VISUAL QUICKSTART GUIDE

UNIX AND LINUX

FOURTH EDITION

Learn Unix and Linux the Quick and Easy Way!

DEBORAH S. RAY
ERIC J. RAY

Table of Contents

3

Working with Your Shell

When you access a Unix system, the first thing you see is the prompt, called the *shell prompt*, which is where you interact with Unix. The shell determines how easily you can enter and reenter commands and how you can control your environment. What's cool about Unix is that you're not stuck with one shell—that is, on most systems you can choose to use shells that have different features and capabilities.

In this chapter, we'll look at your shell, show you how to change it, and get you started using a few of the more common shells.

Chapter Contents

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

Discovering Which Shell You're Using

When you first log in to your Unix account, you'll be using the default shell on your system. The default shell, its features, and its options depend completely on what your system administrator specifies. **Code Listings 3.1** and **3.2** show examples of how default shell prompts differ on two different systems.

To discover what shell you're using:

- `echo $SHELL`

  At your shell prompt, type `echo $SHELL` (capitalization counts!). This command tells Unix to display (echo) information about shell settings. This information, by the way, is contained in one of the environment variables, so the technical phrasing (which you might hear in Unix circles) is to "echo your shell environment variable."

  The system's response will be the full path to your shell—something like `/bin/zsh`, `/bin/bash`, or `/bin/ksh`.

✔ Tips

- You can also use `finger userid`, substituting your login name for `userid`, to find out more about your shell settings. You can substitute any other `userid` and see comparable information about the other account holders. See Chapter 7 for more about `finger`. (Some systems do not support `finger`, because `finger` can be a bit of a security hole.)
- You'll find more information about different shells and their capabilities throughout this chapter.

```
xmission> echo $SHELL
/bin/csh
xmission> finger ejray
Login name: ejray    In real life: "RayComm"
Directory: /home/users/e/ejray    Shell:
→ /bin/csh
On since Jul 23 06:58:48 on pts/16 from
→ calvin.raycomm.com
1 minute 28 seconds Idle Time
No unread mail
No Plan.
xmission>
```

Code Listing 3.1 This ISP account uses the `/bin/csh` shell by default.

```
[ejr@hobbes ejr]$ echo $SHELL
/bin/bash
[ejr@hobbes ejr]$ finger ejr
Login: ejr      Name: Eric J. Ray
Directory: /home/ejr    Shell: /bin/bash
On since Wed Jul 22 07:42 (MDT) on tty1
→ 3 hours 15 minutes idle
On since Thu Jul 23 08:17 (MDT) on ttyp0
→ from calvin
No mail.
Project:
Working on UNIX VQS.
Plan:
This is my plan—work all day, sleep all
→ night.
[ejr@hobbes ejr]$
```

Code Listing 3.2 On hobbes, a Linux system, the default shell is `/bin/bash`.

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

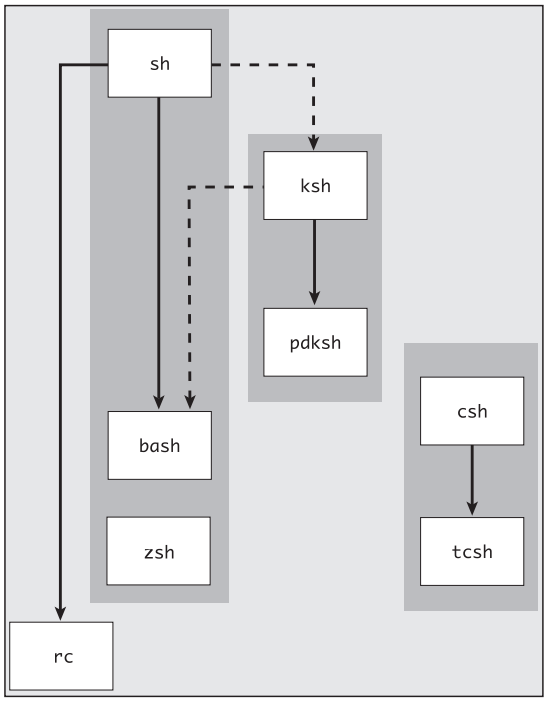

Figure 3.1 Shells fit neatly into a few "families" with the exception of a few stragglers. Each shell in a family shares many characteristics with the others in the same family.

