3

Fundamental Command Line Skills

CERTIFICATION OBJECTIVES

The Red Hat exams are an advanced challenge. This chapter covers RHCSA requirements that were formerly listed as prerequisites for the now-obsolete RHCT certification. Many of these requirements specify basic command line tools associated with entry-level certifications such as those offered by the Linux Professional Institute.

As such command line skills are no longer listed as prerequisites, they have been combined with networking configuration objectives, something you can set up with some of these command line tools.

As most candidates for the RHCSA exam should already be familiar with these command line tools, this chapter covers the related topics in a minimum of detail. If after reading this chapter, you feel the need for more guidance about these topics, other excellent beginning Linux books described in Chapter 1 can help.

Linux gurus should recognize that I've "oversimplified" a number of explanations, to keep this chapter as short as possible. But as most IT professionals are specialists, you may feel a bit uncertain about a few topics in this chapter. That is okay. In fact, it's natural that many experienced Linux administrators don't frequently use every command. Many candidates are successfully able to fill in the gaps in their knowledge with some self-study and practice.

INSIDE THE EXAM

Shells

The related RHCSA exam objective is pretty generic:

- Access a shell prompt and issue commands with correct syntax; use pipelines and I/O redirection

The default shell for Linux is bash, the "Bourne-Again shell." In fact, the original release of the RHCSA objectives specified the use of bash. However, many Linux gurus use one of the many other shells available.

Whatever shell you select, you need to know how to get to a shell prompt and run regular commands from that prompt. Some basic commands are described in some of the other objectives. It's fairly easy to open a shell prompt from a console and within the GUI.

Pipelines and Redirection

Data into and out of a shell are often thought of in Linux as streams of information. One basic Linux skill is the ability to redirect those streams. As described in the RHCSA requirements, that's the ability to

- Use input/output redirection (>, >>, |, 2>, etc.)

The operators in parentheses can redirect the streams from command output, command error, data files, and more.

File and Directory Management

So when you get to a command line, what's next? That's the province of file and directory management. With related commands, you can navigate around the Linux directory tree as well as perform all of the tasks suggested in the related objectives:

- Create/delete/copy/move files and directories
- Create hard and soft links

The Analysis of Text Output

Most Linux configuration files are text files. As such, it's important to understand and analyze the flow of text as it is sent through the shell. To that end, tools like the **grep** command can help focus on needed information. In that way, you'll examine how to meet the following objective:

- Use **grep** and regular expressions to analyze text output

The Variety of Local Documentation

While Internet access is not available during the Red Hat exams, that's okay. Google is not your only friend. Linux has some excellent documentation installed with most packages. Command manuals are also available. The following objective is straightforward, as it describes the commands and directory associated with most Linux online documentation.

- Locate, read, and use system documentation using **man**, **info**, and files in /usr/share/doc

The objectives include an interesting modifier to that objective:

- Note: Red Hat may use applications during the exam that are not included in Red Hat Enterprise Linux for the purpose of evaluating candidate's abilities to meet this objective.

Most Linux developers follow the basic parameters just described for system documentation. Does Red Hat's "note" mean they'll "hide" some key information in a man page or a file in the /usr/share/doc directory? The wording suggests you need to be prepared for such a scenario.

The Use of Text Editors

To configure Linux, you need to know how to edit text files. And for those newer to Linux, that requires a different paradigm. While word processors like OpenOffice.org Writer and Microsoft Word can save files in text format, a mistake with a key configuration file can render a Linux system unbootable. So you need to know how to handle the following objective:

- Create and edit text files

The Management of Network Services

While there are excellent GUI tools to help manage network services, mistakes are too easy to make with such tools. Command line tools can help you understand and manage network services directly or through associated configuration files. The associated objective is

- Start, stop, and check the status of network services

Of course, this objective requires a basic understanding of IP networking.

The Configuration of Networking and Name Resolution

Name resolution depends on databases of hostnames or fully qualified domain names (FQDN) such as server1.example.com and IP addresses like 192.168.122.50. Name resolution depends on the local hostname, the local /etc/hosts database of hostnames and IP addresses, and available databases of Domain Name Service (DNS) servers as well. That is an interpretation of the following RHCSA objective:

- Configure networking and hostname resolution statically or dynamically

When the RHCSA was first released, this was depicted as two objectives. While these objectives are no longer officially in effect, they do provide more information on what it means to configure networking and hostname resolution:

- Manage network devices: understand basic IP networking/routing, configure IP addresses/default route statically or dynamically
- Manage name resolution: set local hostname, configure /etc/hosts, configure to use existing DNS server

While network troubleshooting is no longer a part of the entry-level Red Hat exam, the way you address problems with respect to network configuration and hostname resolution can help you better understand how networks operate.

CERTIFICATION OBJECTIVE 3.01

Shells

A *shell* is a user interface. It is also used as a command line interpreter. In Linux, the shell is the interpreter that allows you to interact with Linux using various commands. With the right file permissions, you can set up commands in scripts to run as needed, even in the middle of the night. Linux shells can process commands in various sequences, depending on how you manage the input and output of each command. The way commands are interpreted is in part determined by variables and parameters associated with each shell.

The default shell in Linux is bash, also known as the Bourne-Again Shell. The focus of commands in this book is based on how they're used in bash. However, a number of other shells are available that are popular with many users. As long as the appropriate RPMs are installed, users can start any of these shells. If desired, you can change the default shell for individual users in the /etc/passwd file.

Other Shells

As four shells are included with RHEL 6, users have their choice in command line interpreters. While bash is the default, long-time Linux and Unix users may prefer something else:

- **bash** The default Bourne-Again shell, based on the command line interpreter originally developed by Stephen Bourne.
- **dash** A simpler shell with fewer features than bash, but faster.
- **tcsh** An enhanced version of the Unix C shell.
- **zsh** A sophisticated shell, similar to the Korn shell.

These shells are configured in the /bin directory. If a user prefers one of these options as their default shell, it's easy to change. The most direct method is to change the default shell in the /etc/passwd file. For example, the line that applies to my regular account is

```
michael:x:1000:1000:Michael Jang:/home/michael:/bin/bash
```

For example, to change the default to the dash shell, change /bin/bash to /bin/dash.

exam Watch

Even though it should be trivial for most Linux users, a part of one RHCSA objective is to "access a shell prompt." You should now know how to set up access to different shell prompts.

Terminal Consoles

By default, six command line consoles are available on RHEL systems. They're defined by the start-ttys.conf file in the /etc/init directory. Take a look at that file. You'll see that consoles are defined for runlevels 2, 3, 4, and 5. Active consoles are defined as device files /dev/tty1 through /dev/tty6. When a GUI is configured, it takes /dev/tty1. It's possible to configure more virtual consoles, limited by those allowed for the root administrative user in the /etc/securetty file.

Normally, to change between consoles, press ALT and the function key associated with the console. For example, the ALT-F2 key combination moves to the second console. However, in the RHEL GUI, the ALT-F2 key combination is used to start the Run Application tool; therefore, you'll need to press CTRL-ALT-F2 to move to that second virtual console.

At a text console login, you'd see the following prompt, which depends a bit on the release of RHEL, the version number of the kernel, and the system hostname:

```
Red Hat Enterprise Linux release 6.0 (Santiago)
Kernel 2.6.32-71.el6.x86_64 on x86_64

server1 login:
```

The graphical login, which requires the installation of the GNOME Display Manager (GDM), is more intuitive, as shown in Figure 3-1.

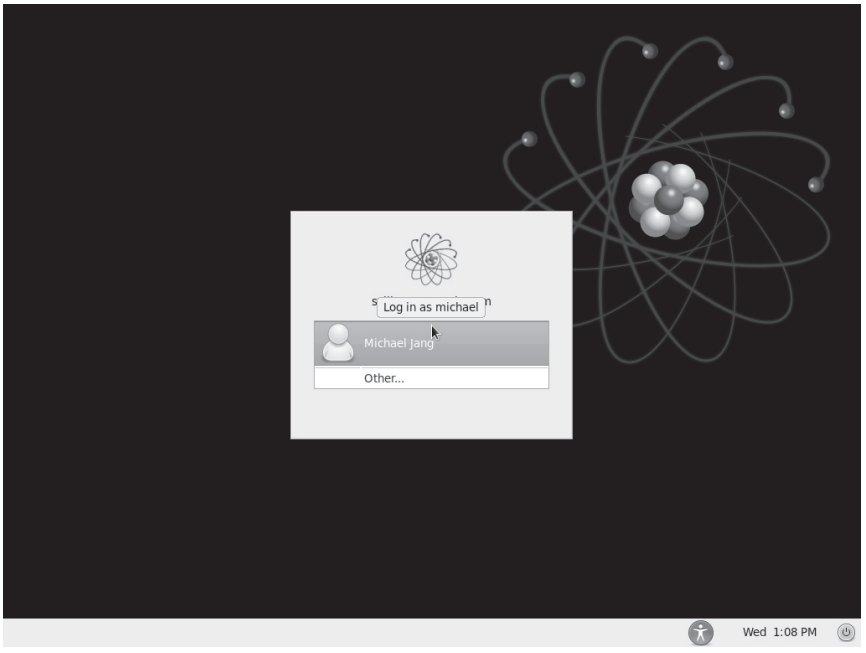
GUI Shell Interfaces

Once logged in to the GUI, access to the bash shell is easy. If you're in the default GNOME desktop environment, click Applications | System Tools | Terminal. Traditionally, administrators have worked from the console. But in many cases, working the command line from the GUI can be helpful, especially with the consoles that can be placed side by side. A right-click on a GUI terminal screen supports opening of additional terminals in different windows or in tabs. It also supports copy and paste as needed.

The screenshots of the command line taken for this book are based on the GUI-based command line, in part because dark text on a white screen is easier to read.

FIGURE 3-1

A first GUI login console



Differences Between Regular and Administrative Users

What you can do at the command line depends on the privileges associated with the login account. Two basic prompts are available. The following is an example of what you might see when logged in as a regular user:

```
[michael@server1 ~]$
```

Note how it includes the username, the hostname of the local system, the current directory, and a $ prompt. The $ prompt is the standard for regular users. As noted in the introduction to the book, examples of commands run from a regular user account just show the following:

```
$
```

In contrast, take a look at a prompt for the root administrative user on the same system. It should look familiar. Except for the name of the account, the only consistent difference is the prompt.

```
[root@server1 ~]#
```

So examples of commands run from the root administrative account just show the following:

```
#
```

Besides ownership and permissions, other differences between regular and administrative accounts are discussed in Chapter 8.

Text Streams and Command Redirection

Linux uses three basic data streams. Data goes in, data comes out, and errors are sent in a different direction. These streams are known as standard input (stdin), standard output (stdout), and standard error (stderr). Normally, input comes from the keyboard and goes out to the screen, while errors are sent to a buffer. Error messages are also sent to the display (as text stream 2). In the following example, *filename* is stdin to the **cat** command:

```
# cat filename
```

When you run **cat *filename*,** the contents of that file are sent to the screen as standard output.

You can redirect each of these streams to or from a file. For example, if you have a program named database and a datafile with a lot of data, the contents of that datafile can be sent to the database program with a left redirection arrow (**<**). As shown here, datafile is taken as standard input:

```
# database < datafile
```

Standard input can come from the left side of a command as well. For example, if you need to scroll through the boot messages, you can combine the **dmesg** and **less** commands with a pipe:

```
# dmesg | less
```

The output from **dmesg** is redirected as standard input to **less**, which then allows you to scroll through that output as if it were a separate file.

Standard output is just as easy to redirect. For example, the following command uses the right redirection arrow (**>**) to send the standard output of the **ls** command to the file named filelist.

```
# ls > filelist
```

exam Watch

Command redirection symbols like >, >>, 2>, and | are associated with the "input/output redirection" objective in the RHCSA exam objectives.

You can add standard output to the end of an existing file with a double redirection arrow with a command such as **ls >> filelist**.

If you believe that a particular program is generating errors, redirect the error stream from it with a command like the following:

```
# program 2> err-list
```

CERTIFICATION OBJECTIVE 3.02

Standard Command Line Tools

While newer Linux users may prefer to use the GUI, the most efficient way to administer Linux is from the command line interface. While excellent GUI tools are available, the look and feel of those tools varies widely by distribution. In contrast, if you know the standard command line tools, you'll be able to find your way around every Linux distribution.

Two basic groups of commands are used to manage Linux files. One group helps you get around Linux files and directories. The other group actually does something creative with the files. Remember, in any Linux file operation, you can take advantage of the HISTORY (this is capitalized because it's a standard environment variable) of previous commands, as well as the characteristics of command completion, which allow you to use the TAB key almost as a wildcard to complete a command or a filename, or give you the options available in terms of the absolute path.

exam Watch

This section covers only the most basic of commands available in Linux. It describes only a few capabilities of each command. Nevertheless, it allows you to "issue commands with correct syntax," as described in the RHCSA objectives.

