Chapter 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.

Command-line skills are no longer listed as prerequisites, but they are required to achieve the exam objectives. As most candidates for the RHCSA exam should already

be familiar with these command-line tools, this chapter covers the related topics with minimum detail. If after reading this chapter you feel the need for more guidance about these topics, the excellent beginning Linux books described in Chapter 1 can help.

Linux gurus should recognize that we've "oversimplified" a number of explanations to keep this chapter as short as possible. However, because 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

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. Although many Linux gurus use one of the many other shells available, in the exam it is extremely likely you will be faced with bash.

Whatever shell you prefer, 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 is often thought of in Linux as a stream of information. One basic Linux skill is the ability to redirect such 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

Now that you have access to a command line, file and directory management is next. With related commands, you can navigate around the Linux directory tree, as well as perform all the tasks suggested in the related objectives:

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

The Analysis of Text Output

Most Linux configuration files are text files. It is important to understand and analyze the

flow of text as it is sent through the shell. Tools such as the **grep** command can help you focus on needed information. In this chapter, you will examine how to meet the following objective:

- Use grep and regular expressions to analyze text

The Variety of Local Documentation

Internet access is not available during the Red Hat exams, but 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; it describes the commands and directory associated with most Linux online documentation:

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

The objectives include an interesting remark:

- 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 will "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. Although word processors such as 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, and these editors can inject hidden data or otherwise cause problems when used for simple text editing. Therefore, you need to know how to handle the following objective with standard non-GUI utilities:

- Create and edit text files

The Management of Network Services

Although 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 (FQDNs) such as server1.example.com and IP

(*Continued*)

addresses such as 192.168.122.50. The sources from which Linux obtains name resolution information are usually the local /etc/hosts database of hostnames and IP addresses, as well as available databases of Domain Name Service (DNS) servers. 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. Although 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. A text-based shell is also used as a command-line interpreter. In Linux, the shell is the interpreter that allows you to interact with the operating system 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

Users can choose between four command-line shells in RHEL 7. Although 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.
- **ksh** The Korn shell, developed by David Korn at Bell Labs in the 1980s, to incorporate the best features of the Bourne and C shell.
- **tcsh** An enhanced version of the Unix C shell.
- **zsh** A sophisticated shell, similar to the Korn shell.

These shells are located 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 one of the authors' regular accounts is

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

As an example, to change the default shell to ksh, change /bin/bash to /bin/ksh. You also need to install the corresponding RPM package for the Korn shell. Package management will be covered in Chapter 7.

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.

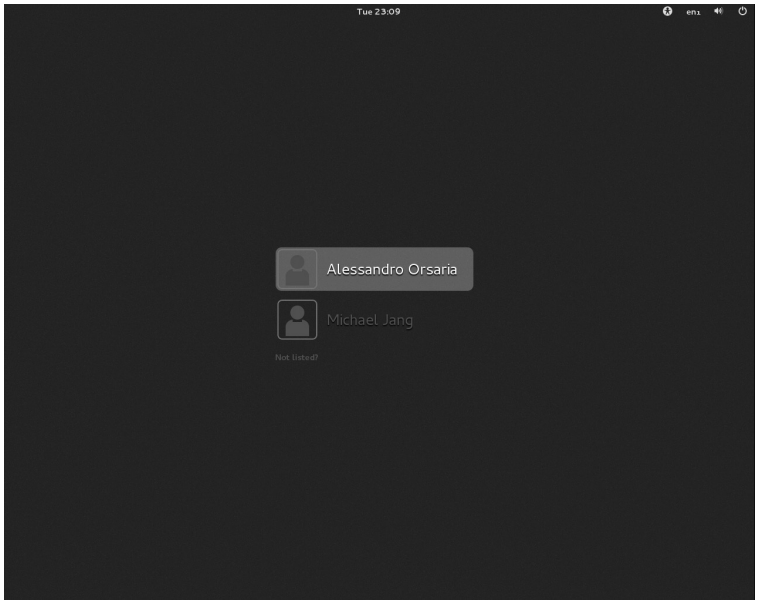
Virtual Terminals

If you have access to the console of a RHEL system, you can use six virtual terminals to open six independent login sessions. However, only one virtual terminal is activated by default. The other login prompts are launched dynamically when you switch to an unused terminal. Virtual terminals are defined by the logind.conf file in the /etc/systemd directory. Take a look at that file. You'll see an option named NAutoVTs, which defines the maximum number of virtual terminals that can be activated. Virtual terminals are associated with device files /dev/tty1 through /dev/tty6. When a GUI is configured, it takes /dev/tty1. It's possible to configure more virtual terminals, limited by those allowed for the root administrative user in the /etc/securetty file.

Normally, to change between virtual terminals, press ALT and the function key associated with that terminal. For example, the ALT-F2 key combination moves to the second terminal. However, in the RHEL GUI, the ALT-F*n* key combinations are used to provide other functionalities, such as to start the Run Application tool via ALT-F2. Therefore, you'll need to press CTRL-ALT-F*n* to move to the *n*th virtual console from the GUI.

FIGURE 3-1

A first GUI login 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 Server
Kernel 3.10.0-123.el7.x86_64 on an 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 you are logged in to the GUI, access to the bash shell is easy. If you're in the default GNOME desktop environment, click Applications | Utilities | Terminal. Traditionally, administrators have worked from the console. But in many cases, working on the command line from the GUI can be helpful, especially with the windows 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 from the GUI-based command line, in part because dark text on a white screen is easier to read.

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 ~]#
```

In this book, 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, whereas standard output and standard error go out to the screen terminal. In the following example, when you run **cat *filename,*** the contents of that file are sent to the screen as standard output (as are any errors):

```
# cat filename
```

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 such as >, >>, 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 want to save the error messages of a program into a file, redirect the error stream from it with a command such as the following:

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

Sometimes you may want to discard all errors. This can be achieved by redirecting the error stream to the special device file /dev/null:

```
# program 2> /dev/null
```

Another useful redirection operator is the ampersand and right arrow (&>), which sends both the standard output and error to a file or device. An example is shown here:

```
# program &> output-and-error
```

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. Although 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.

Remember, in any bash session you can go through the history of previous commands, using the UP- and DOWN-ARROW keys, and CTRL-R to make a search. You can also take advantage of text completion, which allows you to use the TAB key almost as a wildcard to complete a command, a filename, or a variable (if the text begins with the **$** character).

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 options (or "switches") and arguments. Command options allow you to change the behavior of a command and are usually prepended by one or two dashes (such as **ls -a** or **ls --all**). Arguments specify files, devices, or other targets that a command should act on. Only a few commands are covered in this chapter. If you're less familiar with some of them, use their man pages. Study the command options. Try them out! Only with practice, practice, and more practice can you really understand the power behind some of these commands.

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. Those commands will be covered in the next sections, but first we'll review some basic filesystem concepts.

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 commands and concepts to tell you where you are and how to move from directory to directory. The most important 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 in either 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 the user michael runs that command in his home directory, he gets 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 you log in as the root administrative user and run the **ls** command, you should see the following:

```
anaconda-ks.cfg  initial-setup-ks.cfg
```

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

You need to know two path concepts 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 you're running a command absolute paths are essential. Otherwise, commands referencing a wrong directory 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 its relative path to the root (/). If you happen to be in the /home directory when restoring that backup, the files for user michael, for example, would be restored to the /home/home/michael directory.

In contrast, if the /home directory was backed up using the absolute path, the current working directory doesn't matter when you are 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. A shell, such as bash, 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 7:

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:↵
/usr/sbin:/home/michael/.local/bin:/home/michael/bin

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

The directories in the PATH for regular users and the root administrative user are slightly different. But the differences matter because 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 or by scripts in the /etc/profile.d directory. 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.

As well as the global settings, the PATH for individual users can be customized with an appropriate entry in that user's home directory, in the hidden files named ~/.bash_profile or ~/.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. Another common shortcut is two consecutive dots (..) to represent the directory that is one level up in the hierarchy. Thus, **cd ..** moves to the parent directory of the current working directory.

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 last modification time, last access time, 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 list sorted by modification time, and **ls -i** for inode numbers (inodes are internal data structures in a filesystem that store information about a file). Other useful command options are **-r**, to reverse the listing order, and **-R**, to list the contents of all subdirectories recursively.

You can combine switches; we often use the **ls -ltr** command to display a recursive long listing with the most recently changed files last. The **-d** switch, when combined with others, can give you more information on the current directory, or on a directory that you have passed as an argument to the **ls** command.

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       adm
drwxr-xr-x. root root system_u:object_r:var_t:s0       cache
drwxr-xr-x. root root system_u:object_r:kdump_crash_t:s0 crash
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:public_content_t:s0 ftp
drwxr-xr-x. root root system_u:object_r:games_data_t:s0 games
drwx--x--x. gdm  gdm  system_u:object_r:xserver_log_t:s0 gdm
drwxr-xr-x. root root system_u:object_r:var_t:s0       gopher
drwxr-xr-x. root root system_u:object_r:var_t:s0       kerberos
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
lrwxrwxrwx. root root system_u:object_r:var_lock_t:s0  lock -> ../run/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
lrwxrwxrwx. root root system_u:object_r:var_run_t:s0   run -> ../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_t:s0       var
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 www
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, although 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 time of the last modification of a file. For example, try the following three commands:

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

Note the timestamps listed with the output of each **ls -l** command, and compare them with the current date and time returned by **date**. After you run the **touch** command, the timestamp of /etc/passwd is updated to the current date and time.

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 on *file2* in the current directory. 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.

Another usage of the cp command is with multiple file sources to be copied into a single destination directory. In this case the syntax is **cp** ***file1 file2 ... dir***.

The **cp** command, with the **-a** switch, supports recursive changes and preserves all file attributes, such as permissions, ownerships, and timestamps. For example, the following command copies all subdirectories of the noted directory, along with associated files, into /mnt/backup:

```
# cp -a /home/michael/. /mnt/backup/
```

mv

Although there isn't a "rename" command in Linux, you can use **mv**. The **mv** command 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 filesystem, everything about the file, including the inode number, remains the same. The **mv** command works with directories too.

ln

Linked files allow users to refer to the same file using different names. When linked files are devices, they may represent more common names, such as /dev/cdrom. File links can be hard or soft.

Hard links are directory entries that point to the same inode. They must be created within the same filesystem. You could delete a hard-linked file in one directory, and it would still exist in the other directory (files are only deleted when the number of dentry records pointing to them hit 0, which is tracked via a counter per file). 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. Although 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 we downloaded the source files for the Linux kernel, several thousand files were included in the /root/rpmbuild/BUILD /kernel-3.10.0-123.el7 directory. Obviously, it's not practical to delete those files one by one. Therefore, the **rm** command includes some powerful switches. The following command removes all of those files in one go:

```
# rm -rf /root/rpmbuild/BUILD/kernel-3.10.0-123.el7
```

The **-r** switch works recursively, and the **-f** switch overrides any safety precautions, such as the **-i** switch shown in the output to the **alias** command for the root administrative user. It's still quite a dangerous command. For example, a simple typing mistake such as putting a space between the first forward slash and the directory name, as shown here, would first delete every file starting with the top-level root directory (/) before looking for the root/rpmbuild/BUILD/kernel-3.10.0-123.el7 subdirectory:

```
# rm -rf / root/rpmbuild/BUILD/kernel-3.10.0-123.el7
```

This would delete every file on the system, including any mount points.

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, you can use the following command to create 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 provide a bit of safety. To see the aliases for the current user, run the **alias** command. The following output shows the default Red Hat aliases for the root user:

```
alias cp='cp -i'
alias egrep='egrep --color=auto'
alias fgrep='fgrep –color=auto'
alias grep='grep --color=auto'
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'
```

TABLE 3-1

Wildcards in the Shell

Wildcard	Description
*	Any number of 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 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.

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.

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.

The use of wildcards is 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 a 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
```

But the speed of that search depends on the memory and disk speed available on the local system. 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** script.

exam Watch

When you take Red Hat exams, you can run updatedb **or the noted** mlocate **script early on to find needed files more quickly later on without delay.**

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
```

CERTIFICATION OBJECTIVE 3.03

The Management of Text Files

Linux and Unix are typically managed through a series of text files. Linux administrators do not normally use graphical editors to manage these configuration files. Editors such as OpenOffice.org Writer or Microsoft Word normally either save files in a binary format or change the encoding of plain-text files. Unless text files are preserved in their original format, 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. However, that data can be overwhelming without tools that can sort through that data. 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 such as **cd**, **ls**, and **pwd** that can help you get around Linux files. With commands such as **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 command utilities that can help you scroll through the file text at your leisure. These utilities are known as *pagers,* and the most common are **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, PAGE DOWN, and arrow 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 use the forward slash and question mark to search forward or backward 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

```
Dec 28 09:36:31 server1 NetworkManager: DHCPREQUEST on eth0 to 255.255.255.255 p
ort 67 (xid=0x14b16e00)
Dec 28 09:36:31 server1 NetworkManager[829]: <info> (eth0): DHCPv4 state changed
 nbi -> preinit
Dec 28 09:36:31 server1 dhclient[1707]: DHCPACK from 192.168.122.1 (xid=0x14b16e
00)
Dec 28 09:36:31 server1 NetworkManager: DHCPACK from 192.168.122.1 (xid=0x14b16e
00)
Dec 28 09:36:31 server1 dhclient[1707]: bound to 192.168.122.225 -- renewal in 1
441 seconds.
Dec 28 09:36:31 server1 NetworkManager: bound to 192.168.122.225 -- renewal in 1
441 seconds.
Dec 28 09:36:31 server1 NetworkManager[829]: <info> (eth0): DHCPv4 state changed
 preinit -> reboot
Dec 28 09:36:31 server1 NetworkManager[829]: <info>   address 192.168.122.225
Dec 28 09:36:31 server1 NetworkManager[829]: <info>   plen 24 (255.255.255.0)
Dec 28 09:36:31 server1 NetworkManager[829]: <info>   gateway 192.168.122.1
Dec 28 09:36:31 server1 NetworkManager[829]: <info>   server identifier 192.168.
122.1
Dec 28 09:36:31 server1 NetworkManager[829]: <info>   lease time 3600
Dec 28 09:36:31 server1 NetworkManager[829]: <info>   hostname 'server1'
Dec 28 09:36:31 server1 NetworkManager[829]: <info>   nameserver '192.168.122.1'
Dec 28 09:36:31 server1 NetworkManager[829]: <info> Activation (eth0) Stage 5 of
:
```

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 such as **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.

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 tools 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 **-n*xy*** switch. For example, the **tail -n 15 /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 with failed login attempts, the following command monitors the noted file and displays new lines on the screen as new log entries are recorded:

```
# 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 terminal. When these files get large, it's convenient to have commands that can filter and otherwise process these streams of text.

