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

4 Creating and Editing Files

Creating and editing files are likely the most common tasks you'll perform in Unix. If you're programming, developing Web pages, sending email (uh-huh, really), or just writing a letter, you'll spend a lot of time in an editor.

In this chapter, we'll introduce you to three of the most common editors: `pico` (and `nano` comes along for free), `vi`, and `emacs`. We'll launch this chapter with a general overview of each, and then discuss some how-tos of using each one. With the information presented here, you'll be able to choose an editor based on your needs and get started using it (or using all of them).

Chapter Contents

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

Choosing an Editor

Basically, all editors are designed to do the same things: enable you to create, modify, and save text files. These files could include configuration files, email messages, or shell scripts—essentially any text file you can create. Exactly which editor you choose is up to you, depending on your specific needs and how much you're willing to learn.

In this book, we'll stick to three biggies—`pico`, `vi`, and `emacs`—which will likely give you all the capabilities you'll need. We chose these because `pico` is (arguably) the easiest Unix editor to use, `vi` is one of the most powerful and is available on almost every Unix system, and `emacs` provides an unbelievable number of options and is a handy tool for the up-and-coming Unix pro to have.

Figure 4.1 `pico` offers onscreen command reminders to make it easier to use.

About `pico`

`pico` is one of the more straightforward Unix editors and has become quite popular because it's extremely easy to use. In particular, as shown in **Figure 4.1**, it's menu-driven and intuitive. All of the commands are visible, and you can open, modify, and close files with little effort. `pico` is a great choice if you're just getting started with Unix or if you won't be needing an editor able to leap tall files in a single bound.

For a variety of reasons, mostly connected to open source licensing issues, a clone of `pico`, called `nano`, has been developed and is included in a number of Linux/Unix distributions as well as on systems that you might be using. The `nano` editor is command-for-command the same as `pico`, but it does offer some supplemental higher-end (yet still easy-to-use) features.

Editors Abound

By the way, dozens of other editors exist, such as

- `ed`, `ex`, and `red`, which are simple (in functionality, but not necessarily usage) line-by-line editors
- `joe` and `jed`, which are fairly simple editors and comparable to `pico` in many ways

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

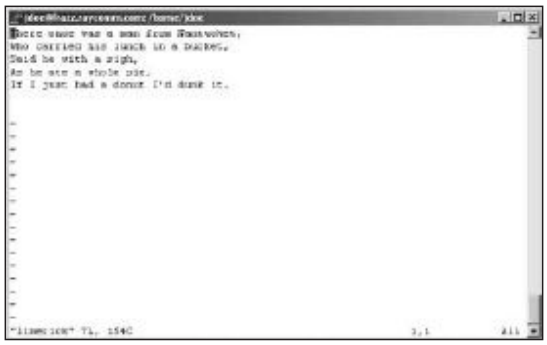

Figure 4.2 vi gives you a clean screen and makes you remember all of its cryptic commands.

For the purposes of this book, we're going to treat pico and nano as equivalent—if you have nano, just mentally write that in wherever you see pico.

pico is distributed with the pine email program, so if you have pine available to you, you likely also have pico. (See Chapter 1 for a reminder on how to find out if pine and pico are available to you.) If pico is not available to you, and if you cannot find nano either, ask your system administrator to install one or the other.

About vi

Although vi is likely responsible for much of Unix's reputation for being complicated and confusing, it offers enormous power and flexibility. Plus, vi is universally available (unlike pico), so for these two reasons, you should consider taking the time to learn it. You might find vi cryptic, counterintuitive, and nitpicky, and for this reason, you might want to choose a different editor if you won't require vi's capabilities. As **Figure 4.2** shows, if you use vi, you won't have menus at your disposal—you'll have to get used to using commands like Esc :q or Esc :%s/vi is arcane/vi is powerful/.

Yes, continuing the theme from a couple of paragraphs ago, there is an equivalent of vi, called vim, that's licensed differently and that's somewhat more powerful. For basic use—everything in this book and far more—the two are identical. In this case, though, you will always find vi, even if it's really vim (vi may actually be a symlink, or shortcut, to vim). If you find vim, though, it will assuredly be vim. All commands will be the same, so just dive in and enjoy.