Almost all Linux commands include switches, options that allow you to do more. Few are covered in this chapter. If you're less familiar with any of these commands, use their man pages. Study the switches. Try them out! Only with practice, practice, and more practice can you really understand the power behind some of these commands.

File and Directory Concepts

As noted previously, everything in Linux can be reduced to a file. Directories are special types of files that serve as containers for other files. To navigate and find important files, you need some basic concepts to tell you where you are and how to move from directory to directory. The command is **pwd**, a variable that always leads to a user's home directory is the tilde (~), and the concept that describes where you are in the Linux directory tree is the path. Closely related are the directories searched when a command is typed in, which is based on the environment variable known as the PATH. Once these concepts are understood, you can navigate between directories with the **cd** command.

pwd

At the command line interface, the current directory may be either in the top-level root (/) directory, or a subdirectory. The **pwd** command identifies the current directory. Try it out. It'll give you a directory name relative to the top-level root directory (/). With this information in hand, you can move to a different directory if needed. Incidentally, **pwd** is short for print working directory (which has nothing to do with modern printers, but respects the days when output was printed on a teletype). For example, when I run that command in my home directory, I get the following output:

```
/home/michael
```

The Tilde (~)

Upon a standard login, every Linux user is taken to a home directory. The tilde (~) can be used to represent the home directory of any currently active user. For example, when user john logs in, he's taken to his home directory, /home/john. In contrast, the home directory of the root administrative user is /root.

Thus, the effect of the **cd ~** command depends on your username. For example, if you've logged in as user mj, the **cd ~** command navigates to the /home/mj directory. If you've logged in as the root user, this command navigates to the /root directory. You can list the contents of your home directory from anywhere in the directory tree with the **ls ~** command. The **cd** and **ls** commands are described shortly. When I log in as the root administrative user and run the **ls** command, I see:

```
anaconda-ks.cfg  install.log  install.log.syslog
```

Incidentally, these files describe what happened during the installation process, the packages that were installed, and the users and groups added to the local system. The anaconda-ks.cfg command is important for automated Kickstart installations, as described in Chapter 2.

Directory Paths

There are two path concepts you need to know when working with Linux directories: absolute paths and relative paths. An absolute path describes the complete directory structure in terms of the top-level directory, root (/). A relative path is based on the current directory. Relative paths do not include the slash in front.

The difference between an absolute path and a relative one is important. Especially when creating a script, absolute paths are essential. Otherwise, scripts executed from other directories may lead to unintended consequences. For example, say you're in the top-level root directory, and you have backed up the /home directory using the relative path. If you happen to be in the /home directory when restoring that backup, the files for user michael would be restored to the /home/home/michael directory.

In contrast, if the /home directory was backed up using the absolute path, the current directory doesn't matter when restoring these files. That backup will be restored to the correct directories.

Environment PATHs

Strictly speaking, when running a command, you should cite the full path to that command. For example, since the **ls** command is in the /bin directory, users should actually run the **/bin/ls** command to list files in the current directory.

With the benefit of the PATH, an environment variable, that's not required. The bash shell automatically searches through the directories listed in a user's PATH for the command that user just typed at the command line. Environment variables are constant from console to console.

To determine the PATH for the current user account, run the **echo $PATH** command. You should see a series of directories in the output. The differences between the PATH for a regular user and one for a root user have narrowed in RHEL 6:

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin

# echo $PATH
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
```

Today, the directories in the PATH for regular and the root administrative users are essentially the same. But the differences matter, as the directories are searched in order. For example, the **system-config-keyboard** command is available from both the /usr/bin and /usr/sbin directories. As you can see from the default PATH for regular and root users, the version that is run varies because of the differences in the PATH.

The PATH is determined globally by current settings in the /etc/profile file. You might notice differences between the PATH as configured for User ID (UID) 0 and all other users. UID 0 corresponds to the root administrative user.

The PATH for individual users can be customized with an appropriate entry in that user's home directory, in the hidden file named .profile.

cd

It's easy to change directories in Linux. Just use **cd** and cite the absolute path of the desired directory. If you use the relative path, just remember that the destination depends on the present working directory.

By default, the **cd** command by itself navigates to your home directory. The tilde is not required for that command.

File Lists and ls

Now that you've reviewed those commands that can navigate from one directory to another, it's time to see what files exist in a directory. And that's the province of the **ls** command.

The Linux **ls** command, with the right switches, can be quite powerful. The right kind of **ls** can tell you everything about a file, such as creation date, last access date, and size. It can help you organize the listing of files in just about any desired order. Important variations on this command include **ls -a** to reveal hidden files, **ls -l** for long listings, **ls -t** for a time-based list, and **ls -i** for inode numbers. You can combine switches; I often use the **ls -ltr** command to display the most recently changed files last. The **-d** switch, when combined with others, can give you more information on the current directory.

One important feature that returns SELinux contexts is the **ls -Z** command. Take a look at the output in Figure 3-2. The system_u, object_r, var_t, and s0 output demonstrates the current SELinux contexts of the noted files. During the RHCSA exam (and RHCE as well), you'll be expected to configure a system with SELinux enabled. Starting with Chapter 4, this book covers how SELinux can be configured for every service that's installed.

FIGURE 3-2

Current SELinux contexts

```
[root@server1 ~]# \ls -Z /var/
drwxr-xr-x. root root system_u:object_r:acct_data_t:s0 account
drwxr-xr-x. root root system_u:object_r:var_t:s0       cache
drwxr-xr-x. root root system_u:object_r:var_t:s0       crash
drwxr-xr-x. root root system_u:object_r:cvs_data_t:s0  cvs
drwxr-xr-x. root root system_u:object_r:var_t:s0       db
drwxr-xr-x. root root system_u:object_r:var_t:s0       empty
drwxr-xr-x. root root system_u:object_r:games_data_t:s0 games
drwxrwx--T. root gdm  system_u:object_r:xserver_log_t:s0 gdm
drwxr-xr-x. root root system_u:object_r:var_lib_t:s0   lib
drwxr-xr-x. root root system_u:object_r:var_t:s0       local
drwxrwxr-x. root lock system_u:object_r:var_lock_t:s0  lock
drwxr-xr-x. root root system_u:object_r:var_log_t:s0   log
lrwxrwxrwx. root root system_u:object_r:mail_spool_t:s0 mail -> spool/mail
drwxr-xr-x. root root system_u:object_r:var_t:s0       nis
drwxr-xr-x. root root system_u:object_r:var_t:s0       opt
drwxr-xr-x. root root system_u:object_r:var_t:s0       preserve
drwxr-xr-x. root root system_u:object_r:var_t:s0       report
drwxr-xr-x. root root system_u:object_r:var_run_t:s0   run
drwxr-xr-x. root root system_u:object_r:var_spool_t:s0 spool
drwxrwxrwt. root root system_u:object_r:tmp_t:s0       tmp
drwxr-xr-x. root root system_u:object_r:var_yp_t:s0    yp
[root@server1 ~]#
```

File Creation Commands

Two commands are used to create new files: **touch** and **cp**. Alternatively, you can let a text editor such as vi create a new file. Of course, while the **ln**, **mv**, and **rm** commands don't create files, they do manage them in related ways.

touch

Perhaps the simplest way to create a new file is with the **touch** command. For example, the **touch abc** command creates an empty file named abc in the local directory. The **touch** command is also used to change the last access date of a file. For example, try the following three commands:

```
# ls -l /etc/passwd
# touch /etc/passwd
# ls -l /etc/passwd
```

Note the date and time associated with the output of each **ls -l** command. The change is associated with the current date and time, which is associated with the **date** command.

cp

The **cp** (copy) command allows you to take the contents of one file and place a copy with the same or different name in the directory of your choice. For example, the **cp *file1 file2*** command takes the contents of *file1* and saves the contents in *file2*. One of the dangers of **cp** is that it can easily overwrite files in different directories, without prompting you to make sure that's what you really wanted to do.

The **cp** command, with the **-r** switch, supports recursive changes. For example, the following command copies all subdirectories of the noted directory, along with associated files:

```
# cp -ar /usr/share/doc/. /doc/
```

mv

While you can't rename a file in Linux, you can move it. The **mv** command essentially puts a different label on a file. For example, the **mv *file1 file2*** command changes the name of *file1* to *file2*. Unless you're moving the file to a different partition, everything about the file, including the inode number, remains the same. The **mv** command works with directories too.

ln

Linked files allow users to edit the same file from different directories. When linked files are devices, they may represent more common names, such as /dev/dvd. Linked files can be hard or soft.

Hard links include a copy of the file. As long as the hard link is made within the same partition, the inode numbers are identical. You could delete a hard-linked file in one directory, and it would still exist in the other directory. For example, the following command creates a hard link from the actual Samba configuration file to smb.conf in the local directory:

```
# ln /etc/samba/smb.conf smb.conf
```

On the other hand, a soft link serves as a redirect; when you open a file created with a soft link, the link redirects you to the original file. If you delete the original file, the file is lost. While the soft link is still there, it has nowhere to go. The following command is an example of how you can create a soft linked file:

```
# ln -s /etc/samba/smb.conf smb.conf
```

rm

The **rm** command is somewhat dangerous. At the Linux command line, there is no trash bin. So if you delete a file with the **rm** command, it's at best difficult to recover that file.

The **rm** command is powerful. For example, when I downloaded the source files for the Linux kernel, it included several thousand files in the /root/rpmbuild/BUILD/kernel-2.6.32-71.1.el6 directory. As it's not practical to delete those files one by one, the **rm** command includes some powerful switches. The following command removes all of those files in one command:

```
# rm -rf /root/rpmbuild/BUILD/kernel-2.6.32-71.1.el6
```

The **-r** switch works recursively, and the **-f** switch overrides any safety precautions, such as shown in the output to the **alias** command for the root administrative user. It's still quite a dangerous command, as a simple typing mistake that puts a space between the first forward slash and the directory name, as shown here:

```
# rm -rf / root/rpmbuild/BUILD/kernel-2.6.32-71.1.el6
```

would first delete every file starting with the top-level root directory, before looking for the root/rpmbuild/BUILD/kernel-2.6.32-71.1.el6 subdirectory.

Directory Creation and Deletion

The **mkdir** and **rmdir** commands are used to create and delete directories. The ways these commands are used depend on the already-discussed concepts of absolute and relative paths. For example, the following command creates the test subdirectory to the current directory. If you're currently in the /home/michael directory, the full path would be /home/michael/test.

```
# mkdir test
```

Alternatively, the following command creates the /test directory:

```
# mkdir /test
```

If desired, the following command creates a series of directories:

```
# mkdir -p /test1/test2/test3
```

That command is equivalent to the following commands:

```
# mkdir /test1
# mkdir /test1/test2
# mkdir /test1/test2/test3
```

Conversely, the **rmdir** command deletes a directory only if it's empty. If you're cleaning up after the previous **mkdir** commands, the **-p** switch is useful there as well. The following command deletes the noted directory and subdirectories, as long as all of the directories are otherwise empty:

```
# rmdir -p /test1/test2/test3
```

alias

The **alias** command can be used to simplify a few commands. For the root administrative user, the default aliases provides a bit of safety. To see the aliases for the current user, run the **alias** command. The following output is the default Red Hat aliases for the root user:

```
alias cp='cp -i'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias
--show-dot --show-tilde'
```

Some of these aliases help protect key files from mistakes. The **-i** switch prompts the user for confirmation before a file is deleted or overwritten with the **cp**, **mv**, or **rm** command. Just be aware, the **-f** switch supersedes the **-i** for the noted commands.

on the Job

As suggested by the technical editor, some administrators set a different alias for the* rm *command:* alias rm='mv -t ~/.Trash'. *Files in that directory are like a standard trashbin. In contrast, the default trash directory for the GNOME desktop can be found in each user's .local/share/Trash/files/ subdirectory.

Wildcards

Sometimes you may not know the exact name of the file or the exact search term. That is when a wildcard is handy, especially with the commands described throughout the book. Three basic wildcards are shown in Table 3-1.

on the Job

Wildcards are sometimes known in the Linux world as globbing.

File Searches

Most users who study Linux for a while become familiar with key files. For example, named.conf is the key configuration file for the standard DNS (Domain Name Service) servers, based on the Berkeley Internet Name Domain (BIND). But not many people remember that the sample named.conf file, with all kinds of useful configuration hints, can be found in the /usr/share/doc/bind-*/sample/etc directory.

To that end, there are two basic commands for file searches: **find** and **locate**.

find

The **find** command searches through directories and subdirectories for a desired file. For example, if you wanted to find the directory with the named.conf DNS sample configuration file, you could use the following command, which would start the search in the root directory:

```
# find / -name named.conf
```

TABLE 3-1

Wildcards in the Shell

Wildcard	Description
*	Any number of alphanumeric characters (or no characters at all). For example, the **ls ab*** command would return the following filenames, assuming they exist in the current directory: ab, abc, abcd.
?	One single alphanumeric character: For example, the **ls ab?** command would return the following filenames, assuming they exist in the current directory: abc, abd, abe.
[]	A range of options. For example, the **ls ab[123]** command would return the following filenames, assuming they exist in the current directory: ab1, ab2, ab3. Alternatively, the **ls ab[X-Z]** command would return the following filenames, assuming they exist in the current directory: abX, abY, abZ.

But the speed of that search depends on the memory and processing power available on the local system. With the advent of virtual machines, that processing power may be relatively small. Alternatively, if you know that this file is located in the /usr subdirectory tree, you could start in that directory with the following command:

```
# find /usr -name named.conf
```

That command should now find the desired file more quickly.

locate

If this is all too time consuming, RHEL allows you to set up a database of installed files and directories. Searches with the **locate** command are almost instantaneous. And **locate** searches don't require the full filename. The drawback is that the **locate** command database is normally updated only once each day, as documented in the **/etc/cron.daily/mlocate.cron** script.

exam Watch

When I've taken Red Hat exams, I ran the noted* mlocate.cron *script to help me find needed files more quickly.

As daily jobs are run only once every 24 hours, that's not good enough, especially during a 2.5 hour exam. Fortunately, the noted script can be executed directly from the command line interface, by the root administrative user. Just type in the full path to the file as if it were a command:

```
# /etc/cron.daily/mlocate.cron
```

CERTIFICATION OBJECTIVE 3.03

The Management of Text Files

Linux and Unix are managed through a series of text files. Linux administrators do not normally use graphical editors to manage these configuration files. Editors such as WordPerfect, OpenOffice.org Writer, and yes, even Microsoft Word normally either save files in a binary format or add tags. Unless text files are preserved in their original format, without tags, changes that are made can render a Linux system unbootable.

Linux commands have been set up to manage text files as streams of data. You've seen tools such as redirection arrows and pipes. But that data can be overwhelming

without tools that can sort through that data. But even before files are edited, it's important to know how to read these files at the command line interface.

Commands to Read Text Streams

Previously, you reviewed commands like **cd**, **ls**, and **pwd** that can help you get around Linux files. With commands like **find** and **locate**, you reviewed how to identify the location of desired files.

Now it's time to start reading, copying, and moving the files around. Most Linux configuration files are text files. Linux editors are text editors. Linux commands are designed to read text files. To identify the types of files in the current directory, try the **file *** command.

cat

The most basic command for reading files is **cat**. The **cat *filename*** command scrolls the text within the *filename* file. It also works with multiple filenames; it concatenates the filenames that you might list as one continuous output to your screen. You can redirect the output to the filename of your choice, as described in the section "Text Streams and Command Redirection."

less and more

Larger files demand a command that can help you scroll through the file text at your leisure. Linux has two of these commands: **more** and **less**. With the **more *filename*** command, you can scroll through the text of a file, from start to finish, one screen at a time. With the **less *filename*** command, you can scroll in both directions through the same text with the PAGE UP and PAGE DOWN keys. Both commands support vi-style searches.

As the **less** and **more** commands do not change files, they're an excellent way to scroll through and search for items in a large text file such as an error log. For example, to search through the basic /var/log/messages file, run the following command:

```
# less /var/log/messages
```

You'll then be able to scroll up and down the log file for important information. You can then use the forward slash and question mark to search through the file. For example, once you've run the command just shown, you'll be taken to a screen similar to that shown in Figure 3-3.

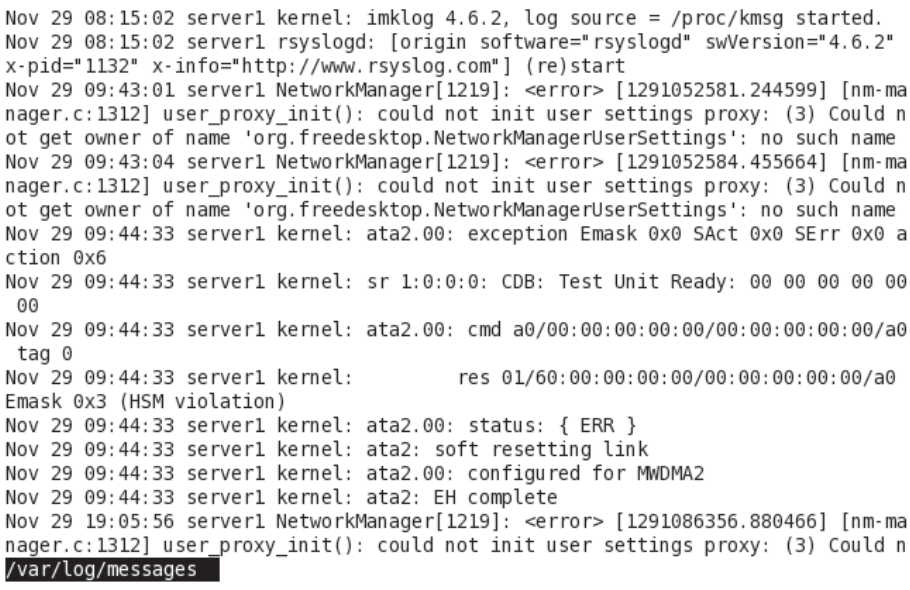


FIGURE 3-3

The less pager and /var/log/messages

For example, to search forward in the file for the term "IPv4 tunneling," type the following in the pager:

```
/IPv4 tunneling
```

To search in the reverse direction, substitute a **?** for the **/**.

The **less** command has one more feature unavailable to commands like **more** and **cat**; it can read text files compressed in Gzip format, normally shown with the .gz extension. For example, the man pages associated with many standard commands that are run in the shell can be found in the /usr/share/man/man1 directory. All of the files in this directory are compressed in .gz format. Nevertheless, the **less** command can read those files, without uncompressing them.

And that points to the operation of the **man** command. In other words, these two commands are functionally equivalent:

```
# man cat
# less /usr/share/man/man1/cat.1.gz
```

head and tail

The **head** and **tail** commands are separate commands that work in essentially the same way. By default, the **head *filename*** command looks at the first 10 lines of a file; the **tail *filename*** command looks at the last 10 lines of a file. You can specify the number of lines shown with the **-nxy** switch. Just remember to avoid the space when specifying the number of lines; for example, the **tail -n15 /etc/passwd** command lists the last 15 lines of the /etc/passwd file.

The **tail** command can be especially useful for problems in progress. For example, if there's an ongoing problem, the following command monitors the noted file for login attempts:

```
# tail -f /var/log/secure
```

Commands to Process Text Streams

A text stream is the movement of data. For example, the **cat *filename*** command streams the data from the *filename* file to the screen. When these files get large, it's convenient to have commands that can filter and otherwise process these streams of text.

To that end, Linux includes simple commands to help you search, check, or sort the contents of a file. And there are special files that contain others; some of these container files are known colloquially as "tarballs."

on the job

Tarballs are a common way to distribute Linux packages. They are normally distributed in a compressed format, with a .tar.gz or .tgz file extension, consolidated as a package in a single file.

sort

You can sort the contents of a file in a number of ways. By default, the **sort** command sorts the contents in alphabetical order depending on the first letter in each line. For example, the **sort /etc/passwd** command would sort all users (including those associated with specific services and such) by username.

grep and egrep

The **grep** command uses a search term to look through a file. It returns the full line that contains the search term. For example, **grep 'Michael Jang' /etc/passwd** looks for the name of this author in the /etc/passwd file.

The **egrep** command is more forgiving; it allows you to use some unusual characters in your search, including +, ?, |, (, and). While it's possible to set up **grep** to search for these characters with the help of the backslash, the command can be awkward.

diff

One useful option to find the difference between files is the **diff** command. If you've just used a tool such as the Network Connections tool described later in this chapter, it'll modify a file such as ifcfg-eth0 in the /etc/sysconfig/network-scripts directory.

If you've backed up that ifcfg-eth0 file, the **diff** command can identify the differences between the two files. For example, the following command identifies the differences between the ifcfg-eth0 file in the /root and the /etc/sysconfig/network-scripts directories:

```
# diff /root/ifcfg-eth0 /etc/sysconfig/network-scripts/ifcfg-eth0
```

So if you've backed up the ifcfg-eth0 file to the /root directory, the command shown can help identify changes made by configuration tools.

wc

The **wc** command, short for word count, can return the number of lines, words, and characters in a file. The **wc** options are straightforward; for example, **wc *-w filename*** returns the number of words in that file.

sed

The **sed** command, short for stream editor, allows you to search for and change specified words or even text streams in a file. For example, the following command changes the first instance of the word "Windows" with "Linux" in each line of the file opsys, and writes the result to the file newopsys:

```
# sed 's/Windows/Linux' opsys > newopsys
```

However, this may not be enough. If there's more than one instance of "Windows" in a line in the opsys file, it does not change the second instance of that word. But you can fix this by adding a "global" suffix:

```
# sed 's/Windows/Linux/g' opsys > newopsys
```

The following example would make sure that all Samba shares configured with the **writable = yes** directive are reversed:

```
# sed 's/writable = yes/writable = no/g' /etc/samba/smb.conf > ~/smb.conf
```

Of course, you should then review the results in the /root/smb.conf file before overwriting the original /etc/samba/smb.conf file.

awk

The **awk** command, named for its developers (Aho, Weinberger, and Kernighan), is more of a database manipulation utility. It can identify lines with a keyword, and read out the text from a specified column in that line. A common example is with the /etc/passwd file. For example, the following command will read out the username of every user with a listing of "mike":

```
# awk '/mike/ {print $1}' /etc/passwd
```

Edit Text Files at the Console

The original version of the RHCSA objectives specified the use of the vim editor. Strictly speaking, it doesn't matter what text editor you use to edit text files. However, I believe that you need to know how to use the vim editor, and apparently some at Red Hat agree. The vim editor is short for vi, improved. When installed, you can also start the vim editor with the **vi** command. Hereafter, I refer to that text editor as vi.

I believe every administrator needs at least a basic knowledge of vi. While emacs may be more popular and flexible, vi may help you save a broken system. If you ever have to restore a critical configuration file using emergency boot media, vi may be the only editor that you'll have available.

While RHEL 6 also includes access to the more intuitive nano editor, a knowledge of vi commands can help you identify key sections of man pages and other text files more quickly. While RHEL rescue media supports more console-based editors, I describe vi here simply because it's the editor I know best.

You should know how to use the two basic modes of vi: command and insert. When you use vi to open a file, it opens in command mode. Some of the commands start insert mode. Opening a file is easy: just use the **vi *filename*** command. By default, this starts vi in command mode. An example of vi with the /etc/nsswitch.conf file is shown in Figure 3-4.

The following is only the briefest of introductions to the vi editor. For more information, there are a number of books available, as well as an extensive manual formatted as a HOWTO available from the Linux Documentation Project at www.tldp.org. Alternatively, a tutorial is available through the **vimtutor** command.

vi Command Mode

In command mode, you can do everything to a text file except edit it. The options in command mode are broad and varied, and they are the subject of a number of book-length texts. In summary, options in vi command mode fall into seven categories:

- **Open** To open a file in the vi editor from the command line interface, run the **vi *filename*** command.
- **Search** For a forward search, start with a backslash (/), followed by the search term. Remember, Linux is case sensitive, so if you're searching for

FIGURE 3-4

The vi editor with /etc/nsswitch.conf