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

Tarballs are a common way to distribute Linux packages. Packages 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

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.

You can use regular expressions within a **grep** command. Regular expressions provide a powerful way to specify complex search patterns. Some of the characters that have a special meaning inside a regular expression are shown in Table 3-2. If you want a metacharacter to lose its special meaning and be taken literally, precede it by a blackslash (\).

The **grep** command supports some useful switches. To make the search case-insensitive, you can pass the **-i** option to the command line. The **-E** option enables the use of extended regular expression syntax. Another interesting switch is **-v**, which reverses the matching logic—that is, it tells **grep** to select only the lines that do *not* match a regular expression.

As an example, suppose you want to select only the lines from /etc/nsswitch.conf that are not blank and do not contain a comment (that is, they do not start with the # character). This can be achieved by running the following command:

```
# grep -v '^$' /etc/nsswitch.conf | grep -v '^#'
```

TABLE 3-2 Special Characters in Regular Expressions

Metacharacter	Description
.	Any single character. Often used with the * multiplier to indicate any number of characters.
[]	Match any single character included within the square brackets. For example, the command **grep 'jo[ah]n' /etc/passwd** would return all lines in /etc/passwd that contain the string *joan* or *john*.
?	Match the preceding element zero or one time. For example, the command **grep -E 'ann?a' /etc/passwd** would return all lines in /etc/passwd that contain the string *ana* or *anna*.
+	Match the preceding element one or more times. For example, the command **grep -E 'j[a-z]+n' /etc/passwd** would return all lines in /etc/passwd that contain the letters *j* and *n*, with one or more lowercase letters in between. Therefore, this regular expression would match strings such as *joan*, *john*, and also *jason* and *jonathan*.
*	Match the preceding element zero or more times. For example, the command **grep 'jo[a-z]*n' /etc/passwd** would return all lines in /etc/passwd that contain the string *jo* followed by zero or more lowercase letters and terminated by the character *n*. Therefore, the previous regular expression would match strings such as *jon*, *joan*, or *john*.
^	Match the beginning of a line. For example, the command **grep '^bin' /etc/passwd** would return all lines in /etc/passwd that start with the sequence of characters *bin*.
$	Match the end of a line. For example, the command **grep '/bin/[kz]sh$' /etc/passwd** would return all lines in /etc/passwd that terminate with the sequence of characters */bin/ksh* or */bin/zsh* (that is, all records corresponding to the users who have set Korn or Zsh as their default shells).

Note how the first **grep** command selects all the lines that are not blank (the regular expression that matches a blank line is **^$**—that is, start of line and immediately end of line). Then, the output is piped to a second **grep** command, which excludes all the lines that start with a hash character.

The same result can be obtained with a single instance of **grep** and the **-e** switch, which allows you to specify multiple search patterns on the same command:

```
# grep -v -e '^$' -e '^#' /etc/nsswitch.conf
```

For more information on regular expressions, type **man 7 regex**.

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 Manager Connections Editor 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
```

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 exchanges 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), rather than a command, is more of a full programming language. 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 fourth field in /etc /passwd (the group ID) of every user with a listing of "mike":

```
# awk -F : '/mike/ {print $4}' /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, we 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, the vim editor can be started with the **vi** command. Hereafter, we refer to that text editor as vi.

We believe every administrator needs at least a basic knowledge of vi. Although emacs would also be a valid choice, 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 you'll have available.

While RHEL 7 also includes access to the more intuitive nano editor, a knowledge of vi commands can help you in searching and editing key sections of text files more quickly. Although the RHEL rescue media supports more console-based editors, vi is one of the most feature-rich and efficient editors available on Linux.

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. 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, you can consult a number of the books available on the topic, as well as a tutorial that you can start through the **vimtutor** command.

FIGURE 3-4

The vi editor with /etc/nsswitch.conf

```
passwd:     files sss
shadow:     files sss
group:      files sss
#initgroups: files

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

# 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
```

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 "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, **yy** copies the current line into a buffer, **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, respectively.**

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. Although 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. There are several ways to do this in command mode, including the DOWN ARROW key, the PAGE DOWN key, and the **G** command.
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 UID numbers of 1000 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 the buffer. You can then restore or put in 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, 1001, 1001, Tweedle Dee, and /home/tweedle. Make sure the username also corresponds to the home directory.

```
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/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
named:x:25:25:Named:/var/named:/sbin/nologin
oprofile:x:16:16:Special user account to be used by OProfile:/var/lib/oprofile:/
sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/:/sbin/nologin
colord:x:998:996:User for colord:/var/lib/colord:/sbin/nologin
abrt:x:173:173::/etc/abrt:/sbin/nologin
chrony:x:997:995::/var/lib/chrony:/sbin/nologin
libstoragemgmt:x:996:994:daemon account for libstoragemgmt:/var/run/lsm:/sbin/no
login
qemu:x:107:107:qemu user:/:/sbin/nologin
radvd:x:75:75:radvd user:/:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
saslauth:x:995:76:"Saslauthd user":/run/saslauthd:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:993:991::/run/gnome-initial-setup/:/sbin/nologin
michael:x:1000:1000:Michael Jang:/home/michael:/bin/bash
tweedle:x:1001:1001: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 because 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 in the previous illustration, this would be a group name of tweedle and a group number of 1001.
11. Repeat the aforementioned **:wq** command to close vi and save the change.
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. 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.
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 such as **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
```

Because 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/v2.3/nano.html.

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

Edit Text Files in the GUI

No question, the Red Hat exams have become friendlier for users of 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.)

FIGURE 3-5

The nano editor with /etc/nsswitch.conf

```
  GNU nano 2.3.1          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 64 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
```

If the gedit editor is not installed in your system, run the **yum install gedit** command. Once gedit is installed, you can start it by clicking Applications | Accessories | gedit. Because 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 and use X forwarding 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

Although there's no Internet access allowed during Red Hat exams, there is a lot of documentation available already installed on a RHEL 7 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. Although fewer commands and files have such documents, when available, they do provide even more information.

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.

When You Need Help

The first thing we usually do when we 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** switch. Sometimes a mistake leads to some hints; the output to the following command suggests legal options to the **cd** command:

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

FIGURE 3-6

Help with the ls command

```
[alex@server1 ~]$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      scale sizes by SIZE before printing them; e.g.,
                               '--block-size=M' prints sizes in units of
                               1,048,576 bytes; see SIZE format below
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                               modification of file status information);
                               with -l: show ctime and sort by name;
                               otherwise: sort by ctime, newest first
  -C                         list entries by columns
      --color[=WHEN]         colorize the output; WHEN can be 'never', 'auto',
                               or 'always' (the default); more info below
  -d, --directory            list directories themselves, not their contents
  -D, --dired                generate output designed for Emacs' dired mode
  -f                         do not sort, enable -aU, disable -ls --color
```

Sometimes the **-h** switch is more helpful; take a look at the output to the **fdisk -h** command. But the **-h** option doesn't always show a help message; in that case, the **--help** switch may serve that purpose. Look at Figure 3-6 as an example, which displays the output to the **ls --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 **lvextend** command. To browse the man page for that command, run **man lvextend**. As with many other commands, there's an EXAMPLES section, like that shown in Figure 3-7. If you've run the **lvextend** command before, that section may help jog your memory.

