Linux Recipes for Oracle DBAs

ISBN-13 (pbk): 978-1-4302-1575-2

ISBN-13 (electronic): 978-1-4302-1576-9

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jonathan Gennick
Technical Reviewers: Bernard Lopuz, Charles Kim
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Kylie Johnston
Copy Editor: Kim Wimpsett
Associate Production Director: Kari Brooks-Copony
Production Editor: Elizabeth Berry
Compositor: Susan Glinert Stevens
Proofreader: Nancy Sixsmith
Indexer: Carol Burbo
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

The source code for this book is available to readers at http://www.apress.com.

CHAPTER 1

Getting Started

Linux continues to gain market share as a reliable 24/7 mission-critical enterprise server platform. A rapidly increasing number of employers are specifically seeking database administrators (DBAs) with Linux expertise. In fact, as a database administrator, it's inevitable that you'll someday find yourself using Linux servers to store your data. You will be responsible for ensuring that your database is working symbiotically with the underlying Linux operating system. Managers will look to you to guarantee that corporate databases are competently implemented and maintained.

DBAs are crucial members of every information technology team (well, that's our unabashed and biased opinion). Database administrators are responsible for mission-critical tasks such as the following:

- Installing software and creating databases
- Providing a highly scalable and well-behaving database environment
- Monitoring and maintaining company databases
- Ensuring that your corporate data is backed up, secured, and protected
- Troubleshooting system performance and availability issues
- Being the holistic source of database engineering information

DBAs must possess a combination of database and operating system (OS) expertise. It's a fact that you (the DBA) cannot architect, implement, and maintain a large, high-transaction database environment without being intimately familiar with the underlying operating system. In many situations, the OS is your only conduit to the database. Therefore, it's imperative that you *must* be particularly knowledgeable of the operating system to competently perform your database administration duties.

When first building a database, DBAs have to be able to specify solid, reliable, and scalable system configurations. Furthermore, DBAs often find themselves as the cog between system administrators, (damn) users, network administrators, managers, and corporate executives. If you're working (after the fact) on a poorly designed system, you have to possess the tools to diagnose and resolve bottlenecks in the entire stack of technology. Regardless of the source of the <fill in the blank> issues, team members often look to the seasoned database administrator to resolve systemic performance and availability concerns. We know this to be true because we live it every day (including nights and weekends).

The information in this book will enable you to function as an expert DBA in performing key responsibilities. We provide direct answers to specific problems regarding Oracle database technology and the Linux operating system. The recipes in this first chapter assume you know nothing about Linux and cover situations that you'll be presented with when first connecting to a server and using operating system commands. If you are already fairly experienced with the Linux operating system, then you may want to skip this chapter.

In this chapter, we start by walking you through some of the most common methods for logging on to a Linux server. We then cover the basics of running Linux commands and detail how to use the built-in help and online documentation. We finish the chapter by showing techniques for correcting command-line mistakes and resetting your terminal screen.

■ **Note** You might be wondering, what is the difference between Linux and Unix? On the surface, Linux and Unix are mostly (83.1 percent) identical in commands and syntax. However, as a DBA, you must understand the nuances of the Linux kernel and how it impacts the database. These important concepts are explained in later chapters in this book.

1-1. Connecting Securely to a Remote Server

Problem

You want to securely connect over the network to your remote Linux database server.

Solution

This solution shows how to download and use the PuTTY application to initiate secure remote connections over the network:

1. To get started, first open your favorite Internet search engine, and search for the string *PuTTY download*. You should see information on your screen similar to Figure 1-1.

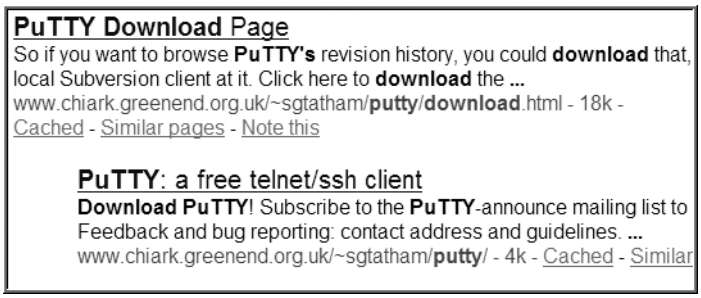

Figure 1-1. *Internet page with link to PuTTY download*

2. Click the link to the PuTTY download site. In this example, that link is titled "PuTTY Download Page." On the PuTTY download site, you'll find links that will allow you to download the PuTTY application. You can download just the `putty.exe` file or all utilities available via the `putty.zip` file.