```
passwd:     files
shadow:     files
group:      files

#hosts:     db files nisplus nis dns
hosts:      files dns

# Example - obey only what nisplus tells us...
#services:   nisplus [NOTFOUND=return] files
#networks:   nisplus [NOTFOUND=return] files
#protocols:  nisplus [NOTFOUND=return] files
#rpc:        nisplus [NOTFOUND=return] files
#ethers:     nisplus [NOTFOUND=return] files
#netmasks:   nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files

ethers:     files
netmasks:   files
networks:   files
protocols:  files
rpc:        files
```

"Michael" in /etc/passwd, use the **/Michael** (not **/michael**) command. For a reverse search, start with a question mark (?).

- **Write** To save your changes, use the **w** command. You can combine commands; for example, **:wq** writes the file and exits vi.
- **Close** To leave vi, use the **:q** command.
- **Abandon** If you want to abandon any changes, use the **:q!** command.
- **Edit** You can use a number of commands to edit files through vi, such as **x**, which deletes the currently highlighted character, **dw**, which deletes the currently highlighted word, and **dd**, which deletes the current line. Remember, **p** places text from a buffer, and **U** restores text from a previous change.
- **Insert** A number of commands allow you to start insert mode, including **i** to start inserting text at the current position of the editor, and **o** to open up a new line immediately below the current position of the cursor.

Basic Text Editing

In modern Linux systems, editing files with vi is easy. Just use the normal navigation keys (arrow keys, PAGE UP, and PAGE DOWN), and then one of the basic commands such as **i** or **o** to start vi's insert mode, and type your changes directly into the file. When you're finished with insert mode, press the ESC key to return to command mode. You can then save your changes, or abandon them and exit vi.

on the job

There are several specialized variations on the* vi *command. Three are* vipw, vigw, *and* visudo, *which edit /etc/passwd, /etc/group, and /etc/sudoers, respectively. The* vipw -s *and* vigr -s *commands edit the /etc/shadow and /etc/gshadow files.

EXERCISE 3-1

Using vi to Create a New User

In this exercise, you'll create a new user by editing the /etc/passwd file with the vi text editor. While there are other ways to create new Linux users, this exercise helps you verify your skills with vi and at the command line interface.

1. Open a Linux command line interface. Log in as the root user, and type the **vipw** command. This command uses the vi editor to open /etc/passwd.

2. Navigate to the end of the file. As you should already know, there are several ways to do this in command mode, including the DOWN ARROW key, the PAGE DOWN key, the **G** command, or even the K key.
3. Identify a line associated with a regular user. If you've just created a new user, it should be the last line in the file, with numbers of 500 and above. If a regular user does not yet exist, identify the first line, which should be associated with the root administrative user, with the number 0 in the third and fourth column.
4. Make one copy of this line. If you're already comfortable with vi, you should know that you can copy an entire line to the buffer with the **yy** command. This "yanks" the line into buffer. You can then restore or put that line as many times as desired with the **p** command.
5. Change the username, user ID, group ID, user comment, and home directory for the new user. For detailed information on each entry, see Chapter 8. For example, in the following illustration, this corresponds to tweedle, 501, 501, Tweedle Dee, and /home/tweedle. Make sure the username also corresponds to the home directory.

```
dbus:x:81:81:System message bus:/:/sbin/nologin
avahi-autoipd:x:170:170:Avahi IPv4LL Stack:/var/lib/avahi-autoipd:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/cache/rpcbind:/sbin/nologin
rtkit:x:499:499:RealtimeKit:/proc:/sbin/nologin
abrt:x:498:498::/etc/abrt:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
nslcd:x:65:55:LDAP Client User:/:/sbin/nologin
saslauth:x:497:495:"Saslauthd user":/var/empty/saslauth:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
pulse:x:496:494:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
oprofile:x:16:16:Special user account to be used by OProfile:/home/oprofile:/sbi
n/nologin
michael:x:500:500:Michael Jang:/home/michael:/bin/bash
tweedle:x:501:501:Tweedle Dee:/home/tweedle:/bin/bash
```

6. Return to command mode by pressing the ESC key. Save the file with the **:w** command, and then exit with the **:q** command. (You can combine the two

commands in vi; the next time you make a change and want to save and exit, run the **:wq** command.)

7. You should see the following message:

```
You have modified /etc/passwd.
You may need to modify /etc/shadow for consistency.
Please use the command 'vipw -s' to do so.
```

   That message can be ignored, as the next step adds appropriate information to the /etc/shadow file. However, you don't need to modify /etc/shadow directly.

8. As the root user, run the **passwd *newuser*** command. Assign the password of your choice to the new user. For this example, the new user is tweedle.
9. The process is not yet complete; every user needs a group. To that end, run the **vigr** command. Repeat the earlier steps that copied an appropriate line from near the end of the file. Note that group names and group ID numbers normally are identical to their usernames and user ID numbers.
10. All you need to change for the new entry is the group name and group ID number. Based on the information shown the previous illustration, that would be a group name of tweedle and a group number of 501.
11. Repeat the aforementioned **:wq** command to close vi and save the change. Actually, you'll get a message that suggests that the file is read only. You'd have to run the **:wq!** in this case to write to this "read only" file, overriding current settings.
12. Pay attention to the following message:

```
You have modified /etc/group.
You may need to modify /etc/gshadow for consistency.
Please use the command 'vigr -s' to do so.
```

13. As suggested, run the **vigr -s** command to open the /etc/gshadow file. You'll note that there's less information in this file. Once a copy is made of an appropriate line, all you'll need to do is change the group name.
14. Repeat the aforementioned **:wq!** command to close vi and save the change.
15. Additional steps are required to properly set up the new user, related to that user's home directory and standard files from the /etc/skel directory. For more information, see Chapter 8.

If You Don't Like vi

By default, when you run commands like **edquota** and **crontab**, associated quota and cron job configuration files are opened in the vi editor. If you absolutely hate vi, the default editor can be changed with the following command:

```
# export EDITOR=/bin/nano
```

To change the default editor for all users, add the preceding line to the /etc/environment configuration file. You don't absolutely have to use the vi editor to change /etc/environment; instead, the following appends the noted command to the end of the /etc/environment file:

```
# echo 'export EDITOR=/bin/nano' >> /etc/environment
```

As the nano editor is fairly intuitive, as shown in Figure 3-5, instructions will not be provided in this book. The full manual is available from www.nano-editor.org/dist/v2.1/nano.html.

Similar changes can be made if you prefer a different editor such as emacs, pico, or joe.

FIGURE 3-5

The nano editor with /etc/nsswitch.conf

```
  GNU nano 2.0.9          File: /etc/nsswitch.conf

#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.
#
# The entry '[NOTFOUND=return]' means that the search for an
# entry should stop if the search in the previous entry turned
# up nothing. Note that if the search failed due to some other reason
# (like no NIS server responding) then the search continues with the
# next entry.
#
# Valid entries include:
#
#       nisplus                 Use NIS+ (NIS version 3)
#       nis                     Use NIS (NIS version 2), also called YP
#       dns                     Use DNS (Domain Name Service)
#       files                   Use the local files
#       db                      Use the local database (.db) files
                          [ Read 63 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Edit Text Files in the GUI

No question, the Red Hat exams have become more friendly toward the GUI. The gedit text editor was even included for a short time in the RHCSA objectives. More traditional Linux administrators may have been horrified. (The gedit editor has since been deleted from the objectives.)

The gedit text editor is not installed by default. Fortunately, installation is easy with the **yum install gedit** command. Once installed, you can start it by clicking Applications | Accessories | gedit Text Editor. As it is an intuitive GUI text editor, its use is trivial. Don't obsess about editors; they are just tools on exams and in real life.

However, if you're editing configuration files on remote systems, it's possible that you won't have access to gedit on that system, especially if the GUI hasn't been installed there. Of course, you can install the GUI on any Red Hat system. But many administrators set up VMs without the GUI to save space and reduce security risks.

CERTIFICATION OBJECTIVE 3.04

Local Online Documentation

While there's no Internet access allowed during Red Hat exams, there is a lot of help available online, already installed on an RHEL 6 system. It starts with the man pages, which document the options and settings associated with most commands and many configuration files. It continues with the info documents. While fewer commands and files have such documents, when available, they do provide even more information.

exam Watch

When Red Hat says that it "may use applications during the exam that are not included in Red Hat Enterprise Linux for the purpose of evaluating candidate's abilities" with respect to documentation, I would not be shocked to find important exam information in the /usr/share/doc directory.

Many packages include extensive documentation in the /usr/share/doc directory. Just apply the **ls** command to that directory. Every subdirectory there includes information about the capabilities of each associated package. Of course, there's more.

When You Need Help

The first thing I do when I need help with a command is to run it by itself. If more information is required, the command prompts with a request for more information, including a variety of options. As an example, look at the output to the following command:

```
$ yum
```

If that approach doesn't work, generally some amount of help is available with the **-h** or the **--help** switches. Sometimes a mistake leads to some hints; the output to following command suggests legal switches to the **cd** command:

```
$ cd -h
bash: cd: -h: invalid option
cd: usage: cd [-L|-P] [dir]
```

FIGURE 3-6

Help with Process Management

```
-A all processes                    -C by command name
-N negate selection                 -G by real group ID (supports names)
-a all w/ tty except session leaders -U by real user ID (supports names)
-d all except session leaders       -g by session OR by effective group name
-e all processes                    -p by process ID
T  all processes on this terminal   -s processes in the sessions given
a  all w/ tty, including other users -t by tty
g  OBSOLETE -- DO NOT USE           -u by effective user ID (supports names)
r  only running processes           U  processes for specified users
x  processes w/o controlling ttys   t  by tty
*********** output format **********  *********** long options ***********
-o,o user-defined  -f full          --Group --User --pid --cols --ppid
-j,j job control   s  signal        --group --user --sid --rows --info
-O,O preloaded -o  v  virtual memory --cumulative --format --deselect
-l,l long          u  user-oriented --sort --tty --forest --version
-F   extra full    X  registers     --heading --no-heading --context
                    ********* misc options *********
-V,V  show version      L  list format codes  f  ASCII art forest
-m,m,-L,-T,H  threads   S  children in sum    -y change -l format
-M,Z  security data     c  true command name  -c scheduling class
-w,w  wide output       n  numeric WCHAN,UID  -H process hierarchy
michael@Maui:~$ ls bookRHCE6/Chapter3/
56503.doc  Ch3Lab3          ch3.zip     F03-01.tif  F03-03.tif  fifthedition
Backup     Ch3Lab3testfile  F03-01.png  F03-02.tif  F03-04.tif
michael@Maui:~$
```

Sometimes the **-h** switch is more helpful; take a look at the output to the **fdisk -h** command. But the **-h** switch doesn't always work; sometimes the **--help** switch is more helpful. Look at Figure 3-6 as an example, which displays the output to the **ps --help** command.

A Variety of man Pages

Few people can remember every switch to every command. That's one reason why command documentation is so important. Most Linux commands are documented in a format known as the man page. If you run the **man** command by itself, RHEL returns the following message:

```
What manual page do you want?
```

For example, say you need to set up a physical volume but have forgotten the switches associated with the **lvexpand** command. To browse the man page for that command, run **man lvexpand**. As with many other commands, there's an EXAMPLES section, like that shown in Figure 3-7. If you've run the **lvexpand** command before, that section may help jog your memory.

FIGURE 3-7

Examples from the lvexpand man page

```
       -r, --resizefs
              Resize  underlying  filesystem  together with the logical volume
              using fsadm(8).

Examples
       "lvextend -L +54 /dev/vg01/lvol10 /dev/sdk3" tries to extend  the  size
       of  that  logical volume by 54MB on physical volume /dev/sdk3.  This is
       only possible if /dev/sdk3 is a member of volume group vg01  and  there
       are enough free physical extents in it.

       "lvextend  /dev/vg01/lvol01 /dev/sdk3" tries to extend the size of that
       logical  volume  by  the  amount  of  free  space  on  physical  volume
       /dev/sdk3.   This is equivalent to specifying "-l +100%PVS" on the com-
       mand line.

       "lvextend -L+16M vg01/lvol01 /dev/sda:8-9 /dev/sdb:8-9"
       tries to extend a logical volume "vg01/lvol01" by 16MB  using  physical
       extents /dev/sda:8-9 and /dev/sdb:8-9 for allocation of extents.

SEE ALSO
       fsadm(8),  lvm(8), lvcreate(8), lvconvert(8), lvreduce(8), lvresize(8),
       lvchange(8)

:
```

Such man pages are available for most configuration files and commands. However, there may be more. So what if you're not sure about the name of the man page? In that case, the **whatis** and **apropos** commands can help. For example, to find the man pages with "nfs" in the title, run the following command:

```
# whatis nfs
```

If you want to find the man pages with nfs in the description, the following command can identify related commands.

```
# apropos nfs
```

However, if you've just installed a service such as Samba, associated with the Linux implementation of Microsoft networking, commands like **whatis smb.conf** and **apropos smbpasswd** probably won't provide any information. These commands work from a database in the /var/cache/man directory. You can update that database with the makewhatis.cron job in the /etc/cron.daily directory. Since that script is already executable, the following command updates the database of man pages:

```
# /etc/cron.daily/makewhatis.cron
```

If you encounter a situation, such as during a Red Hat exam, where the associated man page is not installed, there are at least three possible reasons. The associated functional software package may not be installed. The RPM package named man-pages may also not be installed. In some cases, there is a package specifically dedicated to documentation that must be installed separately. For example, there's a system-config-users-doc package that includes GUI-based documentation for the User Manager configuration tool. There's a separate httpd-manual package installed separately from the Apache Web server.