Understanding Shells and Options

Depending on the particular Unix system you're using, you may have several shells available to you. **Table 3.1** describes a few of the more common ones. Each of these shells has slightly different capabilities and features. Keep in mind that the differences in shells do not affect what you can do in Unix; rather, they affect how easily and flexibly you can interact with the system.

You'll likely have `bash` as your shell, but you can change to one of many other shells fairly easily. As **Code Listings 3.3** and **3.4** show, you can start by finding out which shells are available to you. **Figure 3.1** shows some shells and how they relate to each other.

Table 3.1

Common Unix Shells

Shell Name	Features
`sh`	This shell, which is the original Unix shell (often called the Bourne shell), is fine for scripting but lacks a lot of the flexibility and power for interactive use. For example, it doesn't have features like command completion, e-mail checking, history, or aliasing.
`csh` and `tcsh`	This family of shells adds great interactive uses but discards the popular scripting support that `sh`-related shells offer in favor of a C programming-like syntax. Because of the C syntax, this shell is often just called the C shell. Unless you're a C programmer, these are not likely to be your best choices.
`ksh`, `bash`, and `zsh`	These provide a good blend of scripting and interactive capabilities, but they stem from different sources (`bash` is most similar to `sh`, hence the Bourne Again SHell name).

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

To see which shells are available to you:

- `cat /etc/shells`

  At the shell prompt, type `cat /etc/shells` to find out which shells are available to you. **Code Listings 3.3** and **3.4** show the results of this command on two different systems.

✔ Tips

- Before you go leaping forward through the next sections and changing your shell, you might check with your system administrator or help desk to find out which shells they support and sanction.
- If all else is equal in terms of support from your system administrator or help desk, and you have no clear preference, we'd suggest `zsh` as a first choice, with `bash` as a close second. For most purposes, either will be fine. Power users will like `zsh` better in the long run.
- Not all systems use `/etc/shells` to list acceptable shells—you may have to just look for specific shells, as shown later in this chapter.

```
[ejr@hobbes]$ cat /etc/shells
/bin/bash
/bin/sh
/bin/tcsh
/bin/csh
[ejr@hobbes]$
```

Code Listing 3.3 A minimal listing of available shells on a Unix system, including the basics but not too much in the way of choices.

```
xmission> cat /etc/shells
/usr/local/bin/tcsh
/bin/csh
/usr/bin/csh
/bin/ksh
/usr/bin/ksh
/sbin/sh
/usr/bin/sh
/usr/local/bin/zsh
/usr/local/bin/bash
/usr/local/bin/nologin
/usr/local/bin/terminated
/usr/local/bin/xmmenu.email
/usr/local/bin/xmmenu.noshell
/usr/lib/uucp/uucico
xmission>
```

Code Listing 3.4 These shells are available through an ISP. Notice the additional, custom shells that this ISP uses, including shells that provide special features such as not allowing logins.

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

```
[ejr@hobbes ejr]$ cat /etc/shells
/bin/bash
/bin/sh
/bin/tcsh
/bin/csh
/bin/zsh
[ejr@hobbes ejr]$ chsh
Changing shell for ejr.
Password:
New shell [/bin/bash]: /bin/zsh
Shell changed.
ejr@hobbes ~ $
ejr@hobbes ~ $ su - ejr
Password:
ejr@hobbes ~ $
```

Code Listing 3.5 You must remember the path to the shell to change shells on this system. Additionally, the password check helps ensure that only the account owner changes the shell.

Changing Your Shell with chsh

If you decide that you want to change your shell, you probably can, depending on how your system administrator has set things up. As **Code Listing 3.5** shows, you would do so using `chsh`. We usually change to `bash`.