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
```

FIGURE 3-7

Examples from the lvextend man page

```
       --use-policies
              Resizes  the  logical volume according to configured policy. See
              lvm.conf(5) for some details.

Examples
       Extends the size of the logical volume "vg01/lvol10" by 54MiB on physi-
       cal 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 -L +54 /dev/vg01/lvol10 /dev/sdk3

       Extends the size of logical volume "vg01/lvol01" by the amount of  free
       space  on  physical  volume /dev/sdk3. This is equivalent to specifying
       "-l +100%PVS" on the command line:

       lvextend /dev/vg01/lvol01 /dev/sdk3

       Extends a logical volume "vg01/lvol01" by 16MiB using physical  extents
       /dev/sda:8-9 and /dev/sdb:8-9 for allocation of extents:

       lvextend -L+16M vg01/lvol01 /dev/sda:8-9 /dev/sdb:8-9

SEE ALSO
       fsadm(8),  lvm(8), lvm.conf(5), lvcreate(8), lvconvert(8), lvreduce(8),
       lvresize(8), lvchange(8)
```

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 httpd, associated with the Apache web server, commands such as **whatis httpd** and **apropos apachectl** 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 man-db.cron job in the /etc/cron.daily directory. As that script is already executable, the following command updates the database of man pages:

```
# /etc/cron.daily/man-db.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 also an httpd-manual package installed separately from the Apache web server.

In some cases, multiple man pages are 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 man page associated with the **smbpasswd** command. In this case, if you want the man page for the smbpasswd encrypted password file, run the following command:

```
$ man 5 smbpasswd
```

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

The info Manuals

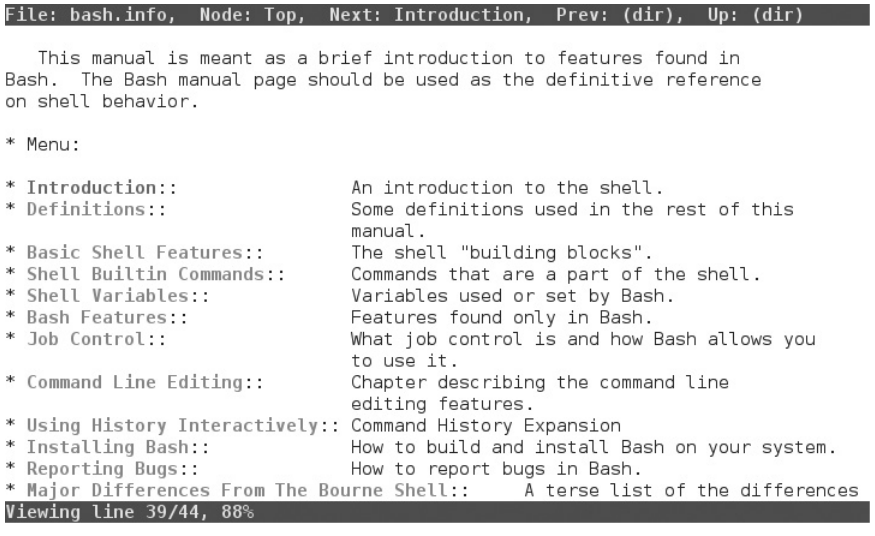
The list of available info manuals is somewhat limited. However, the coverage of some topics (for example, the bash shell) is usually more extensive than a corresponding man page. For a full list of info documents, run the **ls /usr/share/info** command. When an info manual is not available, a request defaults to the associated man page.

To learn more about the bash shell, run the **pinfo bash** command. **pinfo** has a user interface similar to the Lynx web browser and it is a more user-friendly alternative to the traditional **info** 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**.

FIGURE 3-8

A sample info manual



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.

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 you're configuring administrators with different privileges.

The documentation may include entire manuals in HTML format. For an example, take a look at the pam-*/ subdirectory, which includes an entire online manual for the Pluggable Authentication Modules (PAM) system discussed in Chapter 10.

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. IP addresses help you communicate across a network. A number of TCP/IP tools and configurations are available to 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 any Red Hat exam unless you understand networking in some detail.

Although the focus of current networks is still on IP version 4 addressing, some organizations have mandated a move toward IP version 6 (IPv6) networks. The focus of this section is on IPv4, whereas IPv6 will be covered in Chapter 12. However, most of the configuration files and tools that are used for IPv4 also apply to IPv6.

IPv4 Networks

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 known as *dynamic* IP addresses.

TABLE 3-3

IP Address Classes

Class	IP Range	Note
A	1.1.1.0–127.255.255.255	Allows networks of up to 16,777,214 hosts
B	128.0.0.0–191.255.255.255	Allows networks of up to 65,534 hosts
C	192.0.0.0–223.255.255.255	Allows networks of up to 254 computers
D	224.0.0.0–239.255.255.255	Reserved for multicasts
E	240.0.0.0–255.255.255.255	Reserved for experimental use

IPv4 addresses are 32-bit binary numbers and are usually expressed in "dot-decimal" notation (such as 192.168.122.50), with each decimal octet representing 8 bits. An IP address is made up of two parts: a network (or subnet) address and a host part. Until the publication of RFC 1517 in 1993 by the Internet Engineering Task Force (www.ietf.org), IP addresses were categorized into different classes, which defined the size of the network and the host part of the address.

Today, IP addresses are usually analyzed using a classless logic. The subnet mask, rather than the address class, is used to determine the network and host parts of an IP address. The old classful addressing scheme introduced by RFC 791 is shown Table 3-3. Some concepts of RFC 791 still remain in practice today; as an example, the IP range 224.0.0.0–239.255.255.255 is used for multicast addresses.

In addition, a number of private IP addresses are not to be assigned to any computer that is directly connected to the Internet. The most common private network ranges are defined in RFC 1918 and are associated with network addresses 10.0.0.0–10.255.255.255, 172.168.16.0–172.168.31.255, and 192.168.0.0–192.168.255.255. In addition, network addresses 127.0.0.0 through 127.255.255.255 are used for loopback communication on a local host.

Networks and Routing

As we have discussed in the previous section, an IP address has two parts: a network prefix and a host identifier. To determine the network and the host part, IP addresses are associated with a subnet mask (also known as netmask or prefix). This is a 32-bit number made of a sequence of binary ones followed by zeros.

The subnet mask can be represented in the same dot-decimal notation used for IPv4 addresses. For example, 255.255.255.0 is a subnet mask made of 24 binary ones and 8 zeros. An alternative notation is known as Classless Inter-Domain Routing (CIDR) notation and consists of a slash character (/) followed by a number that indicates the amount of one bits in the netmask. As an example, the subnet mask 255.255.255.0 can be written as /24 in CIDR notation.

Given an IP address and a subnet mask, all you have to do to determine the network portion of the IP address is provide a logical AND between the IP address and the netmask. For example, given the IP address 192.168.122.50 with netmask /24, the first three bytes of the address (192.168.122) represent the network part, whereas the last byte (50) is the host identifier.

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, as you have seen, helps your computer define the network and the host portion of an IP address. You can assign IP addresses between the network and broadcast addresses (not including these addresses) to any computer on the network.

As an example, let's define the range of addresses for a private network. Start with the private network address 192.168.122.0 and a subnet mask of 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.

IP addresses are assigned to network interfaces. A host with multiple network interfaces that forwards traffic across different networks is called a *router*. IP hosts separated by other groups of IP hosts by a router must be located in different 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 other networks. Although that gateway IP address is part of the local network, that address is assigned to 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 **ip route** command described in the following section.