3. After you have downloaded the desired files, navigate on your personal computer to the directory where you downloaded the PuTTY utility. You should see a screen similar to Figure 1-2. Double-click the PuTTY icon to start the connection utility.

Figure 1-2. *PuTTY application icon*

4. Figure 1-3 shows a partial screen shot of what you are presented with next. From this screen you can enter the hostname or IP address and connection port of the remote server to which you want to connect. Enter the connection details of your Linux database server, and click the Open button to initiate a remote connection. If you are not sure about the connection information, then contact your system administrator for details.

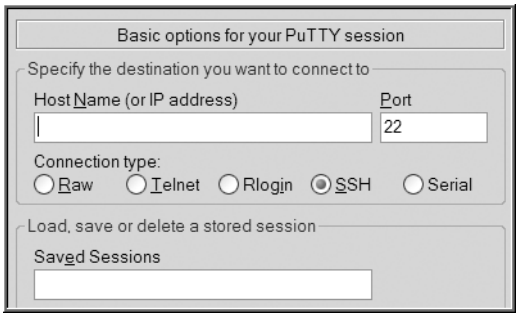

Figure 1-3. *PuTTY connection details*

5. Once you have connected to your database server, you should see the screen shown in Figure 1-4. Enter your username and password (contact your system administrator if you don't know your username and password). From here you can run Linux shell commands to perform tasks on your database server.

Figure 1-4. *Linux server logon screen*

How It Works

PuTTY is a free open source utility that allows you to create a secure shell (SSH) connection to your remote database server. This utility is popular because it's a free, easy-to-use application that allows you to connect securely over the network to remote Linux database servers. This tool allows you to store your server preferences and connection information, which eliminates the need to retype lengthy hostnames or IP addresses.

■ **Note** Other utilities also allow you to initiate remote connections via a secure shell. For example, the Cygwin/X application is a popular Windows-based implementation of the X Window System. This Cygwin/X utility allows you to run X applications on your Windows desktop and start remote secure shell connections to your database server.

You can also use PuTTY to connect via proxy servers and SSH tunneling. Examples of doing this are explained briefly in the next sections.

Connecting via a Proxy Server

Many companies require all their Internet connections to pass through a proxy server for security and performance reasons. To use PuTTY to connect via a proxy server, open the PuTTY Configuration dialog box, and click the Proxy node under the Connection category, as shown in Figure 1-5.

Select HTTP for the proxy type, and provide the hostname or IP address of the proxy server, as well as the corresponding port number. Save the changes of your PuTTY configuration for future use.