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

About emacs

With emacs, you start to understand how incredibly customizable Unix can be. It can be "just" an editor—although a very powerful one with all kinds of helpful features—or it can be an email program, file manager, or darn near anything else. We're going to stick to just the editorial functions, but if you find that you like emacs, don't hesitate to explore the Web for other options and features of this editor. **Figure 4.3** shows you what to expect from emacs, including the handy (and fairly familiar) menus.

Figure 4.3 emacs provides both menus and power, all at once.

✔ Tips

- You aren't bound to one editor or another. You can use any editor at any time. We often use pico for email or plain writing because we can type without thinking. We switch to vi when we really need power or just want to make a quick edit without pico's menus, which often seem cumbersome to us.
- You can specify a default editor that will launch automatically in programs that start up an editor for you. Chapter 8 provides details about setting your editor environment variable.
- See Chapter 8 for more information about configuration files, Chapter 10 for more about shell scripts, and Chapter 11 for more about email.
- If you type pico and get an error message telling you that the command is not found, use find, whereis, or ls to search through the likely directories (/usr/bin or /usr/local/bin) to see whether the program is available but not located where your shell can find it. See Chapter 1 for a quick review.
- After you establish a file and start adding content, save your changes using the instructions in the next section.
- You can get helpful information about pico's features by accessing pico help. See the section called "Getting Help in pico," later in this chapter.

Figure 4.4 pico offers an intuitive interface for editing text.

Starting pico and Dabbling with It

You can start and dabble with pico using the following steps. Notice that the pico interface is intuitive and easy to navigate in, as shown in **Figure 4.4**.

To start pico and dabble with it:

1. pico

   To begin, type pico at the shell prompt. The program starts up and you'll see something like Figure 4.4, with the text area up at the top of the window and the command hints down at the bottom.

   If you know the name of the file you want to edit, type pico at the shell prompt followed by the path and name of the file you want to edit (hairyspiders, for example).

2. hairyspiders

   Go ahead. Type something—anything—just to try it out.

   - Use Del and Backspace to help edit text.
   - Use the arrow keys to move up, down, right, or left.

✔ Tips

- Start pico with the -w option (e.g., pico -w filename) to disable word wrapping. You'll find this particularly useful when editing configuration files, as covered in Chapter 8.
- Throughout pico, you'll see ^C, ^J, and dozens of other ^something characters hanging out in the menu at the bottom. The ^ stands for Ctrl, so ^C is Ctrl C, ^J is Ctrl J, and so on.

Saving in pico

You'll generally save your files frequently whenever you're editing them—and you should. Remember, Murphy is watching you!

To save in pico:

- Ctrl O
  Use Ctrl O periodically to save (write "out") the text you're editing.
- hairyspiders
  Specify the filename for your file (**Figure 4.5**).

✔ Tips

- After you save a file for the first time and want to save new changes, just press Ctrl O and then press Enter to confirm the current filename and save it.
- When you exit pico, you'll get a last chance to save your changes. See "Exiting pico" in this chapter for the specifics.
- If you try to save a new file over an existing one—which would obliterate the original—pico carefully asks you if you want to overwrite the file. Answer Yes, and you'll no longer have the original; No, and you'll get to choose a new filename.

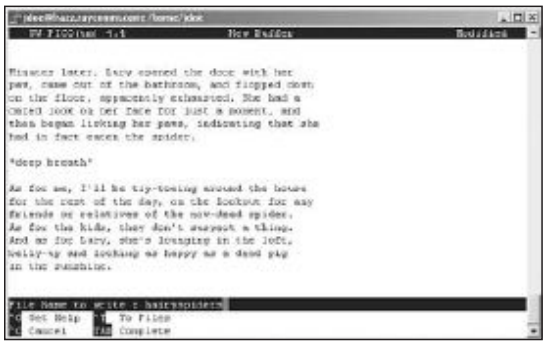

Figure 4.5 In pico lingo, "writing out" just means "saving."

Figure 4.6 Marking, cutting, and pasting text in `pico` can be very handy.

✔ Tips

- You can select and cut blocks of text without also pasting them back into a file. Just skip steps 6 and 7.
- You can paste text blocks as many times as you want. After you select and cut text, just press Ctrl U at each place where you want to insert the cut text.
- If you don't select text, Ctrl K just cuts a single line.