To change your shell with chsh:

1. `cat /etc/shells`

   At the shell prompt, list the available shells on your system with `cat /etc/shells`.

2. `chsh`

   Enter `chsh` (for "change shell"). Code Listing 3.5 shows the system response. Some systems prompt for a password, and some don't.

3. `/bin/zsh`

   Type the path and name of your new shell.

4. `su - yourid`

   Type `su -` and your userid to log in again to verify that everything works correctly. If it doesn't, use `chsh` again and change back to the original shell or to a different one. If you can't change back, e-mail your system administrator for help.

continues on next page

✔ Tips

- After changing shells, you might have problems running some commands or have a prompt or display that's not as good as the original. That's likely a result of your default shell being carefully customized by your system administrator. You're probably on your own to set and configure your new shell, and Chapter 8 can help you do this.
- Some systems don't let users use `chsh` to change shells. If this is the case, you'll need to e-mail your system administrator and ask for a change, or see if there are alternative methods, as shown in **Figure 3.2**. You could also change your shell temporarily, as described in the next section.
- See "Changing Your Identity with `su`," later in this chapter, for more about the `su` command.

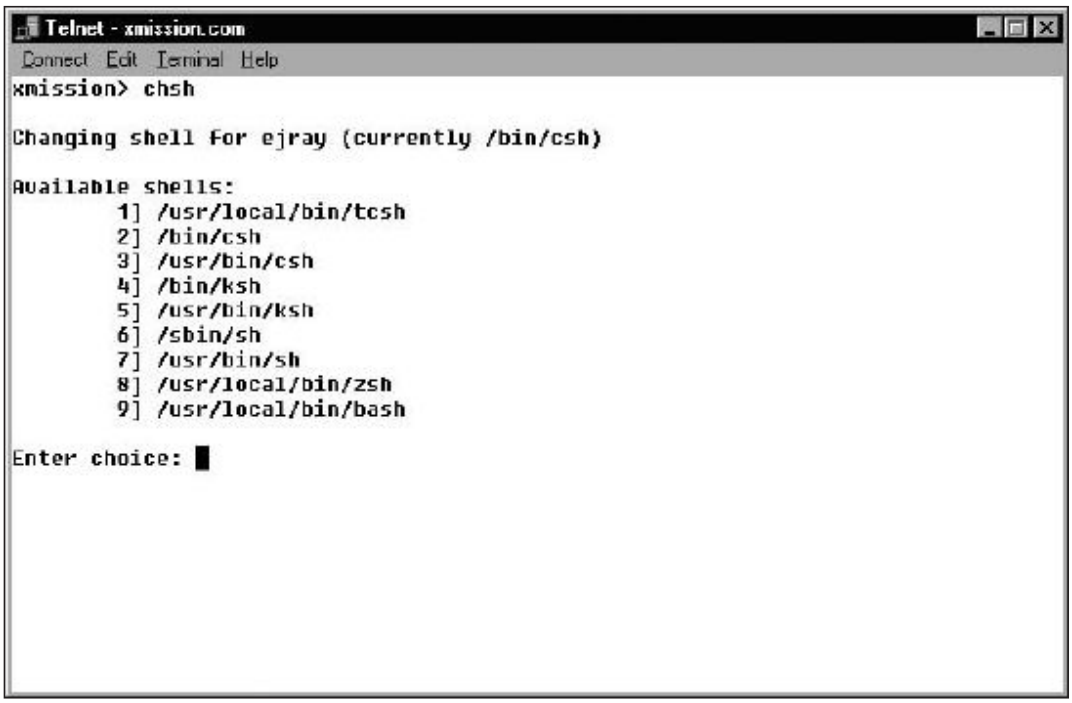

Figure 3.2 Some ISPs provide a handy interface for changing shells that lets users pick their new shells from a menu, like this one.