Tools and Commands

A substantial number of tools are available to manage the TCP/IP protocol suite on your Linux computer. In the previous versions of Red Hat Enterprise Linux, some of the most important network management commands were **ifconfig**, **arp**, **netstat**, and **route**. Those commands have been deprecated. The **ip** tool supports more advanced features. To ease the transition to the **ip** tool, Table 3-4 provides a list of the deprecated commands, along with their equivalent **ip** commands.

By default, Red Hat Enterprise Linux 7 names network interfaces based on their physical location (enoX and emX for onboard network interfaces, and enpXsY and pXpY for PCI slots). RHEL 7 will use the traditional enumeration method of eth0, eth1, ... only as a fallback choice. Hence, you may find the first onboard network interface to be named *eno1*, while an interface located on PCI bus 3, slot 0 would be named *enp3s0*.

TABLE 3-4 The ifconfig, arp, and netstat Commands with Their Equivalent ip Commands

Obsolete Command	Equivalent Command in RHEL 7	Description
ifconfig	ip [-s] link ip addr	Shows the link status and IP address information for all network interfaces
ifconfig eth0 192.168.122.150 ↵ netmask 255.255.255.0	ip addr add ↵ 192.168.122.150/24 dev eth0	Assigns an IP address and netmask to the eth0 interface
arp	ip neigh	Shows the ARP table
route netstat -r	ip route	Displays the routing table
netstat -tulpna	ss -tupna	Shows all listening and non-listening sockets, along with the program to which they belong

Other important network commands are **ping** and **traceroute**, which are often used to diagnose and troubleshoot network problems.

But these are just the tools. 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.

ping and traceroute

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 host 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 your router on the Internet. 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). It could also indicate a problem with the /etc/hosts configuration file.

The **traceroute** command automates the process just described by tracking the route path to a destination. For example, the following command finds the path to the IP address 192.168.20.5:

```
# traceroute -n 192.168.20.5
traceroute to 192.168.20.5 (192.168.20.5), 30 hops max, 60 byte
packets
 1  192.168.122.1  0.204 ms  0.152 ms  0.148 ms
 2  192.168.1.1  1.826 ms  2.413 ms  4.050 ms
 3  192.168.20.5  2.292 ms  2.630 ms  2.554 ms
```

Look at the **-n** option in this command. This tells **traceroute** to display IP addresses rather than hostnames. The command also shows the round trip time (RTT) to reach each hop along the path. By default, three different probes are sent for each hop.

Please note that some **traceroute** command options require root privileges. Another command that serves the same purpose and is not subjected to this limitation is **tracepath**.

By default, traceroute **relies on UDP probe packets with an increasing time-to-live (TTL) value in the IP header in order to find the route path to a given destination. Sometimes, a firewall along the path may block UDP packets. In that case, you may try to run** traceroute **with the** -I **or** -T **option to enable ICMP or TCP probe packets, respectively.**

Review Current Network Adapters with ip

The **ip** command can display the current state of active network adapters. It also can be used to assign network addresses and more. Run the **ip link show** command to review the link status of the active network adapters on the system. Optionally, include the **-s** switch if you want to display statistics about network performance.

To review IP address information, try the **ip address show** command, which gives the same output of **ip link show**, but it also includes IP addresses and their properties. The

ip address show eth0 command listed next reflects the current configuration of the first Ethernet network adapter:

```
# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state ↵
UP qlen 1000
    link/ether 52:54:00:40:1e:6a brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.50/24 brd 192.168.122.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::2e0:4cff:fee3:d106/64 scope link
       valid_lft forever preferred_lft forever
```

The **ip** command is flexible. For example, the **ip a s** command is functionally equivalent to **ip addr show** or **ip address show**.

Configure a Network Adapter with ip

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

```
# ip addr add 192.168.122.150/24 dev eth0
```

The first argument, **192.168.122.150/24**, specifies the new IP address and netmask. The next argument, **dev eth0**, tells you which device is being configured. To make sure the change worked, run the **ip addr show eth0** command again.

With the right options, the **ip** command can modify additional network settings. Some of these options are shown in Table 3-5.

TABLE 3-5 ip Command Options

Command	Description
ip link set dev *device* up	Activates the specified interface.
ip link set dev *device* down	Deactivates the specified interface.
ip addr flush dev *device*	Removes all IP addresses from the specified interface.
ip link set dev *device* txqlen ***N***	Changes the length of the transmit queue for the specified interface.
ip link set dev *device* mtu ***N***	Sets the maximum transmission unit as *N*, in bytes.
ip link set dev *device* promisc on	Activates promiscuous mode. This allows the network adapter to read all packets received, not just the packets addressed to the host. Can be used to analyze the network for problems or to try to decipher messages between other hosts.
ip link set dev *device* promisc off	Deactivates promiscuous mode.

Of course, you'll want to make sure the changes survive a reboot, whether it be for the exam or for a server that you want to administer remotely. That depends on appropriate changes to configuration files in the /etc/sysconfig/network-scripts directory, described shortly. Also, any changes made with the **ip** command are, by definition, temporary.

Activate and Deactivate Network Adapters

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

```
# ip link set dev eth0 down
# ip link set dev eth0 up
```

However, a couple of more intuitive scripts are designed to control network adapters: **ifup** and **ifdown**. Unlike the **ip** command, they call appropriate configuration files and scripts in the /etc/sysconfig/network-scripts directory.

For example, the **ifup eth0** command activates the Ethernet network adapter named eth0, based on the ifcfg-eth0 configuration file and the ifup-eth script in the /etc/sysconfig/network-scripts directory.

ip as a Diagnostic Tool

The Address Resolution Protocol (ARP) associates the hardware address of a network interface (MAC) with an IP address. The **ip neigh** command displays a table of hardware and IP addresses on the local computer. This command can help detect problems such as duplicate addresses on the network. Such problems may happen with improperly configured systems or cloned virtual machines. If needed, the **ip neigh** command can set or modify ARP table entries manually. As hardware addresses are not routable, an ARP table is limited to the local network. Here's a sample output from the command, showing all ARP entries in the local database:

```
# ip neigh show
192.168.122.150 dev eth0 lladdr 52:a5:cb:54:52:a2 REACHABLE
192.168.100.100 dev eth0 lladdr 00:a0:c5:e2:49:02 STALE
192.168.122.1 dev eth0 lladdr 00:0e:2e:6d:9e:67 REACHABLE
```

The first column in the output lists known IP addresses on the LAN, followed by the interface to which the neighbor is attached, and its link layer address (MAC address). The last entry shows whether or not the neighbor's hardware address is reachable. A STALE entry may indicate that its ARP cache timeout has expired since a packet was last seen from that host. If the ARP table is empty, no recent connections exist to other systems on the local network.

Routing Tables with ip route

The **ip** command is versatile. One important version of this command, **ip route**, displays routing tables. It's functionally equivalent to the deprecated **route** command. When run with the **-r** switch (**ip -r route**), this command looks to /etc/hosts files and DNS servers to display hostnames rather than numeric IP addresses.