In some cases, there are multiple man pages available. Take a look at the following output to the **whatis smbpasswd** command:

```
smbpasswd             (5)   - The Samba encrypted password file
smbpasswd             (8)   - change a user's SMB password
```

The numbers (5) and (8) are associated with different sections of man pages. If you're interested in details, they're shown in the output to the **man man** command. The man page shown by default is the command. In this case, if you want the man page for the encrypted password file, run the following command:

```
$ man 5 smbpasswd
```

To exit from a man page, press **q**.

The info Manuals

The list of available info manuals is somewhat limited. For a full list, run the **ls /usr/share/info** command. When an info manual is not available, a request defaults to the associated man page.

To learn more about the bash shell, run the **info bash** command. As shown in Figure 3-8, info manuals are organized into sections. To access a section, move the cursor to the asterisked entry and press ENTER.

To exit from an info page, press **q**.

Detailed Documentation in /usr/share/doc

The list of documentation available in the /usr/share/doc directory seems impressive. But the quality of the documentation depends on the work of its developers. The subdirectories include the name and version number of the installed package. Some of these subdirectories include just one file, normally named COPYING, which specifies the license under which the given software was released. For example, most of the system-config-* packages include a copy of the GNU GPL in the COPYING file in the associated /usr/share/doc directory.

FIGURE 3-8

A sample info manual

```
* Menu:

* Introduction::               An introduction to the shell.
* Definitions::                Some definitions used in the rest of this
                               manual.
* Basic Shell Features::       The shell "building blocks".
* Shell Builtin Commands::     Commands that are a part of the shell.
* Shell Variables::            Variables used or set by Bash.
* Bash Features::              Features found only in Bash.
* Job Control::                What job control is and how Bash allows you
                               to use it.
* Command Line Editing::       Chapter describing the command line
                               editing features.
* Using History Interactively:: Command History Expansion
* Installing Bash::            How to build and install Bash on your system.
* Reporting Bugs::             How to report bugs in Bash.
* Major Differences From The Bourne Shell::     A terse list of the differences
                                                between Bash and historical
                                                versions of /bin/sh.
* GNU Free Documentation License::      Copying and sharing this documentation.
* Indexes::                    Various indexes for this manual.

--zz-Info: (bash.info.gz)Top, 44 lines --Bot-----------------------------------
```

Sometimes, the documentation directory includes useful examples. For example, the sudo-*/ subdirectory includes sample configuration files and directives for administrative control, which can be helpful when configuring administrators with different privileges.

Sometimes the documentation includes entire manuals in HTML format. For an example, take a look at the rsyslog-*/ subdirectory, which includes an entire online manual for the logging daemon server discussed in Chapters 9 and 17.

CERTIFICATION OBJECTIVE 3.05

A Networking Primer

TCP/IP is a series of protocols organized in layers, known as a protocol suite. It was developed for Unix and eventually adopted as the standard for communication on the Internet. With IP addresses, it can help you organize a network. There are a number of TCP/IP tools and configurations that can help you manage a network.

As with the previous sections in this chapter, the statements here are oversimplifications. So if you find this section overwhelming and/or incomplete, read the references cited in Chapter 1. Linux is built for networking, and there is no practical way to pass either any Red Hat exam unless you understand networking in some detail.

While the focus of current networks is still on IP version 4 addressing, some organizations have mandated a move toward IP version 6 (IPv6) networks. Even though the Internet has run out of new public IPv4 addresses, hardware that supports the routing of IPv6 networks is still somewhat rare. Hopefully, that will change during the life of this book.

IP Version 4 Numbers and Address Classes

Every computer that communicates on a network needs its own IP address. Some addresses are assigned permanently to a particular computer; these are known as *static* addresses. Others are leased from a DHCP server for a limited amount of time; these are also known as *dynamic* IP addresses.

IPv4 addresses are organized into five different classes, as shown in Table 3-2. The academics among you may note that this table differs slightly from the official addresses in each IPv4 class as specified in RFC 1518 from the Internet Engineering

Task Force (www.ietf.org). The *assignable* address range includes those IP addresses that can be assigned to a specific computer on a network.

In addition, there are a number of private IP addresses that are not to be assigned to any computer that is directly connected to the Internet. They are associated with network addresses 10.0.0.0, 172.168.0.0, and 192.168.0.0 through 192.168.255.0.

Basic IP Version 6 Addressing

Network experts have been predicting the demise of IPv4 for years. It is true, there aren't enough IPv4 addresses for the Internet. However, with the help of private IP address blocks, users on enterprise-level networks don't need that many public IP addresses.

Nevertheless, there will be a time where IPv6 addresses are the norm. First, to compare, IPv4 addresses have 32 bits and are set up in octets in dotted decimal notation. IPv6 addresses have 128 bits and are set up in hexadecimal notation, also known as base 16. In other words, the "numbers" in an IPv6 address may include the following:

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f
```

An IPv6 address is normally organized in eight groups of four hexadecimal numbers each, and it may look like 4abe:03e2:c132:69fa:0000:0000:c0b8:2148. The current IPv6 address of the local system is shown in the output to the **ifconfig** command.

With 128 bits, IPv6 addresses can be divided into a number of categories. First, there are three relevant address formats.

- **Unicast** A unicast address is associated with a single network adapter. Routable unicast addresses include a 48-bit network prefix, a 16-bit subnet

TABLE 3-2 IP Address Classes

Class	Assignable Address Range	Note
A	1.1.1.1–126.255.255.254	Allows networks of up to 16 million computers
B	128.0.0.1–191.255.255.254	Allows networks of up to 65,000 computers
C	192.0.0.1–223.255.255.254	Allows networks of up to 254 computers
D	224.0.0.1–239.255.255.254	Reserved for multicasts
E	240.0.0.1–255.255.255.254	Reserved for experimental use

identifier, and a 64-bit interface identifier associated with a network adapter hardware address. Link-local unicast addresses include a 10-bit prefix, 54 zeros, and the same 64-bit interface identifier. Link-local unicast addresses are not routable.

- **Multicast** A multicast address is used to send a message to multiple network adapters simultaneously. The organization of a multicast address varies.
- **Anycast** An anycast address is used to send a message to one of several optional network adapters. It's useful for systems with multiple backups, such as a group of Web servers. Anycast addresses have the same basic organization as a unicast address.

With that diversity of address formats, IPv4-style broadcast addresses aren't used. Instead, IPv6 addressing uses multicast addressing for the purpose. IPv6 addresses are also organized in a number of different ranges, as described in Table 3-3. The default IPv6 address is sometimes also listed as ::/128.

How to Define a Network with IP Addresses

Three key IP addresses define a network: the network address, the broadcast address, and the subnet mask. The network address is always the first IP address in a range; the broadcast address is always the last address in the same range. The subnet mask helps your computer define the difference between the two addresses. You can assign IP addresses between the network and broadcast addresses (not including these addresses) to any computer on the network.

on the job

A subnet mask is also known as a network mask or netmask. An example of an IPv4 netmask is 255.255.255.0. An example of an IPv6 netmask is /64.

TABLE 3-3 IP Address Classes

Address	Description
::1	Loopback address
::	Default address
::ffff:0000:0000	IPv4 mapped IPv6 addresses in the last 8 zeros
fe80::	Link-local addresses; no routing between networks
fec0::	Site-local addresses, for a single network
ff::	Multicast addresses
2000::	Global unicast addressees are routable

As an example, let's define the range of addresses for a private network. Start with the private network address 192.168.122.0. Use the standard subnet mask for a class C network, 255.255.255.0. Based on these two addresses, the broadcast address is 192.168.122.255, and the range of IP addresses that you can assign on that particular network is 192.168.122.1 through 192.168.122.254. That subnet mask is also defined by the number of associated bits, 24. In other words, the given network can be represented by 192.168.122.0/24. That's also known as Classless Inter-Domain Routing (CIDR) notation.

IPv6 networks use a similar concept for netmasks, which are always expressed in CIDR notation. For example, even point-to-point networks have a netmask of /64. That allows 64 bits to be used for the 48-bit hardware address. Remaining addresses can be allocated to specific network cards.

The standard IPv6 network has a 48-bit netmask. That supports the configuration of 16 bit subnets. The remaining 64 bits are still used in part for network cards, as described previously for point-to-point networks.

Related to networking and netmasks is the concept of the gateway. It's an IP address that defines the junction between the local network and an external network. While that gateway IP address is part of the local network, it's attached to a system or a router with an IP address on a different network such as the public Internet. The gateway IP address is normally configured in the routing table for the local system, as defined by the **route** or **netstat -r** command described in the following section.

If this is confusing to you in any way, please refer to the IP Sub-Networking Mini-HOWTO and the Linux IPv6 HOWTO of the Linux Documentation Project at www.tldp.org.

Tools, Commands, and Gateways

There are a substantial number of tools available to manage the TCP/IP protocol suite on your Linux computer. Four of the more important network management commands are **ping**, **ifconfig**, **arp**, **netstat -r**, and **route**. There's also an IPv6-specific version of the **ping** command, **ping6**. The **dhclient** command is frequently used to automate the configuration that can be done with some of these commands.

But these are just the commands. In the next section, you'll examine the Red Hat files that determine the commands that are called upon to configure networks automatically during the boot process. Such commands are governed by the main network configuration service script, /etc/init.d/network. They can also be manually called with commands such as **ifup** and **ifdown**.

ping and ping6

The **ping** command allows you to test connectivity. It can be applied locally, within a network, and across networks on the Internet. For the purpose of this section, assume your IP address is 192.168.122.50, and the gateway address on the local network is 192.168.122.1. If you're having problems connecting to a network, try the following **ping** commands in order. The first step is to test the integrity of TCP/IP on your computer:

```
# ping 127.0.0.1
```

Normally, **ping** works continuously on Linux; you'll need to press CTRL-C to stop this command. If you need to verify a proper connection to a LAN, **ping** the IP address of the local network card:

```
# ping 192.168.122.50
```

If that works, **ping** the address of another computer on your network. Then start tracing the route to the Internet. **ping** the address for the network gateway, in this case, 192.168.122.1. If possible, **ping** the address of the network's connection to the Internet, which would be on the other side of the gateway. It may be the public IP address of the LAN on the Internet. And finally, **ping** the address of a computer that you know is *active* on the Internet.

You can substitute hostnames such as www.google.com for an IP address. If the hostname doesn't work, there is likely a problem with the database of hostnames and IP addresses, more commonly known as a Domain Name Service (DNS), Berkeley Internet Name Domain (BIND), or nameserver. It could also indicate a problem with the /etc/hosts configuration file.

In contrast, the **ping6** command works in almost the same way. The exception on Red Hat systems is that you need to specify the network adapter. For example, the following command pings the noted IPv6 network through the virtual virbr0 adapter:

```
# ping6 -I virbr0 fe80::5652:ff:fe39:24d8
```

If you've configured global IPv6 addresses and have set up routing to the Internet, Google has a test IPv6 URL at ipv6.google.com.

Review Current Network Adapters with ifconfig

The **ifconfig** command can display the current state of active network adapters. It also can be used to assign network addresses and more. Run the **ifconfig** command

by itself to review the active network adapters on the local system. If there seems to be a missing network adapter, try the **ifconfig -a** command, which displays the current configuration of all network adapters, whether or not they're currently active.

The **ifconfig eth0** command shown here reflects the current configuration of the first Ethernet network adapter:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:50:56:40:1E:6A
          inet addr:192.168.122.50  Bcast:192.168.122.255  Mask:255.255.255.0
          inet6 addr: fe80::2e0:4cff:fee3:d106/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11253 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1304 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2092656 (1.9 Mb)  TX bytes:161329 (157.5 Kb)
```

Configure a Network Adapter with ifconfig

You can also use **ifconfig** to assign IP address information as well. For example, the following command assigns the noted IP address and network mask to the eth0 network adapter:

```
# ifconfig eth0 192.168.122.150 netmask 255.255.255.0
```

The first parameter, **eth0**, tells you which interface is being configured. The next argument, **192.168.122.150**, specifies the new IP address being assigned to this interface. To make sure the change worked, run the **ifconfig eth0** command again to view its current settings.

With the right switch, the **ifconfig** command can modify a number of other settings for a selected network adapter. Some of these switches are shown in Table 3-4.

Activate and Deactivate Network Adapters

It's possible to use the **ifconfig** command to activate and deactivate network adapters. For example, the following commands deactivate and reactivate the first Ethernet adapter:

```
# ifconfig eth0 down
# ifconfig eth0 up
```

TABLE 3-4 ifconfig Switches

Parameter	Description
up	Activates the specified adapter.
down	Deactivates the specified adapter.
netmask ***address***	Assigns the *address* subnet mask.
broadcast ***address***	Assigns the *address* as the broadcast address. Rarely required, since the default broadcast address is standard for most current networks.
metric ***N***	Allows you to set a metric value of *N* for the routing table associated with the network adapter.
mtu ***N***	Sets the maximum transmission unit as *N*, in bytes.
-arp	Deactivates the Address Resolution Protocol (ARP), which collects network adapter hardware addresses.
promisc	Activates promiscuous mode. This allows the network adapter to read all packets to all hosts on the LAN. Can be used to analyze the network for problems or to try to decipher messages between other users.
-promisc	Deactivates promiscuous mode.