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

```
[ejr@hobbes]$ cat /etc/shells
/bin/bash
/bin/sh
/bin/tcsh
/bin/csh
[ejr@hobbes]$ ls /usr/local/bin/*sh
/usr/local/bin/pdksh
[ejr@hobbes]$
```

Code Listing 3.6 Checking the list of shells from `/etc/shells` and looking for other programs that end with sh is a good way to find all of the shells on the system.

```
[ejr@hobbes]$ /usr/bin/csh
ejr>
```

Code Listing 3.7 Type the shell name (which is really just another Unix command) to change shells.

Changing Your Shell Temporarily

You can change your shell temporarily by creating a subshell and using that instead of the original shell. You can create a subshell using any shell available on your Unix system. This means that you can look in the `/etc/shells` file and use a shell listed there, or you can use a shell installed elsewhere on the system (**Code Listing 3.6**).

To find out which temporary shells you can use:

1. `cat /etc/shells`

   At the shell prompt, type `cat /etc/shells` to find out which shells are listed in the shells file.

   If you don't find a shell you want to use in the shells file, look for other shells installed elsewhere on the system.

2. `ls /usr/local/bin/*sh`

   At the shell prompt, type `ls /usr/local/bin/*sh` to find additional shells in the `/usr/local/bin` directory. Note that not all programs that end with `sh` are shells, but most shells end with `sh` (Code Listing 3.6).

To create a temporary shell (subshell):

- `/usr/bin/csh`

  At the shell prompt, type the path and name of the temporary shell you want to use. In this case, we're using the `csh` shell, located at `/usr/bin/csh`. You might see a new prompt, perhaps something like the one shown in **Code Listing 3.7**.

To exit a temporary shell (subshell):

- `exit`

  At the shell prompt, type `exit`. You'll be returned to the shell from which you started the subshell. If you created more than one subshell, you'll have to exit all of them.

✔ Tips

- Using temporary shells is a great way to experiment with other shells and their options. We recommend using a temporary shell to experiment with the shells covered in this chapter.
- You can also often use Ctrl D to exit from a subshell, but this depends on the system configuration. Try it out and see.
- See Chapter 1, specifically the listings of directories containing programs, for other places to look for shells.

```
bash-2.00$ ls
Complete  NewProject  bogus2
→ ftp puppy
Completed News    dead.letter
→ mail   temp
Mail  access  files
 → public_html  testme
bash-2.00$ cd public_html/
bash-2.00$
```

Code Listing 3.8 In this example, we typed only the `ls` command followed by `cd pub` and pressed the Tab key; bash completed the command for us.

Using Completion in the bash Shell

One of the cool features of the `bash` shell is *command argument completion*. With this feature, you can type just part of a command, press Tab, and have `bash` complete the command for you (**Code Listing 3.8**).

To use completion in the bash shell:

1. `ls`

   Use `ls` to list the files in your current directory.

2. `cd pub` Tab

   Type a partial command, as shown here, and then press Tab to complete the command. In this example, we typed the `cd` command and part of the `public_html` directory (truncated to `pub` in the example), then pressed Tab to complete it (see Code Listing 3.8).

✔ Tips

- Completion works only if there's just one possible match to the letters you type before you hit Tab. For example, if you type `cd pu` (for `public_html`) and there's another subdirectory called `puppy`, the shell will beep and wait for you to type in enough letters to distinguish the two subdirectories.
- You can use the completion feature to complete commands, directory names within commands, and nearly anything else you might enter that's sufficiently unambiguous.

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

Viewing Session History in the bash Shell

Another cool feature of the bash shell is that it lets you easily reuse commands from your session history, which shows you the list of commands you've used during a session or in previous sessions (**Code Listing 3.9**). Viewing history is handy for reviewing your Unix session, using previous commands again (rather than retyping them), and modifying (rather than completely retyping) complex commands.

To view session history in the bash shell:

1. Use the shell for a little while, changing directories, redirecting output, or doing other tasks.

   Take your time. We'll wait.

2. Press the ↑ key one time.

   Note that the last (previous) command you used appears on the command line, as shown in Code Listing 3.9. To reissue the command, just press Enter.