Cutting and Pasting Text Blocks in `pico`

As you're typing along in `pico`, you'll probably need to cut and paste blocks of text, as shown in **Figure 4.6**.

To cut and paste text in `pico`:

1. `pico hairyspiders`

   At the shell prompt, type `pico` followed by the name of the file to edit.

2. Move the cursor to the first line of the text you want to cut.

3. Ctrl ^

   Press Ctrl ^ to mark the beginning of the text you want to cut. (Note that Ctrl ^ is really Ctrl Shift 6—it might work without Shift, but it might not, depending on your terminal program. Try it out and see what happens.)

4. Use the arrow keys to move the cursor to the end of the text you want to cut.

   Note that the text gets highlighted as you select it (Figure 4.6).

5. Ctrl K

   This "kuts" the text.

6. Using the arrow keys, move the cursor to where you want to insert the cut text.

7. Ctrl U

   Use this key combination to paste the cut text into the file at the new location.

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

Checking Spelling in pico

Another handy thing you can do in pico is chek yoor speling, as shown in **Figures 4.7** and **4.8**.

To spell-check in pico:

1. `pico hairyspiders`

   At the shell prompt, type pico and the filename of the file to edit.

2. Ctrl T

   Pressing these keys starts spell-checking the file. pico will stop at each misspelled word (Figure 4.7).

3. `correctspelling`

   Type in the correct spelling for any words flagged as misspelled, or press Enter to accept the current spelling and move along to the next word.

✔ Tips

- You can press Ctrl C to cancel spell-checking at any time.
- Because the spell-checker in pico isn't full-featured, consider using an alternate spell-check program by specifying it on the command line, like `pico -s ispell hairyspiders`, so you can get a little more assistance. See Chapter 15 for more information.
- When the entire document has been spell-checked, pico will tell you that it's done checking spelling, and you can continue editing the file (Figure 4.8).

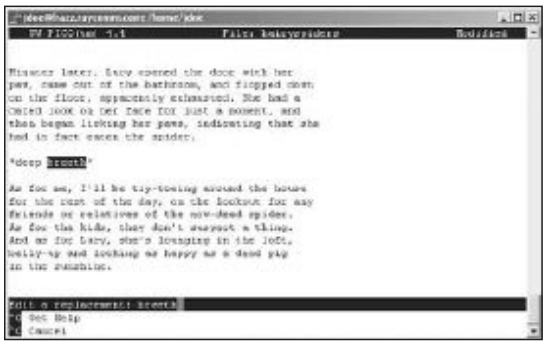

Figure 4.7 pico prompts you to correct the spelling of misspelled words.

Figure 4.8 pico informs you when the procedure is complete.

Figure 4.9 pico gives you all the information you need.

Getting Help in pico

A great way to find out more about pico is to access pico help. In addition to finding answers to your questions, you can find out about pico features and capabilities of which you may not be aware (**Figure 4.9**).

To get help in pico:

1. Ctrl G
   In pico, press Ctrl G to access help.
2. Move through the help pages:
   - Ctrl V moves you down through the help page.
   - Ctrl Y moves you up through the help page.
3. Ctrl X
   Use this combination to exit help.

To get help with pico startup options:

- man pico
  At the shell prompt, type man pico to learn more about startup options, including a variety of options that control how pico works.

✔ Tips

- Keep your eyes on the pico status line for current information, error messages, and occasional hints about using pico. The status line is the third line from the bottom of the screen, just above the menu, as shown in Figure 4.9.
- Keep in mind that pico really is a very basic program. If you're looking for a command or function that isn't readily available, it's probably not there. You might check out vi or emacs instead. And keep in mind that nano is like pico but does have some supplemental features (and you don't have to learn another editor). It too may be worth a try.

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

Exiting pico

When you're done editing in pico, you'll exit it using the following steps.

To exit pico:

1. Ctrl X
   Within pico, press Ctrl X. If you haven't made any changes to the text since you last saved the file, you'll find yourself immediately back at the shell prompt. If you have made changes, you'll be prompted to "Save modified buffer" (**Figure 4.10**).
2. At the "Save modified buffer" prompt:
   - Press Y if you want to save your changes. Proceed to step 3.
   - Press N if you don't want to save your changes. You'll end up back at the shell prompt.
3. `bighairyspiders`
   Specify the filename for your file if it's the first time you've saved it. If you've saved it before, press Enter to confirm the current filename or change the name to save a copy and not change the original file.