However, a couple of more intuitive scripts are designed to control network adapters: **ifup** and **ifdown**. Unlike the **ifconfig** command, they call appropriate configuration files and scripts in the /etc/sysconfig/network-scripts directory, for details on how a network adapter is to be activated and deactivated.

For example, the **ifup eth0** command brings up the first Ethernet network adapter based on the ifcfg-eth0 configuration file and the ifcfg-eth script in the /etc/sysconfig/network-scripts directory. If you've configured the network adapter with the Network Connections tool described later in this chapter, the filename in the /etc/sysconfig/network-scripts directory may be something like ifcfg-System_eth0.

arp as a Diagnostic Tool

The ARP protocol associates the hardware address of a network adapter with an IP address. The **arp** command displays a table of hardware and IP addresses on the local computer. The **arp** command can help detect problems such as duplicate addresses on the network. Such problems may happen with improperly cloned systems. If needed, the **arp** command can be used to set or modify hardware routing tables. As hardware addresses are not routable, an arp table should be limited to the local network. Here's a sample **arp** command, showing all **arp** entries in the local database:

```
# arp
Address            HWtype  HWaddress           Flags Mask        Iface
192.168.122.150      ether   52:A5:CB:54:52:A2   C                 eth0
192.168.100.100      ether   00:A0:C5:E2:49:02   C                 eth0
192.168.122.1        ether   00:0E:2E:6D:9E:67   C                 eth0
```

If the ARP table is empty, no recent connections exist to other systems on the local network. The Address column lists known IP addresses on the LAN. The HWtype column shows the hardware type of the adapter, while the HWaddress column shows the hardware address of the adapter.

Routing Tables with netstat -r and route

The **netstat** command is versatile; it can help you see the channels available for network connections, interface statistics, and more. One important version of this command, **netstat -r**, displays routing tables that can tell you if the system knows where to send a message. It's functionally equivalent to the **route** command. When run on a system, it's frequently run with the **-n** switch, to display addresses in numeric format.

The routing table for the local system normally includes a reference to the local gateway address, coupled with the default route. For example, look at the following output to the **route -n** command:

```
Kernel IP routing table
Destination   Gateway       Genmask       Flags Metric Ref Use Iface
192.168.122.0 0.0.0.0       255.255.255.0 U     0      0     0 eth0
0.0.0.0       192.168.122.1 0.0.0.0       UG    0      0     0 eth0
```

The **netstat -nr** command should display the same table. For this routing table, the gateway IP address is 192.168.122.1. It's the gateway to the destination IP address of 0.0.0.0, which is the default IP address. In other words, network transmission to anything other than the 192.168.122.0 network is sent to the gateway address. The system at the gateway address, usually a router, is responsible for forwarding that message to an external network.

If the destination is on the LAN, no gateway is required, so an asterisk (or 0.0.0.0) is shown in this column. The Genmask column lists the network mask. Networks look for a route appropriate to the destination IP address. The IP address is compared against the destination networks, in order. When the IP address is found to be part of one of these networks, it's sent in that direction. If there is a gateway address, it's sent to the computer with that gateway. The Flags column describes how this is done. Flag descriptions are listed in Table 3-5.

TABLE 3-5

The netstat Flag Indicates the Route

Flag	Description
G	The route uses a gateway.
U	The network adapter, listed in the Iface column, is up.
H	Only a single host can be reached via this route.
D	This entry was created by an ICMP redirect message.
M	This entry was modified by an ICMP redirect message.

In contrast, while an IPv6 routing table is more complex, the principles are the same. In other words, the IPv6 gateway address is associated with the default IPv6 route, symbolized by the ::/128 address. The same **route** and **netstat** commands can be used for IPv6 routing tables, when coupled with the **-A inet6** switch.

Dynamically Configure IP Addresses with dhclient

While the name of the command has changed from time to time, the functionality has remained the same. Ever since Dynamic Host Configuration Protocol (DHCP) servers were created to ration IPv4 addresses, commands have been needed by clients to call upon the services of that server. At this time, the key command is **dhclient**. When used with the device name of a network card, it calls upon a DHCP server for an IP address and more. In fact, a command like the following may call on that DHCP server for a number of parameters.

```
# dhclient eth0
```

Generally, the network options that are configured through a DHCP server include the IP address, the network mask, the gateway address for access to external networks, and the IP address of any DNS servers for that network.

In other words, the **dhclient eth0** command not only assigns IP address information in the way done with the **ifconfig** command described earlier, but also it sets up the default route for the routing table shown with the **route -n** command. In addition, it adds the IP address of the DNS server to the /etc/resolv.conf configuration file.

CERTIFICATION OBJECTIVE 3.06

Network Configuration and Troubleshooting

Now that you've reviewed the basics of IP addressing and associated commands, it's time to look at the associated configuration files. These configuration files determine whether networking is started during the boot process. If started, these files also determine whether addresses and routes are configured statically as documented, or dynamically with the help of commands like **dhclient**.

Basic network configuration only confirms that systems can communicate through their IP addresses. But that is not enough. Whether you're pointing to systems such as server1.example.com or URLs such as www.mheducation.com, network configuration is not enough if the hostname (or FQDN) configuration is not working.

on the job

The most common cause of network problems is physical. This section assumes you've checked all network connections. On a VM, that means making sure the virtual network card wasn't accidentally deleted on the VM or on the physical host.

Network Configuration Files

If there's trouble with a network configuration, one thing to check is the current status of the network. To do so, run the following command:

```
# /etc/init.d/network status
```

The command should list configured and active devices. If a key device such as eth0 is not listed as active, that explains why the network seems to be down. Key configuration files start with /etc/sysconfig/network. They continue with files in the /etc/sysconfig/network-scripts directory.

Sometimes mistakes happen. If you've deactivated an adapter or just lost a wireless connection, one simple solution may be to restart networking. The following command restarts networking with current configuration files.

```
# /etc/init.d/network restart
```

If a simple restart of networking services doesn't work, then it's time to get into the files.

exam Watch

Services such as networking may be configured to start during the boot process, as discussed in Chapter 5.

/etc/sysconfig/network

If you run the **ifconfig** command and see no output, that means all network devices are currently inactive. If you run the **ifconfig -a** command and don't see an **UP** in the output to any configured network device, that confirms the inactivity. The first thing to check in that case is the contents of the /etc/sysconfig/network configuration file. It's a pretty simple file. In general, you should see something similar to the following in that file:

```
NETWORKING=yes
HOSTNAME=server1.example.com
```

If **NETWORKING=no**, then the /etc/init.d/network script doesn't activate any network devices. The one other issue that may prevent networking from starting is the status of the script. Run the **chkconfig --list network** command. The output should look like:

```
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

If the settings next to runlevels 3 and 5 are off, that's a problem. To make sure a service is active in appropriate runlevels, run the **chkconfig network on** command. For more information on **chkconfig**, see Chapter 5.

If IPv6 networking is active on a system, you'll see the following directive in the file:

```
NETWORKING_IPV6=yes
```

One other network-related directive that may appear is **GATEWAY**, if it's the same IP address for all network devices. Otherwise, that configuration is supported either by the **dhclient** command or set up in the IP address information for a specific network device, in the /etc/sysconfig/network-scripts directory.

/etc/sysconfig/network-scripts/ifcfg-lo

Speaking of the /etc/sysconfig/network-scripts directory, perhaps the foundation of networking is the loopback address. That information is configured in the ifcfg-lo file in that directory. The contents of the file can help you understand how files in that directory are used for network devices. By default, you should see the following entries in that file, starting with the name of the loopback device:

```
DEVICE=lo
```

It's followed by the IP address (**IPADDR**), network mask (**NETMASK**), the network IP address (**NETWORK**), along with the corresponding broadcast address (**BROADCAST**).

```
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
BROADCAST=127.255.255.255
```

The next entries specify whether the device is activated during the boot process, and the common name of the device.

```
ONBOOT=yes
NAME=loopback
```

/etc/sysconfig/network-scripts/ifcfg-eth0

What you see in the ifcfg-eth0 file depends on how that first Ethernet network adapter was configured. For example, look at the situation where networking was configured only for the purposes of installation. If you did not configure networking when configuring the hostname during the GUI installation process, networking will not be configured on the system. In that case, the ifcfg-eth0 file would contain the following directives, starting with the name of the device, along with the hardware address:

```
DEVICE="eth0"
HWADDR="F0:DE:F3:06:C6:DB"
```

By default, RHEL 6 uses a service known as the Network Manager. If a network card is controlled by that service, the following directive would be set to yes:

```
NM_CONTROLLED="yes"
```

The Network Manager is a service; to make sure it's running, execute the **/etc/init.d/NetworkManager start** command. Of course, if networking were not configured during the installation process, there's no reason for it to be activated during the boot process:

```
ONBOOT="no"
```

The alternative to the Network Manager service is to configure it directly. For that purpose, the configuration file shown in Figure 3-9 provides a guide.

FIGURE 3-9

A static configuration not controlled by Network Manager

```
DEVICE="eth0"
BOOTPROTO="static"
DNS1="192.168.122.1"
GATEWAY="192.168.122.1"
HWADDR="52:54:00:5A:97:F6"
IPADDR="192.168.122.50"
NETMASK="255.255.255.0"
NM_CONTROLLED="yes"
ONBOOT="yes"
~
~
~
~
```

Of course, if you prefer to use a DHCP server, that static network address information would be omitted, and the following directive would be changed:

```
BOOTPROTO=dhcp
```

Shortly, you'll see how to use Network Manager's Network Connections tool to modify the configuration of a network device. But first, for a different perspective, review Figure 3-10, which illustrates the configuration for a wireless network card on my RHEL 6 laptop system.

FIGURE 3-10

A wireless network configuration

```
ESSID=wifi1
MODE=Managed
KEY_MGMT=WPA-PSK
TYPE=Wireless
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME=wifi1
UUID=b44e0567-d13a-4af1-b30a-38745f99ee91
ONBOOT=yes
IPADDR=192.168.0.200
PREFIX=24
GATEWAY=192.168.0.1
DNS1=192.168.0.1
LAST_CONNECT=1290340839
DEVICE=wlan0
USERCTL=no
~
~
~
~
~
```

Other /etc/sysconfig/network-scripts Files

Most of the files in the /etc/sysconfig/network-scripts directory are actually scripts. In other words, they are executable files based on a series of text commands. Most of those scripts are based on the **ifup** and **ifdown** commands, customized for network device type. If there's a special route to be configured, the configuration settings get their own special file in this directory, with a name like route-eth0. That special route would specify the gateway to a remote network address / network mask pair. One example based on the systems described in Chapter 1 might include the following directives:

```
ADDRESS0=192.168.100.100
NETMASK0=255.255.255.0
GATEWAY0=192.168.122.1
```

Network Configuration Tools

Red Hat includes two tools that can be used to configure network devices in RHEL 6. The first is the console network configuration tool. You can start it from the command line with the **system-config-network** command. The second is the Network Connections tool that you can start from a GUI command line with the **nm-connection-editor** command.

The Network Manager also includes another tool to display the current status of network devices. The output is somewhat similar to the output to the **ifconfig** command.

The Console Network Configuration Tool

As suggested by the name, you can start this tool from a command line console. Just run the **system-config-network** command. With a console tool, you'd need to press TAB to switch between options, and the spacebar or the ENTER key to select the highlighted option. Since this tool is not the Network Manager, you'll have to deactivate the associated device to put the configuration customized with this tool into effect. Exercise 3-2 illustrates this process.

Press TAB until Quit is highlighted and press ENTER. For now, make a backup of the ifcfg-eth0 file from the /etc/sysconfig/network-scripts directory. Based on the **diff** command, Figure 3-11 compares the contents of an eth0 card that uses the DHCP protocol, as configured during the installation process, with a card that uses static IP addressing, configured with the system-config-network tool.

The directives shown in Figure 3-11 are described in Table 3-6.

FIGURE 3-11 The differences between static and dynamic network configuration