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

```
default via 192.168.122.1 dev eth0  proto static  metric 1024
192.168.122.0/24 dev eth0  proto kernel  scope link  src 192.168.122.50
```

The deprecated **netstat -nr** command should display the same table. For this routing table, the gateway IP address is 192.168.122.1. Any network packets with a destination other than the 192.168.122.0 network are sent through the gateway address (in other words, the layer 2 address for this gateway is looked up and put in the frame as the destination MAC address). The system at the gateway address, usually a router, forwards that packet to the next router according to its routing table until it gets to a router that is directly connected to the destination.

Dynamically Configure IP Addresses with dhclient

Although the name of the command has changed from time to time, the functionality has remained the same. The **dhclient** command, used with the device name of a network card, such as **eth0**, calls a Dynamic Host Configuration Protocol (DHCP) server for an IP address and more:

```
# dhclient eth0
```

Generally, the network options 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 **ip** command described earlier, but it also sets up the default route for the routing table shown with the **ip route** command. In addition, it adds the IP address of the DNS server to the /etc/resolv.conf configuration file.

Display Network Connections with ss

The **ss** command replaces the deprecated **netstat** tool to display network connections. With the right combination of command switches, it can show listening and nonlistening TCP and UDP sockets. One command we like to use is

```
# ss -tuna4
```

FIGURE 3-9

Output from the ss -tuna4 command

```
[root@server1 ~]# ss -tuna4
Netid  State      Recv-Q Send-Q     Local Address:Port       Peer Address:Port
tcp    UNCONN     0      0                  *:68                    *:*
tcp    UNCONN     0      0                  *:111                   *:*
tcp    UNCONN     0      0                  *:123                   *:*
tcp    UNCONN     0      0             127.0.0.1:323                *:*
tcp    UNCONN     0      0                  *:609                   *:*
tcp    UNCONN     0      0                  *:43630                 *:*
tcp    UNCONN     0      0             127.0.0.1:659                *:*
tcp    UNCONN     0      0                  *:45931                 *:*
tcp    UNCONN     0      0                  *:5353                  *:*
tcp    UNCONN     0      0                  *:61050                 *:*
tcp    LISTEN     0      100           127.0.0.1:25                 *:*
tcp    LISTEN     0      128                *:52991                 *:*
tcp    LISTEN     0      128                *:111                   *:*
tcp    LISTEN     0      128                *:22                    *:*
tcp    LISTEN     0      128           127.0.0.1:631                *:*
tcp    ESTAB      0      0         192.168.122.50:22       192.168.122.1:43910
[root@server1 ~]#
```

where the command shows all (**-a**) network sockets using IPv4 (**-4**) and both the TCP (**-t**) and UDP (**-u**) protocols in numeric (**-n**) format. If the **-p** switch is specified, **ss** will also show the PID of the process using each socket. Figure 3-9 illustrates the output on the baseline server.

At the end of the output, note the peer address of 192.168.122.1:43910. The 43910 port number is just the source port on the remote server. The corresponding local address of 192.168.122.50:22 specifies a port number of 22 (the local SSH service) for a connection from 192.168.122.1. You may also see a second entry with the same port number, which identifies the associated SSH daemon listening for connections. Other lines in this output identify other listening services.

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 configuration files. These configuration files determine whether networking is started during the boot process. If networking is activated, these files also determine whether addresses and routes are configured statically as documented or dynamically with the help of commands such as **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 hostname resolution is not working.

Some of the most common causes of network problems are physical in nature. 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:

```
# systemctl status network
```

RHEL 7 uses a service known as the Network Manager to monitor and manage network settings. Using the **nmcli** command-line tool, you can interact with Network Manager and display the current status of network devices:

```
# nmcli dev status
```

The command should list configured and active devices. If a key device such as eth0 is not listed as connected, your network connection is probably down or the device is unconfigured. Key configuration files are located in the /etc/sysconfig/network-scripts directory.

Sometimes mistakes happen. If you've deactivated an adapter or just lost a wireless connection, try something simple: restart networking. The following command restarts networking with current configuration files:

```
# systemctl restart network
```

Always use systemctl **to execute the network script. Never run the RHEL 7 /etc/init.d/network script directly because it may fail to execute cleanly.**

If a simple restart of networking services doesn't work, then it's time to get into the files. The /etc/sysconfig/network-scripts directory is where Red Hat Enterprise Linux stores and retrieves networking information. With available Red Hat configuration tools, you don't have to touch these files, but it's good to know they're there. A few representative files are shown in Table 3-6.

/etc/sysconfig/network

If you run the **ip addr show** command and see no output, that means all network devices are currently inactive. The first thing to check in that case is the contents of the

TABLE 3-6 Files in the /etc/sysconfig/network-scripts Directory

File in /etc/sysconfig/network-scripts	Description
ifcfg-lo	Configures the loopback device, a virtual device that is used for network communication within the local host.
ifcfg-*	Each installed network adapter, such as em1, gets its own ifcfg-* script. For example, eth0 is given file ifcfg-eth0. This file includes the IP address information required to identify this adapter on a network.
network-functions	This script contains functions used by other network scripts to bring network interfaces up and down.
ifup-* and ifdown-*	These scripts activate and deactivate their assigned protocols. For example, **ifup-ppp** brings up a PPP device, usually a telephone modem.

/etc/sysconfig/network configuration file. It's a pretty simple file and usually contains one or two configuration lines. On systems that are configured to retrieve addressing information via DHCP, this file is usually empty.

If the /etc/sysconfig/network file contains the setting **NETWORKING=no**, then the /etc/init.d/network script doesn't activate any network devices. 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**), and 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 set up only for the purposes of installation. If you did not configure networking when setting the hostname during the GUI installation process, networking will not be configured on the system. In that case, the ifcfg-eth0 file would contain at least the following two directives:

```
HWADDR="F0:DE:F3:06:C6:DB"
TYPE=Ethernet
```

Of course, if networking were not configured during the installation process, there's no reason for the interface to be activated during the boot sequence:

```
ONBOOT="no"
```

By default, RHEL 7 uses a service known as the Network Manager to manage network settings. To make sure it's running, execute the **systemctl status NetworkManager** command. Network Manager includes **nmcli**, a command-line tool to control the status of the service and apply network configuration changes.

Rather than modifying the configuration via **nmcli**, you can change a device configuration file directly. For that purpose, the configuration file shown in Figure 3-10 provides a guide.

Most of these directives are straightforward. They define the device as an Ethernet network card with name eth0, using a defined IP address, a netmask, a default gateway, and a DNS server. Of course, if you prefer to use a DHCP server, the static network address

FIGURE 3-10

A manual configuration of eth0

```
HWADDR="00:50:56:40:1E:6A"
TYPE="Ethernet"
BOOTPROTO="none"
NAME="eth0"
UUID="394f6436-5524-4154-b26e-6649b4d29027"
ONBOOT="yes"
IPADDR0="192.168.122.50"
PREFIX0=24
GATEWAY0="192.168.122.1"
DEFROUTE="yes"
DNS1="192.168.122.1"
~
~
~
~
```

information specified in the last five lines of the file would be omitted, and the following directive would be changed:

```
BOOTPROTO=dhcp
```

After saving the file, you still have to notify Network Manager of the changes. This is achieved by running the following commands (**con** is short for **connection**):

```
# nmcli con reload
# nmcli con down eth0
# nmcli con up eth0
```

Shortly, you'll see how to use Network Manager's command-line tool to modify the configuration of a network device.