Figure 4.10 pico gives you the opportunity to "Save modified buffer." Without the techno-babble, this means to save the text you just wrote or edited before you exit.

✔ Tip

- A *buffer* is what the computer uses to temporarily store information, and if it's modified, that means that it's temporarily storing something that you haven't saved to disk.

Figure 4.11 The vi editor inundates you with tons of onscreen help and advice, as shown here. Well, documentation is available, but the vi interface itself isn't really helpful at all!

Starting vi and Dabbling with It

Before you go running off to use vi, understand that it has two modes (both of which look pretty much like **Figure 4.11**):

- *Insert mode* (sometimes called input mode), in which the keys you press actually show up in the file that you're editing. You use this mode to add or change text.
- *Normal mode* (sometimes called command mode), in which every keystroke is interpreted as a command. You use this mode to do everything except enter text.

What's confusing for many people about vi is that it starts you in command mode, meaning that if you just start typing, you may see some blank spaces, characters, and bits of words that you type—essentially, a bunch of garbage that does not exactly represent what you're typing—and you'll hear a lot of beeping. So, as we'll show you in the following steps, you'll need to access the input mode as soon as you start vi.

To start vi:

1. vi

   At the shell prompt, type vi. The program starts up and you'll see something like Figure 4.11. The ~ symbols show blank lines below the end of the file.

2. i

   Type i to get into input mode. This itself is a command issued in command mode, so it won't show up on the screen.

continues on next page

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

3. `hairy spiders lurk`

   In input mode, type anything you want. Everything you type will show up on the screen until you return to command mode by pressing Esc. When you are in command mode, you can use the arrow keys to navigate up and down in the file line by line and use Ctrl F and Ctrl B to scroll one screen forward and backward, respectively.

✔ Tips

- To get help for `vi`, type `man vi`. See Chapter 1 for more about `man` pages.
- If you're not sure what mode you're in, press Esc to go into command mode. If you're already in command mode, you'll hear a beep. If you're in input mode, you'll change to command mode.
- Many Unix-like systems, including Linux and Mac OS, actually provide a program called `vim` in the place of `vi`. `vim` (VI iMproved) is like `vi` but feature-rich and more flexible, and you can still start it with the command `vi`.
- You can open specific files or even multiple files when you access `vi`. At the shell prompt, type `vi filetoedit` (or whatever) to open a specific file. Or, for example, type `vi *.html` to open all of the HTML documents in a directory, then use Esc `:n` (for "next") and then press Enter to move to each subsequent file.
- See "Adding and Deleting Text in `vi`" later in this chapter for more details about editing in `vi`.

Figure 4.12 Save early, save often. That's the safe rule for `vi`.

Saving in `vi`

You'll want to save changes to your documents frequently, especially as you're learning to use `vi` (**Figure 4.12**). Until you're accustomed to switching between command and input mode, you may accidentally type in commands when you think you're typing text, with unpredictable results. To save files, just follow these steps.

To save text in `vi`:

- Esc `:w limerick`

  Press Esc to get out of input mode and into command mode, then type `:w` (for "write," as in write to the disk) followed by a space and then the filename (`limerick`, in this example) you want to use for the file, then press Enter. If you've already saved the file once, just press Esc and type `:w`, then press Enter.

✔ Tips

- If you've already saved your file at least once, you can save changes and exit `vi` in one fell swoop. In command mode, type `:wq` (for "write quit"). For more information about quitting `vi`, see the section "Exiting `vi`," later in this chapter.
- If you want to save a file over an existing file (obliterating the original as you do), use `:w! existingfilename` in command mode. The `!` forces `vi` to overwrite the original.

Adding and Deleting Text in vi

Adding and deleting text in vi is a bit more complicated than doing the same in pico. Whereas in pico, you basically just place your cursor where you want to make changes, vi has a whole slew of commands that you use to specify where the changes should occur. (**Tables 4.1**, **4.2**, and **4.3** list only a very few of your options.) Plus, to issue the commands, you have to switch to command mode.

To add or delete text in vi:

1. vi

   To begin, type vi at the shell prompt.

2. i

   Change into input mode.

3. There once was a man from Nantucket

   Type some text that you'll want to add to.