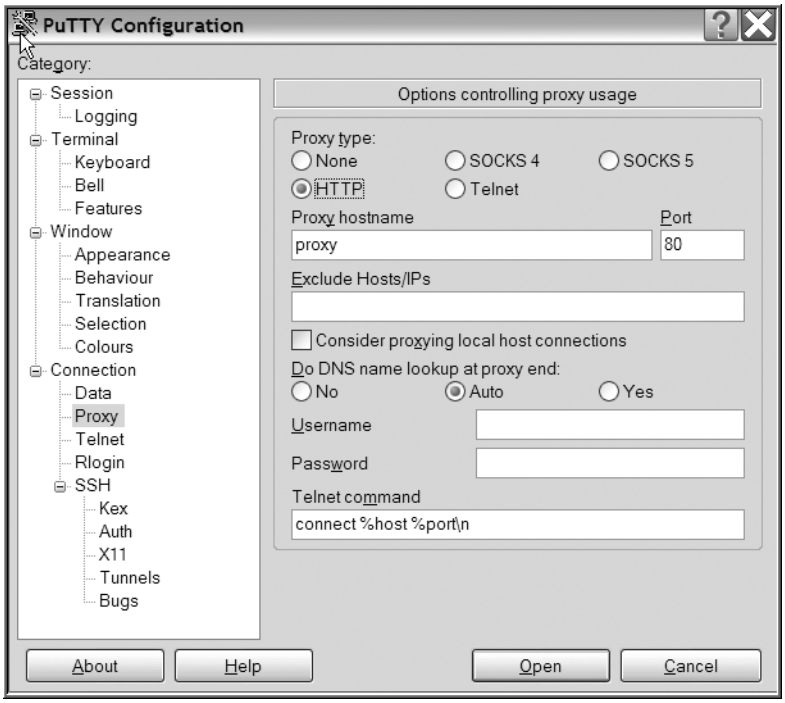

Figure 1-5. *PuTTY—proxy server configuration*

Connecting via SSH Tunneling

You can also use PuTTY to tunnel (also called *port forwarding*) to a remote server. To use tunneling, open the PuTTY Configuration dialog box, as shown in previously in Figure 1-3, and then choose the SSH as the connection type. Next, provide the hostname or IP address of the designated SSH server, as well as the SSH port number (the default is 22). Afterward, click the Connection node, then the SSH node, and finally the Tunnels node, as shown in Figure 1-6.

In the "Source port" field under the "Add new forwarded port" section, provide the port number you will connect to at your Windows client. In the Destination field, provide the hostname or IP address of the Linux database server, as well as the port number. Save the changes of your PuTTY configuration for future use.

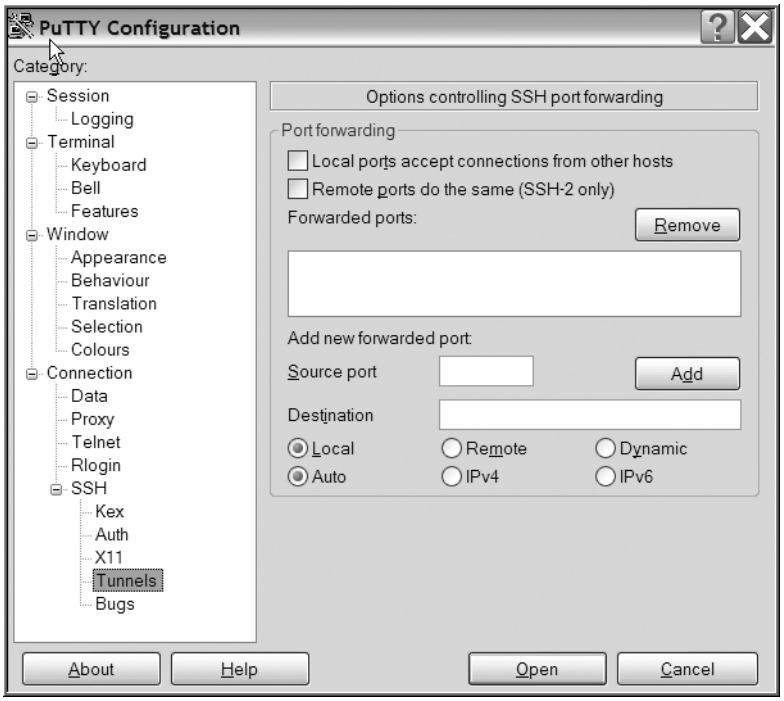

Figure 1-6. *PuTTY—SSH tunneling configuration*

1-2. Logging On Remotely via the Command Line

Problem

Your system administrator has just provided you with a username and password to your database server. You now want to log on to a Linux server via a command-line utility such as `telnet` or `ssh`.

Solution

This example assumes you can access a terminal from which you can initiate an `ssh` command. Depending on your environment, your "terminal" could be a PuTTY session (see recipe 1-1) on your home PC or workstation. Ask your system administrator for help if you're not sure how to start a terminal session (this can vary quite a bit depending on your working environment).

In this next line of code, the username is `oracle`, and the hostname is `rmoug1`:

```
$ ssh -l oracle rmoug1
```

If `ssh` successfully locates the database server, you should be prompted for a password (the prompt may vary depending on the version of the operating system):

```
oracle@rmoug1's password:
```

For security purposes, your password will not be displayed as you type it. After typing your password, hit the Enter or Return key to complete the logon process.

■ **Tip** If you think you have made a mistake while entering your password, press Ctrl+U to erase all invisible text from the password line. This technique will save you time and prevent many accidental failed logins. Alternatively, you can try to use the Backspace key or the Delete key to erase any text you've entered.