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 the 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 directive:

```
192.168.100.0/24 via 192.168.122.1
```

Network Configuration Tools

Red Hat includes several tools that can be used to configure network devices in RHEL 7. The first is the Network Manager command-line tool, **nmcli**. If you prefer a text-based graphical tool, **nmtui** can be started from a virtual terminal. As an alternative, the Network Manager Connections Editor is a GTK+ 3 application that you can start from a GUI command line with the **nm-connection-editor** command. The GNOME shell also includes a graphical utility that can be opened by clicking Applications | Sundry | Network Connections.

All the tools mentioned interact with Network Manager, a system service that is responsible for managing network devices.

The nmcli Configuration Tool

Network Manager can store different profiles, also known as *connections,* for the same network interface. This allows you to switch from one profile to another. For example, you may have a home profile and a work profile for a laptop Ethernet adapter and switch between the two depending on the network to which you are attached.

You can display all the configured connections in Network Manager by running the following command:

```
# nmcli con show
NAME  UUID                                  TYPE            DEVICE
eth0  394f6436-5524-4154-b26e-6649b4d29027  802-3-ethernet  eth0
```

To show how **nmcli** can be used to set up a different connection profile, let's create a new connection for eth0:

```
# nmcli con add con-name "eth0-work" type ethernet ifname eth0
```

Then, a static IP address and default gateway can be configured, as shown here:

```
# nmcli con mod "eth0-work" ipv4.addresses ↵
"192.168.20.100/24 192.168.20.1"
```

You can run **nmcli con show** ***connection-id*** to display the current settings for a connection. Additional properties can be modified from the Network Manager command-line tool. For example, to add a DNS server to the eth-work connection, run

```
# nmcli con mod "eth0-work" +ipv4.dns 192.168.20.1
```

Finally, to switch to the new connection profile, run

```
# nmcli con up "eth0-work"
```

A connection can be prevented from starting automatically at boot with the following command:

```
# nmcli con mod "eth0-work" connection.autoconnect no
```

The nmtui Configuration Tool

As suggested by the name, this tool provides a text-based user interface and can be started from a command-line terminal. Just run the **nmtui** 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.

Press the DOWN ARROW key until Quit is highlighted, and then 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 **nmtui** tool.

The directives shown in Figure 3-11 are described in Table 3-7.

FIGURE 3-11

The differences between static and dynamic network configuration

```
[root@server1 ~]# diff ifcfg-eth0 /etc/sysconfig/network-scripts/ifcfg-eth0
2c2
< BOOTPROTO=dhcp
---
> BOOTPROTO=none
8a9,13
> IPADDR0=192.168.122.50
> PREFIX0=24
> GATEWAY0=192.168.122.1
> DNS1=192.168.122.1
> DOMAIN=example.com
10,11d14
< PEERDNS=yes
< PEERROUTES=yes
[root@server1 ~]#
```

TABLE 3-7 Network Configuration Directives in the /etc/sysconfig/network-scripts Directory

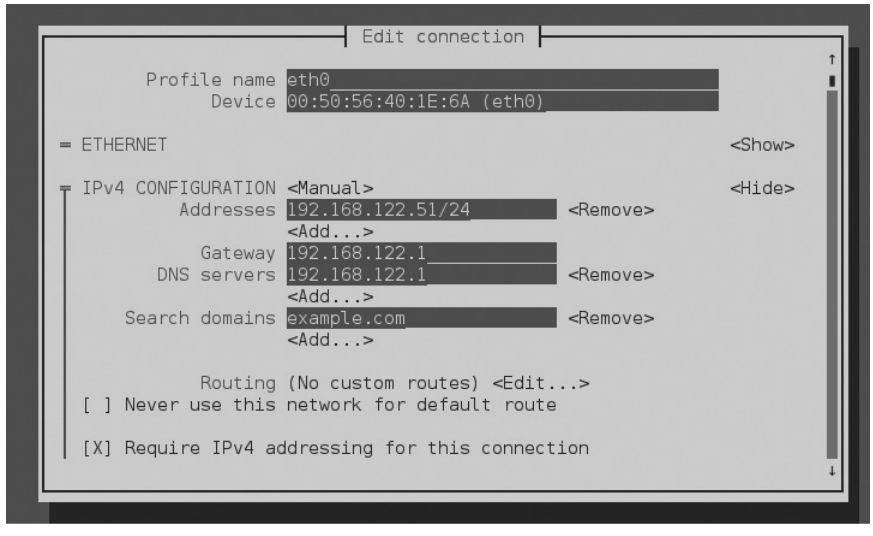
Directive	Description
DEVICE	Network device; eth0 is the first Ethernet network interface.
NAME	Name of the interface connection profile used by Network Manager.
UUID	Universal Unique Identifier for the device.
HWADDR	Hardware (MAC) address for the network device.
TYPE	Network type; should be set to "Ethernet" for an Ethernet device.
ONBOOT	Directive that specifies whether the network device is started during the boot process.
BOOTPROTO	May be set to "none" for static configuration or "dhcp" to acquire IP addresses from a DHCP server.
IPADDR0	Static IP address; additional IP addresses can be specified with the variables IPADDR1, IPADDR2, …
PREFIX	Network mask in CIDR format (i.e., /24)
GATEWAY0	IP address of the default gateway.
DEFROUTE	Binary directive to set the interface as the default route.
DNS1	IP address of the first DNS server.
DOMAIN	Specifies the domain search list in /etc/resolv.conf.
PEERDNS	Binary directive allowing the modification of /etc/resolv.conf.
IPV6INIT	Binary directive that enables the use of IPv6 addressing.
USERCTL	Binary directive to allow users to control a network device.
IPV4_FAILURE_FATAL	Binary directive; if set to "no", when connecting to IPv6 networks, allows the IPv6 configuration to complete if the IPv4 configuration fails.

EXERCISE 3-2

Configure a Network Card

In this exercise, you'll configure the first Ethernet network card with the Network Manager text-based user interface tool. All you need is a command-line interface. It doesn't matter whether the command line is in the GUI or a virtual terminal. 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 interface names such as em1, substitute accordingly. (Hint: use the **cp** and not the **mv** command.)
2. Run the **nmtui** command.
3. In the menu that appears, Edit a Connection should be highlighted. If necessary, press the ARROW or TAB keys until it is. Then press ENTER.
4. In the screen that appears, the first Ethernet network card should be highlighted. When it is, press ENTER.
5. In the Edit Connection window shown here, the Automatic option may be selected under IPv4 Configuration. If so, highlight it and press ENTER; then select Manual.



6. Highlight the Show option at the right of IPv4 Configuration and press ENTER. This will expand the current IPv4 settings.

7. 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.
8. You're taken back to the device screen. Make sure 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 **ip addr show eth0** and **ip route** commands. The configuration of the network card and the associated routing table should reflect the new settings.
10. To restore the original configuration, restore the ifcfg-eth0 file to the /etc/sysconfig/network-scripts directory and restart the network with the **systemctl restart network** command.

The Network Manager Connections Editor

Now you'll work with the default graphical network management tool for RHEL 7, the Network Manager Connections Editor 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.

The Network Manager Connections Editor is not really new, because it has been in use in Fedora for several years. It only runs in the GUI. To start it, you can run the **nm-connection-editor** command. This opens the Network Connections tool shown in Figure 3-12.

FIGURE 3-12

The Network Manager Connections Editor

FIGURE 3-13

Editing an Ethernet connection in the Network Manager Connections Editor

As you can see from the figure, the tool lists the connection profile for the first Ethernet network interface. It also supports the configuration of other types of networks, including wireless, mobile broadband cards (such as those used to connect to 3G and 4G networks), 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.