4. Esc

   Press Esc to enter command mode before you issue the commands.

5. Choose a command, based on what you want to do to the text.

   Table 4.1 lists commands to add text.

   Table 4.2 lists commands to delete text.

   Table 4.3 lists miscellaneous editing commands.

6. dd

   Type the command. Here, we're deleting the current line of text.

Table 4.1

vi Commands to Add Text

Command	Function
a	Adds text after the cursor
A	Adds text at the end of the current line
i	Inserts text before the cursor
I	Inserts text at the beginning of the current line
o	Inserts a blank line after the current line
O	Inserts a blank line before the current line

Table 4.2

vi Commands to Delete Text

Command	Function
x	Deletes one character (under the cursor)
X	Deletes one character (behind the cursor)
dd	Deletes the current line
5dd	Deletes five lines starting with the current line (any number would work here)
dw	Deletes the current word
cw	Changes the current word (deletes it and enters input mode)
r	Replaces the character under the cursor with the next character you type
R	Replaces the existing text with the text you type (like overtype mode in most word processors)

Table 4.3

Other Handy vi Editing Commands

Command	Function
yy	Copies the current line
p	Pastes any copied text after the cursor or line
J	Joins the current and following lines
u	Undoes the last change
U	Undoes all changes on the current line
.	Repeats the last command

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

Figure 4.13 Reading an additional file into the current one can make your editing tasks much easier.

Importing Files into `vi`

You can also merge multiple files in `vi` by reading additional files into the current one, as shown in **Figure 4.13**. Basically, all this means is that you insert one file into the file you're currently editing.

To import files in `vi`:

1. `vi hairyspider`

   At the shell prompt, type `vi` followed by the filename to start `vi` with, in this case, the `hairyspider` file.

2. Esc`:r filename`

   At the point in the file where you want to import text, press Esc, then type `:r` and the filename you want to read into the file.

✔ Tip

- `vi` also lets you read the output of commands into the file. For example, if you want to read the list of files in a specific directory into the file, use Esc`:r !ls` in command mode.

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893
Prepared for vijay.sharma@fmr.com, Vijay Sharma

Searching and Replacing in vi

One of vi's better features (and advantages over pico) is that it allows you to search and replace throughout entire files. As shown in the next sections, you can just find a specific string of text (a *regular expression*, in Unix lingo; see **Figure 4.14**), or you can find the text and replace it with other text, as in **Figure 4.15**.

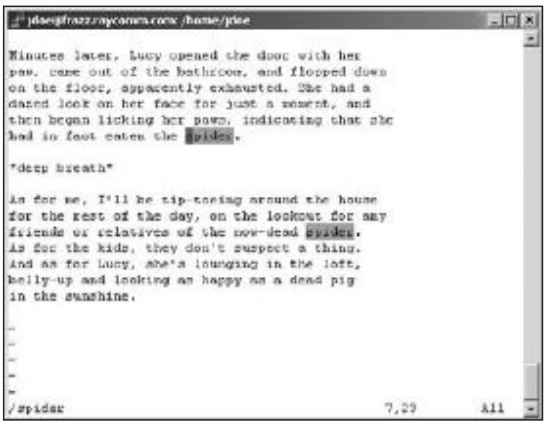

Figure 4.14 Searching for text in vi is quick and reliable.

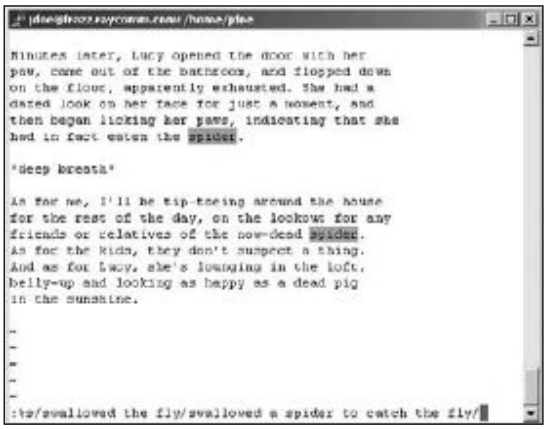

Figure 4.15 Replacing text in vi requires a bit of arcane syntax, but you get used to it quickly.

To find a string of text in vi:

1. `vi hairyspider`

   For starters, access vi and a specific file.