3. Continue to press ↑ or ↓ to scroll back or forward through your history. When you reach a command you want to use, press Enter.

   If you see a command that's close, but not exactly what you want to use, you can edit it. Just use the ← and → keys to move across the line, insert text by typing it, and use Backspace or Delete to delete text. When you've fixed the command, press Enter (you don't have to be at the end of the line to do so).

4. `history`

   Type `history` at the shell prompt to see a numbered list of previous commands you've entered.

```
[ejr@hobbes clean]$ ls
background.htm  info.htm    logo.gif
[ejr@hobbes clean]$ ls
background.htm  info.htm    logo.gif
[ejr@hobbes clean]$ history
     1   free
     2   id deb
     3   id ejr
     4   uname -a
     5   ls
...
    40   cd
    41   cp .bash_history oldhistory
    42   vi .bash_history
    43   elm
    44   ls -la
    45   ls -la .e*
    46   elm
    47   lynx
    48   history
    49   vi .bash*his*
    50   history
    51   cd clean
    52   ls
    53   ls
    54   history
[ejr@hobbes clean]$ !40
cd
[ejr@hobbes ejr]$
```

Code Listing 3.9 In this example, we typed the first command, then pressed the ↑ key to reuse the previous `ls` command. `!40` recycled the 40th command from the listing.

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

✔ Tips

- Commands from the current session are kept in memory to scroll through, while commands from previous sessions are kept in the `~/.bash_history` file. You can edit `.bash_history` with any editor to delete unneeded commands or simply delete the file to get rid of the whole history file, which will then be re-created with the next command you issue. (A history of commands is a great jumping-off point to write a script to do the commands automatically. Chapter 10 gives you the specifics.)
- When you're viewing the history, you can recycle commands by typing an exclamation point (`!`) and the line number of the command you want to run again. You'd type `!40`, for example, to rerun command 40.
- Use `history` followed by a number to specify the number of items to list. For example, `history 10` shows the last 10 commands.

Using Completion in the zsh Shell

The zsh shell also offers completion but with added twists over the bash shell for the power user. Basically, though, you can type just part of a command, press Tab, and have the Z-shell complete the command for you (**Code Listing 3.10**).

To use completion in the zsh shell:

1. `ls`

   Use `ls` to list the files in your current directory.

2. `cd pub` Tab

   Type a partial command, as shown here, and then press Tab to complete the command. In this example, we typed the `cd` command and part of the `public_html` directory (truncated to `pub` in the example), and then pressed the Tab key to complete it (see Code Listing 3.10).

✔ Tips

- In the Z-shell, command completion works even if multiple files might match the partial command that you type. For example, if you type `cd pu` (for `public_html`) and there's another subdirectory called `puppy`, then press Tab to complete the name, the shell will show you the options (`public_html` and `puppy`), and then cycle through the options as you continue hitting Tab.
- You can use command completion to complete commands, directory names within commands, and nearly anything else you might enter.
- The Z-shell is smart enough to show you only the subdirectories you could change to. `bash`, on the other hand, would show you files and directories, and beep at you—not as helpful, for sure.

```
$ ls
Complete  NewProject  bogus2
→ ftp puppy
Completed News    dead.letter
→ mail    temp
Mail  access  files
→ public_htmltestme
$ cd public_html
$
```

Code Listing 3.10 In this example, we typed only the `ls` command followed by `cd pub` and pressed the Tab key; zsh completed the command for us.

```
[ejr@hobbes clean]$ ls
background.htm  info.htm     logo.gif
[ejr@hobbes clean]$ ls
background.htm  info.htm     logo.gif
[ejr@hobbes clean]$ history
    1  free
    2  id deb
    3  id ejr
    4  uname -a
    5  ls
...
   40  cd
   41  cp .bash_history oldhistory
   42  vi .bash_history
   43  elm
   44  ls -la
   45  ls -la .e*
   46  elm
   47  lynx
   48  history
   49  vi .bash*his*
   50  history
   51  cd clean
   52  ls
   53  ls
   54  history
[ejr@hobbes clean]$ !40
cd
[ejr@hobbes ejr]$
```

Code Listing 3.11 In this example, we typed the first command, and then pressed the ↑ key to reuse the previous command. `!40` recycled the 40th command from the listing.