Highlight the connection profile of the first Ethernet device (eth0) and click Edit; then select the IPv4 Settings tab. It'll open the window shown in Figure 3-13. Unless previously configured, it assumes that the network interface will receive its configuration settings from a DHCP server.

Click the Method drop-down text box. Although 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, here are the appropriate options:

- **IP address** 192.168.122.50
- **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** example.com
- **Require IPv4 Addressing for This Connection to Complete** Deselected

If properly entered, the configuration associated with the first Ethernet card is titled with the connection name listed in Figure 3-13. For that configuration, the settings are saved in the ifcfg-eth0 file in the /etc/sysconfig/network-scripts directory.

Configure Name Resolution

The final piece in network configuration is typically name resolution. In other words, does the local system have the information required to translate domain names such as mheducation.com to IP addresses such as 198.45.24.143?

Name resolution was easy when Unix was first being developed. When the predecessor to the Internet was first put into use, the worldwide computer network had four hosts, one computer at each of four different universities. It was easy to set up a static file with a list of each of their names and corresponding addresses. That file has evolved into what is known in Linux as /etc/hosts.

But now the Internet is more complex. Although you could try to set up a database of every domain name and IP address on the Internet in the /etc/hosts file, that would take almost forever and would not be a scalable solution. That's why most users set up connections to DNS (Domain Name Service) servers. On RHEL 7, that's still documented in the /etc/resolv.conf configuration file. As an RHCE, you need to know how to configure a caching-only DNS server; that subject is covered in Chapter 13. DNS server configuration is not an RHCSA requirement.

On smaller networks, some administrators set up the /etc/hosts file as a database for the name of each system and IP address on the local network. If desired, administrators could even set up a few IP addresses of domains on the Internet, although this setup would break if any of those Internet domains made DNS changes on their own.

But if you've configured a connection to a DNS server and systems in /etc/hosts, what's searched first? That's the purpose of the /etc/nsswitch.conf configuration file, which specifies the search order for various name-service databases, including hostnames.

Hostname Configuration Files

RHEL 7 includes at least four hostname configuration files of interest: /etc/hostname, /etc/hosts, /etc/resolv.conf, and /etc/nsswitch.conf. These four files, taken together, contain the local hostname, the local database of hostnames and IP addresses, the IP address of one or more DNS servers, and the order in which these databases are considered.

/etc/nsswitch.conf

The /etc/nsswitch.conf file 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 those configured in the /etc/resolv.conf file.

The resolver library also uses information in the /etc/host.conf file. The entry in this file is simply

```
multi on
```

which tells the system to return all IP addresses in /etc/hosts that are mapped to the same hostname, instead of returning only the first entry.

/etc/hosts

The /etc/hosts file is a static database of hostnames/FQDNs 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:

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.122.50 server1.example.com server1
192.168.122.150 tester1.example.com tester1
192.168.100.100 outsider1.example.org outsider1
```

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 one of the following commands:

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

If needed, substitute the IP address associated with the **nameserver** directive in your /etc/resolv.conf file. You can specify up to three **nameserver** directives in this file.

It may not be wise to edit the /etc/resolv.conf file directly. If you have configured DNS servers with another tool such as nmcli**, the Connections Editor, or with a DNS1 directive in a device configuration file, the Network Manager will overwrite any changes you make when directly editing that file unless you override this behavior with PEERDNS in the ifcfg file.**

Hostname Configuration Options

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

```
tester1.example.com
```

Of course, you can modify the value of the hostname with the **hostname** ***newname*** command. However, this change is only temporary and is not reflected in the /etc/hostname file. To make the change persistent, use the **hostnamectl set-hostname** ***newname*** command.

SCENARIO & SOLUTION	
Networking is down.	Check physical connections. Run **ip link show** to check active interfaces. Run the **systemctl status network** command.
Unable to access remote systems.	Use the **ping** and **traceroute** commands to test access to local and then remote IP addresses.
Current network settings lead to conflicts.	Check network device configuration in the /etc/sysconfig/network-scripts files. Review settings with the Network Manager Connections Editor.
Network settings not consistent.	Check network device configuration in the /etc/sysconfig/network-scripts files. Review settings with the Network Manager Connections Editor. The scenario suggests a desire for a static network configuration, so review accordingly.
Hostname is not recognized.	Review /etc/hostname, run the **hostname** command, and review /etc/hosts for consistency.
Remote hostnames not recognized.	Review /etc/hosts and /etc/nsswitch.conf. Check /etc/resolv.conf for an appropriate DNS server IP address. Run the **dig** command to test the DNS resolution.

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. Although no shell is specified in the objectives, the default shell in most Linux distributions, including RHEL 7, 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, searched, and modified with a variety of commands. Linux text files can be analyzed as streams of data that can be interpreted and processed. To edit a text file, you need a text editor such as vim or gedit.

Documentation online within Linux is extensive. It starts with command switches such as **-h** and **--help**, which 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 addresses. Network review and configuration commands include **ip**, **ifup**, **ifdown**, and **dhclient**. Additional related commands include **ss**, **ping,** and **traceroute**. 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 **nmcli** and **nmtui** commands at the console and the Network Manager Connections Editor.

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 virtual terminals are available by default; if the GUI is installed, it takes over the first virtual terminal.
- ❑ 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 such as **pwd** and **cd** can help navigate directories.
- ❑ Concepts such as 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 such as **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**, **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** and **--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 private LAN.
- The subnet mask (also known as netmask or prefix) is used to find the network and host parts of an IP address.
- Tools such as **ping**, **traceroute**, **tracepath**, **ip**, and **ss** can help you diagnose problems on the LAN.
- Name resolution configuration files such as /etc/resolv.conf determine how a system finds the right IP address; the /etc/resolv.conf file may be configured from a DHCP server with the **dhclient** command or by Network Manager.

Network Configuration and Troubleshooting

- Individual network devices are configured in the /etc/sysconfig/network-scripts directory.
- Network configuration tools include the console-based **nmcli** and **nmtui** commands as well as the Network Manager Connections Editor.
- Name resolution 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 10 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 default full path to the configuration file associated with the eth0 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, the labs in this and future chapters are available from the DVD that accompanies the book in the Chapter3/ subdirectory in .doc, .html, and .txt formats. In case you haven't yet set up RHEL 7 on a system, refer to Chapter 1 for installation instructions.

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

A 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 that 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 10 lines of the /var/log/messages file is **tail -n 10 /var/log/messages**. Because 10 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 because 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 **ip addr add 192.168.100.100/24 dev eth0**.

Network Configuration and Troubleshooting

11. The full path to the configuration file with the hostname of the local system is /etc/hostname.
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 use different connection profiles in Network Manager, you would find a file for each connection profile within the /etc/sysconfig/network-scripts directory.

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 if **NETWORKING=no** is 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 sets 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 a slightly different name, 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 filename.

Lab 4

In this lab, you 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.example.com server1
192.168.122.150  tester1.example.com tester1
192.168.100.100  outsider1.example.org outsider1
```

It doesn't matter that the systems are on different IP networks. As long as there's a routing path between systems, the data in each /etc/hosts file will work. Also, 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 we set up a web server on the 192.168.122.50 system, we could add the following entry to /etc/hosts:

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

Labs 5 and 6

Be sure to learn the common settings associated with directives in the configuration file, such as HWADDR, BOOTPROTO, and DNS1.