How It Works

By default, most SSH servers listen on the TCP port 22. If your system administrator has set up the server to listen on a different port, then you will have to explicitly specify it with the `-p` (port) option. This example connects as the `oracle` user to the `rmoug1` server on port 71:

```
$ ssh -p 71 -l oracle rmoug1
```

After entering a valid username and password, your system may display information such as the last time you logged on, from what machine you initiated the connection, whether your account has unread mail, and so on. Additionally, if your system administrator has entered any text within the `/etc/motd` file (message of the day), then that information will also be displayed. The following text is a typical login message:

```
Last login: Tue Dec 25 15:31:31 2007 from 63-227-41-191.hlrn.qwest.net
```

After your username and password have been successfully authenticated by the Linux server, you should see a `$` (dollar sign) prompt:

```
$
```

The `$` character signifies you are at the shell command-line prompt. The `$` character is the default command-line prompt for most Linux/Unix systems. All command-line examples in this book will show the `$` prompt. You don't have to type the `$` prompt as part of any of the example commands in this book.

Your system administrator may have configured your account to display a different prompt (other than the `$` character). See recipe 2-6 for changing your command-line prompt from something other than the default.

■ **Note** If you are logged as the `root` (sometimes called `superuser`) account, the default command-line prompt is the # character.

Some servers accept remote connections from `telnet` clients. For security reasons, we recommend you do not use `telnet` to initiate a logon to a server over the network. The `telnet` utility does not use encryption and is vulnerable to hackers snooping on the network. Whenever possible, you should use the secure `ssh` tool for remote connections. However, you may

occasionally have to use `telnet` because `ssh` isn't available. The following example uses `telnet` to log on to a remote server over the network:

```
$ telnet -l oracle dbsrver
```

1-3. Logging Off the Server

Problem

You want to log off the server.

Solution

Three methods for logging out of the database server are covered in this solution:

- Pressing Ctrl+D
- Typing `exit`
- Typing `logout`

The quickest way to log off is to press Ctrl+D. This will immediately log you off your server. In this next example, the keys Ctrl and D are pressed at the same time:

```
$ Ctrl+D
```

You should now see a message like this:

```
Connection to <your server> closed.
```

You can also type the `exit` command to log off your database server:

```
$ exit
```

You should now see a message like this:

```
Connection to <your server> closed.
```

You can also type the `logout` command to log off the system:

```
$ logout
```

You should now see a message like this:

```
Connection to <your server> closed.
```

How It Works

If you have started a subshell within an existing shell session, then the logout techniques described in the "Solution" section will exit you from only the innermost shell. For example, say you have logged on to your Linux server and then issued the following command:

```
$ bash
```

You now have started a subshell. If you want to exit the subshell, use one of the techniques described in the "Solution" section:

```
$ exit
```

Similarly, if you have issued an `su` command to switch to another user, then when exiting from that session, you will be returned to the shell from which you initiated the `su` command.

It's a good security practice to log out of your operating system session if you plan on being away from your terminal. As a DBA, you'll typically find yourself logged on to the server as the user who owns the database binaries (usually the `oracle` operating system account). The `oracle` account is like a database `superuser` account.

The database operating system account can do potentially damaging operations like drop databases, remove database files, and so on. Logging out ensures that your database operating system account isn't compromised.

■ **Tip** Set the `TMOUT` variable to limit the amount of idle time a session can have before it is automatically logged off. This parameter can be set globally in the `/etc/bashrc` file. See recipe 2-5 for automatically setting variables when logging on to a server.

1-4. Running a Command

Problem

You're new to Linux, and you want to run a shell command from the operating system prompt.

Solution

Linux commands are run from the command line by typing them and pressing the Enter or Return key. This example uses the `df` (disk-free) command to display the amount of unused disk space on the database server:

```
$ df
```

By default, the output of the commands is displayed on your screen (standard output). After running the `df` command, you should see some output similar to this:

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda2            236141180   7026468 217119380   4% /
/dev/sda1               101086      9448     86419  10% /boot
none                   1037452         0   1037452   0% /dev/shm
```

How It Works

When you log on to a Linux or Unix box, you are by default placed at the command line. The command line is where you type shell commands to accomplish a given DBA task. The default command-line prompt for most systems is the $ character.

You can modify the default behavior of a command by running it with one or more *options* (sometimes called *flags* or *switches*). This next example shows using the `ls` (list) command with the `-a` option to display all files:

```
$ ls -a
```

Commands may also take *arguments*. Arguments typically designate a file or text that the command should use. When running a command, arguments are usually placed after the options. This next example uses the `df` (disk-free) command with the `-h` (human-readable) option and uses the argument of `/dev/sda2` (which is a directory):

```
$ df -f /dev/sda2
```

This book does not detail the use of various graphical user interfaces (GUIs) or browser user interfaces (BUIs). These tools are great for situations when you don't know the actual command. As useful as these graphical tools are, we strongly recommend you explore using Linux commands from the command-line prompt. As a DBA, you will encounter situations where the GUI/BUI tool doesn't do everything you need in order to accomplish a task. For some problems, you will need access to the command line to debug and troubleshoot issues. Many complicated or custom DBA tasks require that you be proficient with command-line programming techniques (see Chapter 7 for more information on shell scripting).

If you don't know the appropriate shell commands and their features, then you'll potentially waste a lot of time trying to solve a problem when it could have easily been resolved had you known which tools and options were available.

GOLDEN HAMMER RULE

The Golden Hammer Rule can be stated as "When the only tool you have is a hammer, everything looks like a nail." What does that mean? When people find a tool that solves a problem, they have a natural tendency to use that tool again and again to solve other problems. This might be because once you're familiar with a given tool or technique, you'll continue to use it because it's available, you've had training with the tool, and you've developed a skill set.

Nothing is wrong with that approach per se. However, if you want to be a more marketable DBA, then you should expand your horizons from time to time by learning new skills and investigating up-to-date methods for solving problems. In today's ever-changing technology environment, the DBA with the most current skills is often the one who survives the longest.

It's getting harder and harder to find database environments that don't use Linux. As a DBA, you should take the initiative to learn about Linux and how this technology is used by companies around the world to provide cost-effective information technology solutions. In today's world, Linux is an operating system that every DBA needs to be familiar with.

1-5. Getting Help

Problem

You want to find more information about how to use a shell command.

Solution

One extremely nice feature of Linux is that there are several options for quickly obtaining more information regarding shell commands. Listed next are command-line help features that are readily available on most Linux systems:

`man`: Read the online manual for a command.

`whatis`: View a brief description of a command.

`which` or `whereis`: Find a tool.

`--version`: Display the version.

`--help`: Show help.

`apropos`: Display manual page documentation.

`info`: List extensive documentation.

Tab key: Show available commands.

Each of the help methods in the previous list will be described in the following subsections.

Reading Manual Pages

The `man` (manual) page for a command displays online documentation for almost every shell command. The following command displays information about `man`:

```
$ man man
```

Here's a partial listing of the output:

```
NAME       man - format and display the on-line manual pages
SYNOPSIS   man  [-acdfFhkKtwW]  [--path] [-m system] [-p string] [-C config_file]
       [-M pathlist] [-P pager] [-S section_list] [section] name ...
```

The `man` command uses a screen pager—usually the `less` command—to display the help page. The `less` utility will display a `:` (colon) prompt at the bottom-left corner of the screen. You can use the spacebar to go to the next page and the up and down arrows to scroll through

the documentation line by line. Table 1-1 lists the `less` command options available to you while viewing `man` pages. Press the Q key to exit the `man` utility.

Table 1-1. *The `less` Command Options Available While Viewing `man` Pages*

Keystroke	Action
J, E, or down arrow	Move down one line.
K, Y, or up arrow	Move up one line.
Up arrow	Move up one line.
Down arrow	Move down one line.
/<string>	Search for <string>.
N	Repeat the previous search forward.
Shift+N	Repeat the previous search backward.
H	Display help page.
F, spacebar, or Page Down	Move down one page.
B or Page Up	Move up one page.
Q	Exit `man` page.

The `man` pages are usually divided into ten sections. The `man` command will display the first `man` page match it finds for a specified command. The following list shows the section number and type of commands documented in each section:

1. User commands
2. System calls
3. Subroutines (library functions)
4. Devices
5. File formats
6. Games
7. Miscellaneous
8. System administration
9. Local
10. New

Sometimes a Linux utility will be documented in more than one `man` section. To view all `man` documentation available for a tool, use the `-f` option (this is equivalent to running the `whatis` command). This example views all `man` pages available with the `cd` command:

```
$ man -f cd
```

From the output, you can see that `cd` is documented in two different `man` sections:

```
cd                  (1p)  - change the working directory
cd [builtins]       (1)  - bash built-in commands, see bash(1)
```

Sometimes DBAs are confused when they type `man cd` and are presented with the Bash shell's built-in documentation. To view the `man` documentation specific to the `cd` utility, then specify the `1p` page:

```
$ man 1p cd
```

To scroll through all `man` sections associated with a command, use the `-a` option. When in this mode, press the Q key to advance to the next `man` section of information.

CAPTURING MAN PAGES IN A TEXT FILE

Sometimes it's helpful to capture the output of a `man` command in a file that can be used later to search and scroll through with a text editor. The following command writes the output of the `man` page for the `find` command to a file named `find.txt`:

```
$ man find >find.txt
```

However, if you inspect the output of the `find.txt` file, you'll notice that it contains unreadable characters that are produced from the `man` page output. Run the following command to clean up the output of the `man` page:

```
$ man find | col -b >find.txt
```

The previous command takes the output from the `man` command and sends it to the `col -b` (postprocessing filter) command. This filtering command will remove the unreadable backspace characters from the `man` page output and make it human-readable.

Viewing a Brief Description of a Command

If you're new to Linux (or if you're old to Linux and have forgotten the material), use the aptly named `whatis` command to answer the question, "what is" a command's basic information? The `whatis` command lists the first line of text from the `man` (manual) page. This next example shows how to use the `whatis` command to find more information about the `pwd` command:

```
$ whatis pwd
pwd                 (1)  - print name of current/working directory
pwd [builtins]      (1)  - bash built-in commands, see bash(1)
```

The number (enclosed by parentheses) specifies the section of the `man` page where you can find the command. When you see multiple lines listed by `whatis`, this indicates the command is documented in more than one location in the `man` pages. The output also indicates there is a built-in Bash version of the command (see recipe 2-15 for more details about built-in commands).

Another interesting use of the `whatis` command is to view a one-line description of commands in the `/bin` directory. This example uses `whatis` with `ls`, `xargs`, and `less` to view, one page at a time, the descriptions of all commands in the `/bin` directory:

```
$ cd /bin
$ ls | xargs whatis | less
```

The previous line of code will first list the files in the `/bin` directory; then, the output of that is piped to the `xargs` command. The `xargs` command will take the output of `ls` and send it to the `whatis` utility. The `less` command will display the output one page at a time. To exit from the documentation (displayed by `less`), press the Q key (to quit).

■ **Note** The `whatis` command is identical to the `man -f` command.

Locating a Command

Use `which` or `whereis` to locate the executable binary file of a command. This next line of code locates the binary `man` executable file:

```
$ which man
/usr/bin/man
```

The output indicates that `man` is located in the `/usr/bin` directory. The `whereis` command locates the binary file, source, and manuals for a particular utility. The following example displays the location of the `echo` command and its corresponding `man` documentation files:

```
$ whereis echo
echo: /bin/echo /usr/share/man/man1/echo.1.gz /usr/share/man/man1p/echo.1p.gz
```

Getting the Version

Sometimes it's useful to see the version of a command. This next example uses the `--version` option to display the version of the `who` command:

```
$ who --version
who (coreutils) 5.2.1
```

Showing Help

Use the `--help` option to quickly display basic information about a tool's usage and its syntax. This next example demonstrates how to get help for the `df` command:

```
$ df --help
```

Here's a partial listing of the output:

```
Usage: df [OPTION]... [FILE]...
Show information about the filesystem on which each FILE resides,
or all filesystems by default. Mandatory arguments to long options are mandatory
for short options too.
  -a, --all             include filesystems having 0 blocks
  -B, --block-size=SIZE use SIZE-byte blocks
  -h, --human-readable  print sizes in human readable format (e.g., 1K 234M 2G)
```

Depending on the version of the operating system, all Linux operating system commands may not have a `--help` option. If this is the case, you'll have to use one of the other documentation sources listed in this recipe.

Finding Manual Page Documentation

If you can remember only partially the name of the utility you seek, then use the `apropos` command to find more documentation. The `apropos` command is similar to `whatis` except that it searches for any string that matches your input. For example, the following example searches the `whatis` database for the string `find`:

```
$ apropos find
```

Here's a partial snippet of the output:

```
find                (1)  - search for files in a directory hierarchy
find                (1p)  - find files
find2perl           (1)  - translate find command lines to Perl code
findchip            (8)  - checks the FIR chipset
findfs              (8)  - Find a filesystem by label or UUID
```

The previous output shows you that many different types of `find` commands are available. Use the `man` command (previously discussed in this recipe) to view more information about a particular `find` command. The number in the second column (in the parentheses) lists which section of the `man` page the documentation is contained in.

■ **Note** The `apropos` command is equivalent to `man -k`.

Listing Extensive Documentation

The `info` utility often contains extensive documentation on many Linux commands. To view all documents available, type `info` with no parameters, as shown here:

```
$ info
```

Figure 1-7 shows the `info` introduction screen.

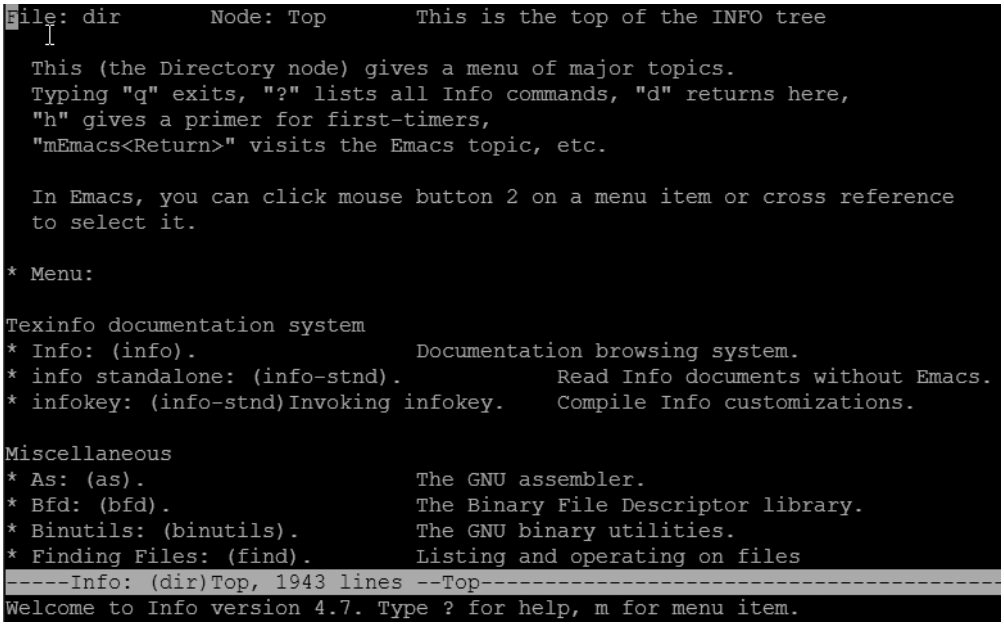

Figure 1-7. *The* `info` *command introduction screen*

Once within the utility, use the N key to go to the next section. The P key will take you to a previous section. Any line that starts with an asterisk (*) is a link to other sections (nodes) in the document. To go to a linked document, navigate to the line containing the asterisk and press the Enter or Return key. Press the Q key to exit the `info` page. Table 1-2 lists some of the commonly used navigational `info` commands.

Table 1-2. *Commonly Used Navigation Keystrokes Within the* `info` *Utility*

Keystroke	Action
N	Move to the next section.
P	Move to the previous section.
Return	Move to a linked document.
Q	Exit.
?	List all commands.
D	Return to introduction page.
H	Go to the tutorial.

You can also view information regarding specific commands. This next example starts the `info` utility to display help for the `cpio` command:

```
$ info cpio
```

To view a tutorial on `info`, type the following:

```
$ info info
```

Showing Available Commands

You can use the Tab key to show all executable files that start with a certain string. For example, if you want to view all commands that start with the string `ls`, then type `ls` and hit the Tab key twice (with no space between the `ls` command and pressing the Tab key):