2. Esc`/spider`

   Enter command mode, then type / followed by the text you're looking for. Here, we're looking for "spider," but you may be looking for "the fly" or "wiggled and jiggled and tickled inside her." Or whatever.

3. Enter

   Press Enter to find the first occurrence of the term. Type n to find the next one.

To search and replace in vi:

1. `vi hairyspider`

   For starters, access vi and a specific file.

2. Esc`:%s/swallowed the fly/swallowed a spider to catch the fly/`

   Enter Esc`:%s/` plus the text to find, another /, followed by the replacement text, as in Figure 4.15. Here, we replace "swallowed a fly" with "swallowed a spider to catch the fly," but perhaps you might forego the spider and simply go for some antacid.

✔ Tips

- A great use for the search-and-replace feature is if you end up with DOS text files in your Unix account (by uploading a text file from a Windows machine as a binary file, most likely). If you view DOS files through a Unix shell, all the lines in the file will end with `^M`. But if you try to type `^M` when you're doing a search and replace, the `^M` won't show up. What to do? Press Ctrl V, then Ctrl M. Just search and replace with `:%s/`Ctrl V Ctrl M`//g`. The Ctrl V command "escapes" the following character, so you can press it without actually doing what the command would otherwise do. If you don't escape the Ctrl M, `vi` thinks you just pressed Enter and tries to execute the unfinished command.
- See the section on `grep` in Chapter 6 for information about searching with regular expressions.
- Add a `g` at the end of the command to make it apply to all occurrences in the file. Otherwise, it applies only to the first occurrence on each line.

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

Exiting vi

Whew! Time to exit vi (**Figure 4.16**).

To exit vi:

- Esc :q
  Enter command mode by typing Esc, then type :q to quit vi. If you haven't saved your latest changes, vi will not quit and will tell you to use ! to override. To quit without saving your changes, use :q!, as shown in Figure 4.16.

✔ Tips

- If you don't really want to quit but want to edit a different file instead, type :e filename to open a new file to edit.
- We recommend that you take a few minutes to try out some of the commands that you'll use throughout your vi experience. If you don't think you'll need this range of commands, consider using pico or nano rather than vi.
- It takes some practice to get accustomed to vi, but the time spent is well worth it. With patience and practice, you'll quickly become proficient in using vi. Take your time, take deep breaths, and plow ahead.

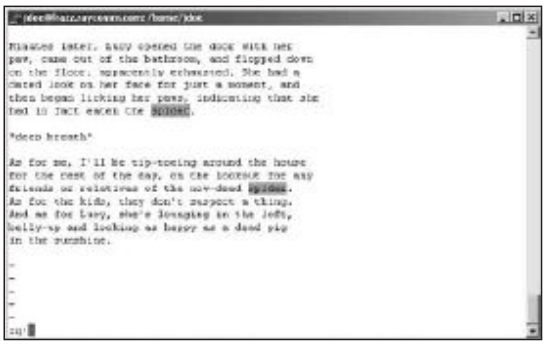

Figure 4.16 Use Esc :q! to quit vi without saving changes.

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

Figure 4.17 emacs starts out with some basic information, but you can just start typing if you want.

Figure 4.18 emacs might helpfully start out in a spiffier interface if you're sitting at the keyboard of a Linux system. You can get the plain variety, though.

Starting emacs and Dabbling with It

For the novice, emacs offers a reasonable middle ground between the user-friendliness of pico and the power of vi (or vim). It's not available on all systems, though, so you'll just have to type in the command to see if you have access to it. (Refer back to Chapter 1 if you don't.)

Using emacs, you can just type, as you'd expect, then use command sequences, which are basically Ctrl keys, to make emacs do useful things like save, quit, and the like. When you start emacs, it'll probably look very much like **Figure 4.17**. Some systems "helpfully" open a new window and give you the graphical version; you'll see something like **Figure 4.18**.

To start emacs:

1. `emacs`

   At the shell prompt, type emacs. The program starts up and you'll see something like Figure 4.17. The helpful information may or may not be present, but you can ignore it for now at any rate.

2. `This morning I got up, went downstairs, and found a humongous spider in the bathroom. After I quietly composed myself, I looked around the house for something to put him in...the kids' bug catcher thing (nowhere to be found)...a jar... tupperware...a lidded cup...the salad spinner (BwaaaaHaaaHaaa!)....`

   Type anything you want.

continues on next page

You can use the arrow keys to navigate up and down in the file line by line. See **Table 4.4** for a brief summary of the most useful commands in `emacs`.