Viewing Session History in the zsh Shell

The Z-shell also lets you easily reuse commands from your session history, which is the list of commands you've used during a session or in previous sessions (**Code Listing 3.11**). The history functions are handy for reviewing your Unix session, reusing previous commands (instead of retyping), and modifying (rather than completely redoing) long or complex commands.

To view session history in the zsh shell:

1. Use `zsh` as you usually would, changing directories, redirecting output, or doing other tasks. For example, review the previous chapter and practice the commands you've learned so far.

2. Press ↑ one time.

   Note that the last (previous) command you used appears on the command line, as shown in Code Listing 3.11. To reissue the command, just press Enter.

3. Continue to press ↑ or ↓ to scroll back or forward through your history. When you reach a command you want to use, press Enter.

   If you see a command that's close but not exactly what you want to use, you can edit it. Just use the ← and → keys to move across the line. Then, insert text by typing it or using Backspace or Delete to delete text. When you've modified the command, press Enter (you don't have to be at the end of the line to do so).

4. Type `history` at the shell prompt to see a numbered list of previous commands you've entered.

continues on next page

✔ Tips

- If you have just a minor change to a command, you can edit it quickly and easily. For example, if you just used `ls /home/_users/e/eric` and wanted to issue `cd /home/_users/e/eric` next, you could just type `^ls^cd` to tell the system to replace `ls` from the previous command with `cd` and then reissue the command.
- You can use Ctrl A and Ctrl E while editing a command line to move to the beginning and end of the line, respectively.
- Commands from the current session are kept in memory to scroll through, while commands from previous sessions are kept in the `~/.zsh_history` file. You can edit `.zsh_history` with any editor to delete unneeded commands or simply delete the file to get rid of the whole history file, which will then be re-created with the next command you issue.
- Reviewing session history is a great way to identify your work patterns and needs. If you find yourself repeatedly using the same series of commands, consider writing a script to do the commands automatically, as Chapter 10 describes.
- Most of the command completion options from `bash` also work in `zsh`. Give them a try!

Changing Your Identity with su

Occasionally, you may need to log in with a userid other than your own or need to relog in with your own userid. For example, you might want to check configuration settings that you've changed before logging out to make sure that they work. Or, if you change your shell, you might want to check it before you log out (and you should do that, by the way).

You can use the su (substitute user) command to either log in as another user (**Code Listing 3.12**) or to start a new login shell.

```
[ejr@hobbes asr]$ ls
Projects  testing
[ejr@hobbes asr]$ su asr
Password:
[asr@hobbes asr]$ ls
Projects  testing
[asr@hobbes asr]$ su - ejr
Password:
[ejr@hobbes ejr]$ ls
Mail                editme                  script2.sed
Projects            fortunes.copy           scriptextra.sed
Xrootenv.0          fortunes1.txt           sedtest
above.htm           fortunes2.txt           sorted.address.temp
address.book        groups                  temp.htm
address.temp        history.txt             tempsort
axhome              html.htm                test
bogus               html.html               test2
chmod.txt           mail                    testing.gif
clean               manipulate              testing.wp
compression         nsmail                  typescript
[ejr@hobbes ejr]$ exit
[asr@hobbes asr]$ exit
[ejr@hobbes ejr]$ exit
```

Code Listing 3.12 Changing back and forth from one user to another (and exiting from multiple shells) can get a little confusing, but the prompt often tells you who you are and what directory you're in.

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

To log in as a different user with su:

- `su asr`

  At the shell prompt, type `su` plus the userid of the user you're logging in as. You'll be prompted for a password just as though you were logging in to the system for the first time (Code Listing 3.12).

  If you do not specify a username, the system will assume you mean the `root` user.

