible to login.

5. Which login sequence is runlevel 3, and which is runlevel 5?
   Runlevel 3 starts the text login, and runlevel 5 starts the graphical login.

6. What is a virtual terminal?
   A virtual terminal is like another monitor attached to your system. It enables you to start a completely different session with a new login.

7. When is it okay to log in as root?
   You should log in as root only when you need to, in order to perform an administrative task.

8. How do you leave superuser status?
   Type `exit` at the command prompt.

Exercises

1. Log into your Linux system using the account you created during installation.
2. Enter the command `pwd`. What is the output?

3. If you know the root password, assume superuser status and then exit.

4. Type `ls /etc`. What command would you use to display only one screenful of information at a time?

5. Type `pwd` again. What directory are you in? How would you go to the `/etc` directory?

6. Copy the `termcap` file from the `/etc` directory to your home directory.

7. Make a new directory in your home directory called "play." Put the `termcap` file in the play directory.

8. Rename the `termcap` file to `this_file`. Hint: use the `mv` command.

9. Delete `this_file`. Remove the play directory.

**Exercise Answers**

1. Log into your Linux system using the account you created during installation.
   
   Enter your user name and press Enter, then enter your password and press Enter.

2. Enter the command `pwd`. What is the output?

   Normally, you log in to your home directory. The output should be something like /home/your_user_name.

3. If you know the root password, assume superuser status and then exit.

   Type `su`, then enter the root password at the prompt. Type `exit` to exit back to your own user identity.

4. Type `ls /etc`. What command would you use to display only one screenful of information at a time?

   The `/etc` directory is probably too long for the entire output to appear in one screen. Type `ls /etc | more` to display one screenful of output at a time.

5. Type `pwd` again. What directory are you in? How would you go to the `/etc` directory?

   You are still in your home directory. To go to the `/etc` directory, type `cd /etc`.

6. Copy the `termcap` file from the `/etc` directory to your home directory.

   `cp /etc/termcap /home/username`
7. Make a new directory in your home directory called "play." Put the `termcap` file in the play directory.

    mkdir play
    cp termcap play

8. Rename the `termcap` file to `this_file`. Hint: use the `mv` command.

    cd play
    mv termcap this_file

9. Delete `this_file`. Remove the play directory.

    rm this_file
    cd ..
    rmdir play

© Copyright Macmillan USA. All rights reserved.