```
[root@Maui ~]# diff ifcfg-eth0 /etc/sysconfig/network-scripts/ifcfg-eth0
1,4c1,17
< DEVICE="eth0"
< HWADDR="F0:DE:F1:06:C6:DC"
< NM_CONTROLLED="yes"
< ONBOOT="no"
---
> DEVICE=eth0
> NM_CONTROLLED=yes
> ONBOOT=no
> BOOTPROTO=none
> NETMASK=255.255.255.0
> TYPE=Ethernet
> IPV6INIT=no
> USERCTL=no
> HWADDR=f0:de:f1:06:c6:dc
> DEFROUTE=yes
> PEERROUTES=yes
> IPV4_FAILURE_FATAL=yes
> NAME="System eth0"
> UUID=5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
> IPADDR=192.168.122.60
> GATEWAY=192.168.122.1
> DNS1=192.168.122.1
[root@Maui ~]#
```

TABLE 3-6 Network Configuration Directives in the /etc/sysconfig/network-scripts Directory

Directive	Description
DEVICE	Network device; eth0 is the first Ethernet network card
HWADDR	Hardware address for the network card
NM_CONTROLLED	Binary directive (yes or no) that specifies whether the card is controlled by the NetworkManager service
ONBOOT	Binary directive that specifies whether the network device is started during the boot process
BOOTPROTO	May be set to none for static configuration, DHCP to acquire IP addresses from a DHCP server
NETMASK	Network mask based on a static IP address configuration
TYPE	Network type, typically Ethernet
IPV6INIT	Binary directive that specifies the use of IPv6 addressing
USERCTL	Binary directive for user control of devices

TABLE 3-6 Network Configuration Directives in the /etc/sysconfig/network-scripts Directory (*continued*)

Directive	Description
DEFROUTE	Binary directive for using the default route, defined by route -n
PEERROUTES	Binary directive allowing the use of defined routes
IPV4_FAILURE_FATAL	Binary directive supporting network failure if there's an error
NAME	Name of the Ethernet device; if present, the device becomes the value, such as System_eth0
UUID	Universal Unique Identifier for the device
IPADDR	Static IP address
GATEWAY	IP address of the default gateway

EXERCISE 3-2

Configure a Network Card

In this exercise, you'll configure the first Ethernet network card with the console-based Network Configuration tool. All you need is a command line interface. It doesn't matter whether the command line is in the GUI. If you're not logged in as the root user, you'll be prompted for the root administrative password. To configure a network card, take the following steps:

1. Back up a copy of the current configuration file for the first Ethernet card. Normally, it's ifcfg-eth0 in the /etc/sysconfig/network-scripts directory. For other cards, such as eth1, substitute accordingly. (Hint: use the **cp** and not the **mv** command.)
2. Run the **system-config-network** command.
3. In the Select Action menu that appears, Device Configuration should be highlighted. If necessary, press the TAB key until it is. Then press ENTER.
4. In the Select A Device screen that appears, the first Ethernet network card should be highlighted. When it is, press ENTER.
5. In the Network Configuration window shown in here, the Use DHCP option may be selected. If so, highlight it and press the SPACEBAR to deselect it.

6. Enter the IP address information for the system. The settings shown in the window are based on the settings described in Chapter 1 for the server1.example.com system. When complete, highlight OK and press ENTER.
7. You're taken back to the Select A Device screen. Make sure Save is highlighted and press ENTER.
8. You're taken back to the Select Action screen. Make sure Save&Quit is highlighted and press ENTER.
9. Deactivate and then reactivate the first Ethernet card with the **ifdown eth0** and **ifup eth0** commands, and check the result with the **ifconfig eth0** and **route -n** commands. The configuration of the network card and the associated routing table should reflect the new configuration.
10. To restore the original configuration, restore the ifcfg-eth0 file to the /etc/sysconfig/network-scripts directory and restart the network with the **/etc/init.d/network restart** command.

The Network Manager Network Connections Tool

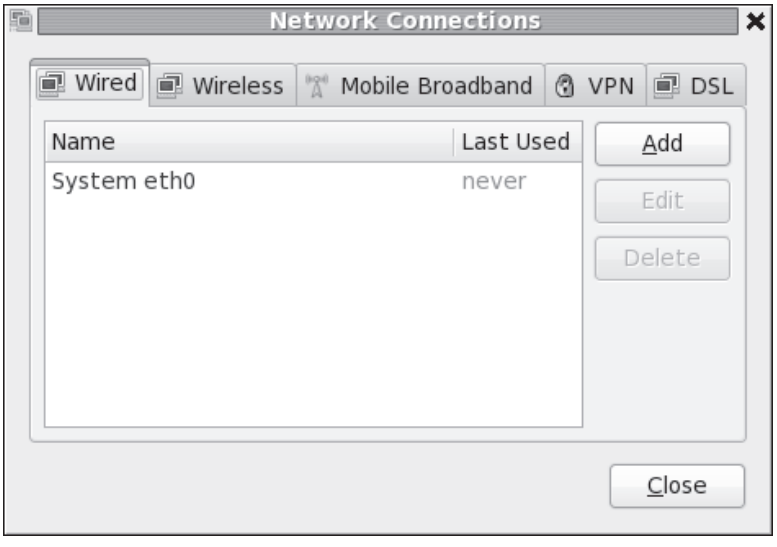
Now you'll work with the new default network management tool for RHEL 6, the Network Connections tool. With the number of users on multiple network connections, the Network Manager is designed to make that switching between say a wireless and an Ethernet connection as seamless as possible. But that's something more applicable to portable systems, as opposed to servers. For our purposes, all you need to know is how to configure a network card with that tool.

It's not really new, as it's been in use on the Fedora Linux test bed for several years. It only runs in the GUI. To start it, you can run the **nm-connection-editor** command or click System | Preferences | Network Connections. It opens the Network Connections tool shown in Figure 3-12.

As you can see from the figure, the tool lists the detected first Ethernet network card, even though it hasn't been used before. The other tabs support the configuration of other types of network connections, including wireless, mobile broadband cards such as those used to connect to 3G and 4G networks, virtual private network (VPN) connections, and Digital Subscriber Line (DSL) connections. On a regular server, the focus is on reliable connections, and that is still based on a standard wired Ethernet device.

FIGURE 3-12

The Network Manager Network Connections tool



Highlight the first Ethernet device (eth0) and click Edit. It'll open the Editing System window shown in Figure 3-13. Note how the window includes the name of the Ethernet device and the hostname of the system. That means I've accessed the window remotely, with the ssh -X command described in Chapter 2. (If access is local, the hostname won't be shown.)

Click the IPv4 settings tab. Unless previously configured, it assumes that network card will receive configuration settings from a DHCP server.

Click the Method drop-down text box. While it supports the configuration of a network card in several different ways, the only one of interest in this case is Manual. Select that option, and the Addresses section of the window should no longer be blanked out. Now add the IP address information for the system. Based on the server1.example.com system described in Chapter 1, the appropriate options include the following:

- **IP Address** 192.168.122.50

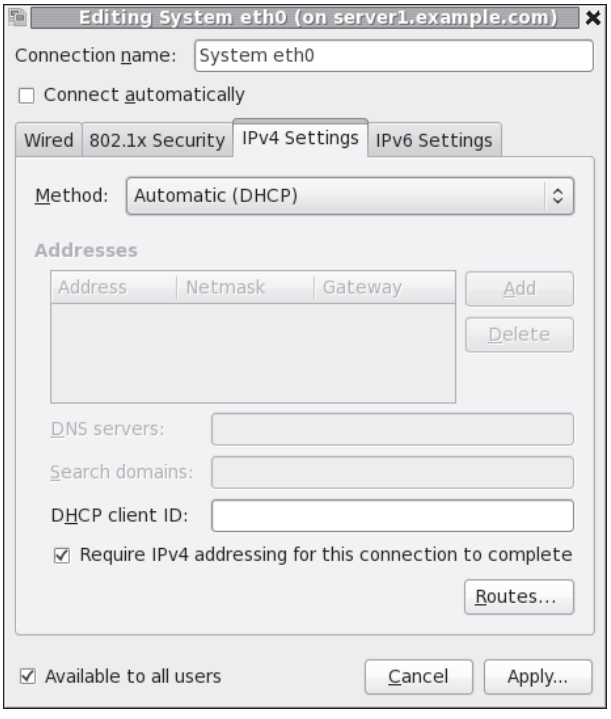


FIGURE 3-13

Editing an Ethernet connection in the Network Settings tool

- **Network Mask** 255.255.255.0 (24 in CIDR notation is an acceptable equivalent in this field)
- **Gateway Address** 192.168.122.1
- **DNS Server** 192.168.122.1
- **Search Domains** No entry required
- **Require IPv4 Addressing For This Connection To Complete** Supports IPv4 addressing
- **Available To All Users** If deselected, access is disabled for all users

If properly entered, the configuration associated with the first Ethernet card is entitled with the Connection Name listed in Figure 3-13. For that configuration, the settings are saved in the ifcfg-System_eth0 file in the /etc/sysconfig/network-scripts directory.

Hostname Configuration Files

RHEL 6 includes at least four hostname configuration files of interest: /etc/sysconfig/network, /etc/nsswitch.conf, /etc/hosts, and /etc/resolv.conf. These four files, taken together, contain the local hostname, the local database of hostnames and IP addresses, the IP address of a DNS server, and the order in which these databases are considered.

/etc/nsswitch.conf

The /etc/nsswitch.conf specifies database search priorities for everything from authentication to name services. As the name server switch file, it includes the following entry, which determines what database is searched first.

```
hosts: files dns
```

When a system gets a request to search for a hostname such as outsider1.example.org, the preceding directive means the /etc/hosts file is searched first. If that name is not found in /etc/hosts, the next step is to search available configured DNS servers, normally using that configured in the /etc/resolv.conf file.

A few older software components use the /etc/host.conf file for this purpose. The entries in this file are simple, as they support searches of multiple entries in

/etc/hosts, along with a search starting with that file, followed by a DNS server configured with the Berkeley Internet Name Domain (BIND) software.

```
multi on
order hosts,bind
```

/etc/hosts

The /etc/hosts file is a static database of hostnames/FQDN and IP addresses. It's suitable for small, relatively static networks. However, it can be a pain for networks where there are frequent changes. Every time a system is added or removed, you'll have to change this file—not only on the local system, but also on every other system on that network.

It's well suited to the local network systems created in Chapter 1. A simple version of the file might include the following entries:

```
192.168.122.50 server1.example.com
192.168.122.150 tester1.example.com
192.168.100.100 outsider1.example.org
127.0.0.1 localhost.localdomain localhost
::1 server1.example.com server1 localhost6.localdomain6 localhost6
```

Due to the IPv6 localhost entries, you can't just copy this file to all three test systems. However, it's not hard to replace an entry like server1 with tester1 in a local /etc/hosts file. In some cases, you may want to set up multiple entries for an IP address. For example, the following entries could be added to specify the IP addresses for Web and FTP servers:

```
192.168.122.50 www.example.com
192.168.122.150 ftp.example.com
```

/etc/resolv.conf

The standard file for documenting the location of DNS servers is still /etc/resolv.conf. Typically, it'll have one or two entries, similar to the following:

```
search example.com
nameserver 192.168.122.1
```

The **search** directive appends the example.com domain name to searches for simple hostnames. The **nameserver** directive specifies the IP address of the configured DNS

server. If in doubt about whether the DNS server is operational, run the following command:

```
# dig @192.168.122.1 mheducation.com
```

If needed, substitute the IP address associated with the **nameserver** directive in your /etc/resolv.conf file.

Hostname Configuration Options

During the boot process, the network service looks to the /etc/sysconfig/network file to define the value of the local hostname. It's okay if that hostname is set as a FQDN like tester1.example.com. As suggested earlier, it's a simple file, where the hostname may be documented with a directive like the following:

```
HOSTNAME=tester1.example.com
```

Of course, you can change the value of the hostname with the **hostname** ***newname*** command. However, as such changes aren't reflected in the /etc/hosts file or any DNS server, such changes may not be very helpful.

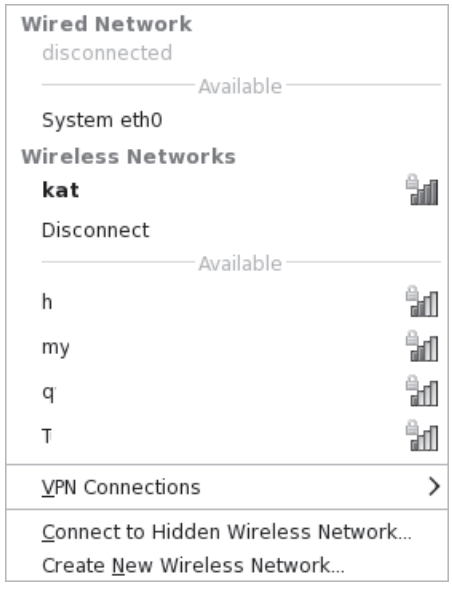
The Network Manager Applet

The Network Manager also includes an applet to help users manage configured network connections. For example, on my personal system, I've configured one Ethernet and one wireless connection. When I left-click the network applet, it reveals available and active connections as shown in Figure 3-14. Note the available connections on wired and wireless networks. I can select different connections as desired. The actual icon associated with the applet varies, depending on whether there's a current active connection. The applet is in the upper-right area of the GNOME Desktop screen, near the date and time, on the top panel. (To help protect the privacy of my neighbors, the names of the wireless networks shown in Figure 3-14 have been partially obscured.)

If you right-click the Network Manager applet, it brings up a menu with configuration options. The options are for our purposes self-explanatory and trivial.

FIGURE 3-14

One Network Manager applet menu



SCENARIO & SOLUTION

Networking is down.	Check physical connections. Run **ifconfig** to check active connections. Run the **/etc/init.d/network status** command. Review the /etc/sysconfig/network file
Unable to access remote systems.	Use the **ping** command to test access to local, and then remote IP addresses.
Current network settings lead to conflicts.	Check network device configuration in /etc/sysconfig/network-scripts files. Review settings with the Network Connections tool.
Network settings not consistent.	Check network device configuration in /etc/sysconfig/network-scripts files. Review settings with the Network Connections tool. The scenario suggests a desire for a static network configuration, so review accordingly.
Hostname is not recognized.	Review /etc/sysconfig/network, run the hostname command, review /etc/hosts for consistency.
Remote hostnames not recognized.	Review /etc/hosts. Check /etc/resolv.conf for an appropriate DNS server IP address. Run the **dig** command to test the DNS server.

CERTIFICATION SUMMARY

The focus of this chapter is two-fold. It covered the basic command line tools formerly associated with Red Hat exam prerequisites. As those objectives have been incorporated into the main body of the RHCSA, they have been combined with network configuration, to allow you to practice these command line tools.

The command line starts with a shell, an interpreter that allows you to interact with the operating system using various commands. While no shell is specified in the objectives, the default shell in most Linux distributions, including RHEL 6, is bash. You can start a command line prompt at one of the default consoles, or at a terminal in the GUI. At the bash prompt, you can manage the files and directories through which Linux is configured and organized. As Linux configuration files are by and large in text format, they can be set up as databases to be searched and modified with a variety of commands. Linux text files can be processed as streams of data that can be interpreted and processed. To edit a text file, you need a text editor such as vim and gedit.

Documentation online within Linux is extensive. It starts with command switches such as **-h** and **--help** that provide hints on what goes with a command. It continues with man and info pages. Many packages include extensive documentation files in the /usr/share/info directory. In many, perhaps most, cases, you do not need Internet access to find the hints needed.

Linux is inherently a network operating system. Network devices such as eth0 can be configured with both IPv4 and IPv6 addressees. Network review and configuration commands include **ifconfig**, **ifup**, **ifdown**, and **dhclient**. Additional related commands include **arp**, **route**, **netstat**, and **ping**. Associated configuration files start with /etc/sysconfig/network. Individual devices are configured in the /etc/sysconfig/network-scripts directory. Network devices can also be configured with the **system-config-network** command at the console and the Network Manager Network Connections tool.

TWO-MINUTE DRILL

Here are some of the key points from the certification objectives in Chapter 3.

Shells

- ❑ The default Linux shell is bash.
- ❑ Six command line consoles are available by default; if the GUI is installed, it takes over the first console.
- ❑ You can open multiple command line terminals in the GUI.
- ❑ Shells work with three data streams: stdin, stdout, and stderr. To that end, command redirection means streams of data can be managed with operators such as >, >>, <, |, and 2>.

Standard Command Line Tools

- ❑ Everything in Linux can be reduced to a file.
- ❑ Commands like **pwd** and **cd** can help navigate directories.
- ❑ Concepts like directory paths, the PATH, and the tilde (~) can help you understand and use commands at the shell.
- ❑ Basic commands allow you to find needed files and read file contents. These commands include **ls**, **find**, and **locate**.
- ❑ File creation (and deletion) commands include **touch**, **cp**, **ln**, **mv**, and **rm**; corresponding directory creation and deletion commands are **mkdir** and **rmdir**.
- ❑ Commands can be customized with the **alias** command.

The Management of Text Files

- ❑ Linux is managed through a series of text configuration files.
- ❑ Text files can be read as streams of data with commands like **cat**, **less**, **more**, **head**, and **tail**.

- ❑ New files can be created, copied, moved, linked, and deleted with the **touch**, **cp**, **mv**, **ln**, and **rm** commands. Commands can be customized with the alias command.
- ❑ File filters such as the **sort**, **grep**, **egrep**, **wc**, **sed**, and **awk** commands support the processing of text streams.
- ❑ Understanding text-editors is a critical skill. An earlier version of the RHCSA objectives specified the use of vim and gedit.

Local Online Documentation

- ❑ If you need a hint for a command, try it by itself; alternatively, try the **-h** or **--help** switches.
- ❑ Command man pages often include examples; **whatis** and **apropos** can search for man pages on different topics.
- ❑ If an info manual is available for a command or file, you'll find it in the /usr/share/info directory.
- ❑ Many packages include extensive documentation and examples in the /usr/share/doc directory.

A Networking Primer

- ❑ IPv4 addresses have 32 bits. There are five classes of IPv4 addresses, and three different sets of private IPv4 addresses suitable for setting up TCP/IP on a LAN.
- ❑ IPv6 addresses have 128 bits, and can be unicast, multicast, or anycast. Unicast addresses can be limited to local networks, or routable.
- ❑ Tools such as **ping**, **ping6**, **arp**, **ifconfig**, and **netstat** can help you diagnose problems on that LAN.
- ❑ Name resolution configuration files such as /etc/resolv.conf determine how a system finds the right IP address; that file may be configured from a DHCP server with the **dhclient** command.

Network Configuration and Troubleshooting

- Linux networking starts with the /etc/init.d/network script and the /etc/sysconfig/network configuration file.
- Individual network devices are configured in the /etc/sysconfig/network-scripts directory.
- Network configuration tools include the console-based **system-config-network** command and the Network Manager Network Connections tool.
- Hostname configuration files include /etc/nsswitch.conf, /etc/hosts, and /etc/resolv.conf.

SELF TEST

The following questions will help you measure your understanding of the material presented in this chapter. As there are no multiple-choice questions on the Red Hat exams, there are no multiple-choice questions in this book. These questions exclusively test your understanding of the chapter. Getting results, not memorizing trivia, is what counts on the Red Hat exams.

Shells

1. What is the name of the default Linux shell?

   _____

2. From the GUI, what key combination moves to virtual console 3?

   _____

Standard Command Line Tools

3. What single command creates the /abc/def/ghi/jkl series of directories?

   _____

4. What symbol represents the home directory of the current user?

   _____

The Management of Text Files

5. What command lists the last ten lines of the /var/log/messages file?

   _____

6. What command returns lines with the term Linux from the /var/log/dmesg file?

   _____

Local Online Documentation

7. What command searches the database of man pages for manuals that reference the passwd command and configuration file?

   _____

8. If there are man pages for the hypothetical abcde command and file, in sections 5 and 8, type in the command that is sure to call up the man pages from section 5?

_____

A Networking Primer

9. In IPv4 addressing, with a network address of 192.168.100.0 and a broadcast address of 192.168.100.255, what is the range of assignable IP addresses?

_____

10. Given the addresses described in question 9, what command assigns IPv4 address 192.168.100.100 to network device eth0?

_____

Network Configuration and Troubleshooting

11. What is the full path to the configuration file with the hostname of the local system?

_____

12. What is the full path to the configuration file associated with the first Ethernet adapter for the local system?

_____

LAB QUESTIONS

Several of these labs involve configuration exercises. You should do these exercises on test machines only. It's assumed that you're running these exercises on virtual machines such as KVM.

Red Hat presents its exams electronically. For that reason, most of the labs in this and future chapters are available from the CD that accompanies the book, in the Chapter3/ subdirectory, in .doc, .html, and .txt formats. In case you haven't yet set up RHEL 6 on a system, refer to Chapter 1 for installation instructions.

The answers for each lab follows the Self Test answers for the fill-in-the-blank questions.

SELF TEST ANSWERS

Shells

1. The default Linux shell is bash, also known as the Bourne-Again shell.
2. From the GUI, the key combination moves to virtual console 3 is CTRL-ALT-F3.

Standard Command Line Tools

3. The single command that creates the /abc/def/ghi/jkl series of directories is **mkdir -p /abc/def/ghi/jkl**.
4. The symbol that represents the home directory of the current user is the tilde (~).

The Management of Text Files

5. The command that lists the last ten lines of the /var/log/messages file is **tail -n10 /var/log/messages**. Since ten lines is the default, **tail /var/log/messages** is also acceptable.
6. The command that returns lines with the term Linux from the /var/log/dmesg file is **grep Linux /var/log/dmesg**. Other variations are acceptable, such as **cat /var/log/dmesg | grep Linux.**

Local Online Documentation

7. The command that searches the database of man pages for manuals that reference the passwd command and configuration file is **whatis passwd**. The **apropos** and **man -k** commands go further, as they list man pages with the text "passwd" in the command or the description.
8. The command that calls up the man page from section 5 for the hypothetical **abcde** command and file is **man 5 abcde**.

A Networking Primer

9. The range of assignable IP addresses in the noted IPv4 network is 192.168.100.1 through 192.168.100.254.
10. Given the addresses described in question 9, the command that assigns IPv4 address 192.168.100.100 to network device eth0 is **ifconfig eth0 192.168.100.100**.

Network Configuration and Troubleshooting

11. The full path to the configuration file with the hostname of the local system is /etc/hosts.
12. The full path to the configuration file associated with the first Ethernet adapter for the local system is /etc/sysconfig/network-scripts/ifcfg-eth0. If you're used to configuring network cards with the Network Connections tool, ifcfg-System_eth0 is also acceptable.

LAB ANSWERS

Lab 1

This lab tested the situation where networking was deactivated with the most innocuous of settings, the NETWORKING directive in the /etc/sysconfig/network file. When set to no, that setting deactivates networking on a system. Nothing else is changed; the IP address information for specific network cards is still correct. Sure, you could still activate networking through other means, but unless **NETWORKING=yes** in the noted file, such changes would not survive a reboot.

The script used in this lab saved the original copy of /etc/sysconfig/network in the /root/backup directory. Now that the lab is complete, you may restore that file to its original location. Be aware, the immutable flag has been applied to the copied file; to delete it from the /root/backup directory, you'd first have to remove the immutable flag with the **chattr -i** command.

Lab 2

This lab set up an invalid IP address configuration for the first Ethernet adapter, eth0. The standard for the systems configured in Chapter 1 is based on the 192.168.122.0/24 network. The configuration file in the /etc/sysconfig/network-scripts directory may go by slightly different names, depending on how that adapter was configured. The original file from that directory was moved to the /root/backup directory. If your efforts in re-creating that configuration file fail, restore the original configuration file from that /root/backup directory.

Be aware, the immutable flag has been applied to the copied file; to delete it from the /root/backup directory, you'd first have to remove the immutable flag with the **chattr -i** command. In fact, before Lab 3 works, you'll have to run the following command:

```
# chattr -i /root/backup/*
```

Lab 3

This lab deactivates the first Ethernet device on the system, and works if that device has the default eth0 device file name. It should also work if you've run the Network Manager Network Connections

tool without changing too many defaults, as the Ch3Lab3 script also deactivates the System_eth0 device file.

Lab 4

This lab replaced the /etc/resolv.conf file. If the local network already uses a DNS server on the 192.168.1.111 IP address, this lab should not cause any problems. The original version of that file was moved to the /root/backup directory. If your efforts in re-creating that configuration file fail, restore the original configuration file from that backup directory.

Lab 5

In this lab, you'll set up the /etc/hosts file on each of the systems described in Chapter 1. Except for the local system settings added by the Network Manager, the data in /etc/hosts on all three systems may be identical. Specifically, that file should include the following entries:

```
192.168.122.50  server1 server1.example.com
192.168.122.150  tester1 tester1.example.com
192.168.100.100  outsider1 outsider1.example.org
```

It doesn't matter that the systems are on different IP networks. As long as there's a routing path between systems, this data in each /etc/hosts file will work. And duplication with data inserted by the Network Manager is not a problem, as long as the data is consistent. In fact, it's possible to set up multiple names for an IP address; for example, if I set up a web server on the 192.168.122.50 system, I could add the following entry to /etc/hosts.

```
192.168.122.50  www.example.com
```

Lab 6

The first four lines were from the original configuration of an eth0 network card. While the values associated with the DEVICE, HWADDR, NM_CONTROLLED, and ONBOOT directives have not changed, the format provided by the RHEL 6 installation is different.

Lab 7

If you've used the Network Connections tool, there may be an ifcfg-System_eth0 file in place of the ifcfg-eth0 file. The system-config-network tool can still handle that file. Just be sure to substitute accordingly. Be sure to learn the settings associated with directives such as USERCTL, BOOTPROTO, and DNS1.