  If you're logged in as `root` to begin with, you won't be prompted to give a password.

  You will now be logged in as the new user and be able to work just as if you were that user, though you'll be in the same directory with the same settings that you had before you issued the `su` command.

To start a new login shell with su:

- `su - yourid`

  At the shell prompt, type `su - yourid` (of course, use your own userid or that of the user you want to change to). The addition of the hyphen (`-`) will force a new login shell and set all of the environment variables and defaults according to the settings for the user.

To return to the previous shell:

- `exit`

  Type `exit` at the shell prompt to leave the current shell and return to the previous one. If you use `exit` from the original login shell, you'll log completely out of the Unix system.

✔ Tips

- If you have root access and you `ssh` to the system to administer it, you should use `su` to provide a little extra security. Rather than log in directly as `root` and leave the remote possibility of having your password stolen (or sniffed) off your local system, log in as yourself, then use `su` (with no other information) to change to `root`.
- If you `su` to another user with `su user` (no hyphen) and the new user doesn't have read and execute permissions for the current directory, you will see shell error messages. You can disregard these. See Chapter 5 for more about read and execute permissions.

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

```
xmission> ls ^?^?^?^?
: No such file or directory
xmission> stty erase '^?'
xmission> ls
```

Code Listing 3.13 You can often straighten out a confused `telnet` program or Unix system by using an `stty` command. This one fixes the errant Backspace key.

```
xmission> jf^H^H
jf^H^H: Command not found
xmission> ls ^H^H
: No such file or directory
xmission> stty erase '^H'
xmission>
```

Code Listing 3.14 The `stty` command here fixes the Delete key to work like Backspace.

✔ Tips

- If `stty sane` doesn't fix a messed-up display, try `reset` or even logging out and logging back in or restarting your terminal program.
- You can fix Backspace oddities permanently by adding the appropriate `stty` command to your configuration files or by making changes in your terminal client. See Chapter 8 for details about your configuration files. Refer to Chapter 1 for more helpful details about terminal programs like `ssh` and `telnet`.

Fixing Terminal Settings with `stty`

Another handy thing you can do with your shell is use it to fix those annoying problems that occur with terminal programs. Back in Chapter 1, we mentioned that you might encounter oddities such as your Backspace and Delete keys not working properly. You can fix these problems using `stty` (see **Code Listing 3.13**).

To fix Backspace and Delete key oddities with `stty`:

- `stty erase '^?'`

  If you're accustomed to using Backspace to erase characters to the left of the cursor and you just get a bunch of `^H` symbols on the screen when you try it, you need to educate the terminal about your preferences. Type `stty erase` and press Backspace to fix it (Code Listing 3.13).

  In some cases, depending on your terminal program, you might need to set `stty erase '^H'` and then press Ctrl H to backspace. To enter this command, type `stty erase` and press Ctrl V, then Ctrl H (**Code Listing 3.14**).

To fix general terminal weirdness with `stty`:

- `stty sane`

  Typing `stty sane` at the shell prompt will fix a lot of oddities. For example, if you accidentally issue a bad command and all of a sudden nothing shows up on the screen or if you have general gibberish showing up on the screen, `stty sane` may return your terminal session to sanity.

  The `reset` command is also often effective at fixing a messed-up terminal.

Exiting the Shell

When you're finished with your Unix session, you need to exit the Unix shell. If you've been playing with the `su` and shell commands, you might actually have shells within shells and need to exit from all of them. All you have to do is type `exit` once for each shell.

To exit from the shell:

- `exit`

  At the shell prompt, type `exit`. Ta-da!

✔ Tips

- If you're located at the login shell prompt, you could also type `logout` rather than `exit`. At all other shells, though, you need to type `exit`. In some cases, you could also press Ctrl D, but that depends on your local system configuration.
- Be sure to log off rather than simply close your window or break your connection. It's possible, if the settings at your Unix host are seriously incorrect, that your session could remain open and someone else could pick up right where you left off with your session under your userid.

Chapter 3. Working with Your Shell. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893