✔ Tips

- To get help in `emacs`, type `man emacs`. See Chapter 1 for more about `man` pages.
- If `emacs` helped you out by starting in the graphical mode, but you want to play along with us in the text mode, use `emacs -nw` to start the program. (The `-nw` flag means "no windows.")
- `emacs` uses both Ctrl keys and the "meta" key to issue commands. PC users should use the Windows key (if available) or Alt in place of the meta key (but you should remember that you'll see *M+* or *Meta+* in most `emacs` documentation). For those of you using keyboards that actually have a key labeled "Meta," by all means, you should use it when you see Alt. Mac users should use Option.
- As useful as `emacs` is, it does have a few quirks. For example, if you want to access help, you press the Backspace key, which issues the Ctrl H command. To fix this idiosyncrasy, press Alt X and then type `normal-erase-is-backspace`.

Table 4.4

Handy `emacs` Commands

Command	Function
Ctrl X, Ctrl F	Opens a new file (existing or new)
Ctrl /	Undoes the last change
Ctrl G	Cancels the current operation
Esc	Bails out of menu selections (and other things)
Ctrl V	Moves down one page (screen)
Alt V	Moves up one page (screen)
Alt <	Moves to the beginning of the file
Alt >	Moves to the end of the file

Figure 4.19 Navigating emacs menus isn't exactly intuitive, but it's straightforward after you get started.

Using emacs Menus to Spell-Check

Spell-checking is good. Learning to use emacs menus is good. And in emacs, learning to spell-check also allows you to familiarize yourself with emacs menus. Use F10, then a key letter of each menu, menu item, and submenu as needed to navigate through the menus. (You'll see hints and prompts at the bottom of the screen, as shown in **Figure 4.19**.) Follow along to use the menus to spell-check your file.

To use emacs menus to spell-check:

1. `emacs hairyspiders`

   For starters, fire up emacs and a specific file.

2. Press F10 to access the menus.

3. `t`

   Next, type the first letter of the menu you want—this example uses `t` for Tools for now.

4. `0`

   Try `0` (zero) for spell-checking.

5. Press Enter and enjoy your spell-check.

✔ Tips

- Press Esc as many times as needed to back out of places (like menu selection choices) you do not want to be.
- Reading and following along with the tips onscreen is essential to having a happy life (or a tolerable coexistence) with emacs.

Saving in emacs

Save yourself potential headaches by saving frequently. To save files in `emacs` (**Figure 4.20**), follow these steps.

To save text in `emacs`:

- Ctrl X/Ctrl S `hairyspiders`
  Press Ctrl X to let `emacs` know that another command is coming, and then Ctrl S to save. Finally, type the filename (`hairyspiders`, in this example) you want to use for the file, then type `e`. If you've already saved the file once, just press Ctrl X, followed by Ctrl S.

✔ Tips

- If you look around in your home directory (or whatever directory you're working in) after experimenting with `emacs`, you'll probably notice a slew of files with names ending in ~. Those are `emacs` backup files, created for your convenience and sanity. If you don't need them, just delete them with `rm -i *~`. If you do need them, just use `mv oopsie~ oopsie` and you're back in business.
- If you want to save a file over an existing file, use Ctrl X, Ctrl W, and then enter the existing filename to overwrite the original.

Figure 4.20 Saving is very important—at least if you want to keep the results of your efforts.

Chapter 4. Creating and Editing Files. Unix and Linux: Visual QuickStart Guide, Fourth Edition, ISBN: 9780321657893

Figure 4.21 Use Ctrl X, Ctrl C to quit emacs.

Exiting emacs

Wow! It's already time to exit emacs (**Figure 4.21**).

To exit emacs:

- Ctrl X, Ctrl C
  Press Ctrl X to let emacs know that another command is coming, then Ctrl C to close. If you haven't saved your latest changes, emacs expects you to decide if you want to save or discard unsaved changes, as shown in Figure 4.20.

✔ Tip

- If you end up down at the command line but don't want to save or anything—you just want to return to your file—use Ctrl G to cancel.

This page intentionally left blank