```
$ ls<Tab><Tab>
ls           lsb_release  lsnrctl      lspgpot
lsattr       lshal        lsnrctl0     lss16toppm
```

You should hear a bell sound (sometimes called a *beep*) after you press the first Tab. After pressing the second Tab, the Bash shell will attempt to find all commands that start with `ls` that are located in any directories contained in the `PATH` variable.

This feature of automatically looking for files is known as *command completion*. See recipe 2-2 for more details on command completion.

How It Works

The "Solution" section of this recipe contains some of the most useful information you'll need to elevate your Linux skills. You should take some time to familiarize yourself with all the helpful techniques described in this recipe.

You'll find that Linux has extensive utilities for easily viewing command documentation. Using these built-in help features allows you to quickly find basic syntax and usage of a given shell command. We suggest you become particularly familiar with `man` and `info`. You'll use these informational tools on a regular basis.

1-6. Correcting Command-Line Mistakes

Problem

You're a typical DBA, and you often mistype things on the command line. You wonder whether there are command-line tools to correct your typing mistakes.

Solution

Press Ctrl+_ at the same time to undo what you just typed at the command line. Notice that you'll have to use the Shift key to get the underscore (_) character. If you type a long command string, then pressing Ctrl+_ will erase everything to the left of the prompt. If you have backspaced over a command, pressing Ctrl+_ will undo what you have backspaced over.

How It Works

Other keystrokes are also available to help you undo what you just typed. For example, you can use Ctrl+T to transpose two characters just to the left of the prompt (ensure there is no space between the command characters and Ctrl+T). This next bit of code will use Ctrl+T to transpose the last two characters of the letters `pdw`:

```
$ pdw Ctrl+T
```

You should now see the following:

```
$ pwd
```

Table 1-3 summarizes the commands available to you to correct typing mistakes at the command line.

1-7. Clearing the Screen

Table 1-3. *Command-Line Keystrokes to Correct Typing Errors*

Keystroke	Action
Ctrl+_	Undo what was just typed in.
Ctrl+U	Clear out everything to the left of the prompt.
Ctrl+T	Transpose two characters that are immediately to the left of the prompt.
Alt+T	Transpose two words that are to the left of the prompt.

Problem

Your screen has become cluttered with command output. You want to clear the screen of any previously displayed text or command output.

Solution

Either use the `clear` command or press Ctrl+L to clear your terminal screen. The `clear` command works in most Linux/Unix shells, and it does what you would expect—it clears the screen. Simply type the command as shown with no options or arguments:

```
$ clear
```

Another method for clearing the screen (on some Linux systems) is to press Ctrl+L:

```
$ Ctrl+L
```

One nice feature about Ctrl+L is that you can enter this command while typing other commands on the command line. Pressing Ctrl+L will clear the screen and retain any current commands you have entered on the command line. For example, say you are in the middle of typing a `find` command; you can enter Ctrl+L as shown here:

```
$ find . -name *.sql Ctrl+L
```

When you press Ctrl+L, it will clear the screen and place the command you are currently typing at the top of the screen. In this example, the `find` command appears at the top of the screen:

```
$ find . -name *.sql
```

How It Works

The `clear` command removes all output visible on your screen. This command retrieves environment information from the terminal database to determine how to clear the screen. To see more information on the terminal database, use the `man terminfo` command.

The Ctrl+L keystroke works with the Bash shell, and it may work with other shells depending on your version of the operating system. Unlike the `clear` command, Ctrl+L will retain whatever command you are currently typing and display that at the top of the cleared-out screen.

1-8. Resetting the Screen

Problem

Your screen has become cluttered with strange, unreadable characters. The `clear` and Ctrl+L commands don't seem to have any effect.

Solution

Try to use the `reset` command to restore the screen to a sane state:

```
$ reset
```

If the `reset` command doesn't work, then try the `stty sane` command:

```
$ stty sane
```

If that doesn't work, try exiting your terminal session and restarting it. That's not the ideal solution, but sometimes that's the only thing that will work.

How It Works

Sometimes your screen can become cluttered with unreadable characters. This might happen if you accidentally use the `cat` command to display the contents of a binary file. Use either the `reset` or `stty sane` command to restore your screen to a normal state.

The `reset` command is actually a symbolic link (see recipe 5-33 for more details on links) to the `tset` (terminal initialization) command. View the manual pages for `tset` for more details on this utility. The `reset` command is particularly useful for clearing your screen when a program abnormally aborts and leaves the terminal in unusual state.

The `stty` (set terminal type) command displays or changes terminal characteristics. If you type `stty` without any options, it will display the settings that are different from those set by issuing the `stty sane` command