Introduction to the Emacs Editor

Scott D. Anderson*
Wellesley College
Scott.Anderson@acm.org

1 What is Emacs?

Emacs is an *editor*: a program that allows you to modify files. Many of you have already used an editor, so it may be easier to think of it as similar to, but different from, editors such as `vi`, Microsoft WORD, or Corel WordPerfect. The latter two are intended to create typeset documents, so they include commands to manage fonts, margins and other word processing features. For that reason, they are pretty different from Emacs. Emacs is not a word processor; it's similar only in that you use them to modify the contents of files (as opposed to, say, playing Solitare).

Emacs and `vi` are editors of ASCII (plain text) files. In this, they are more similar to the Windows program called "Notepad" or the Macintosh program called "SimpleText." Since `vi` is the other commonly used editor on Unix machines, I will often compare and contrast Emacs to `vi`; if you don't know `vi`, just ignore that.

2 Overview of Emacs

Emacs is an editor available on most UNIX systems; indeed, there are implementations for Windows 95 and the Macintosh, too. Like `vi`, which is built-in to all UNIX systems, Emacs is a screen editor. Unlike `vi`, Emacs is "modeless" editor, which means that you don't have to switch to "insert mode" in order to put characters into the file. [1] By default, any character you type is put into the file at that point. (The "point" is usually marked by a rectangular block on the screen.)

Since normal characters like "a" or "$" are inserted into the file, you have to use abnormal characters to give commands to Emacs. There are essentially four ways to give commands to Emacs:

- Control characters. This is when you hold down the control key, which is usually at the left side of your keyboard, near the shift key, and type a letter. For example, hold down control and type "b" and you've typed a "control-b"[2] In Emacs documentation, a control letter like "control-b" is written as "C-b." That means to hold down the control key and type an "b"—don't type the hyphen. By the way, C-b moves the cursor backward one character.

- Prefix characters. This is where the command is actually a sequence of keystrokes, the first of which is called the prefix. For example, you can save your file by typing "C-x C-s." The "C-x" is the *prefix*. Lots of commands start with "C-x." Another is "C-x u," which is the *undo* command.

- Modified characters. On certain keyboards, there are extra modifier keys. A modifier key is like the shift key: it changes the character when held down as the character is typed. Shift turns "b' into "B." Control turns "b" into "control b." Similarly, the *meta* key turns "b" into "meta b." On some keyboards, the meta key is actually marked, just like the control key. On other keyboards, one of the existing keys is assigned to be the meta key, such as the Alt key. Here at Wellesley, the "Alt" key is the meta key. So, for example, to use the "meta b"

* My thanks to J. Ross Beveridge of Colorado State University, who sent me an Emacs primer that became the core of this document.

[1] On the other hand, Emacs has lots of "modes" for editing different kinds of files. The word "mode" is somewhat overused in computer science.

[2] This is actually a ASCII character. It's number 2 in the ASCII code, while an "B" is number 66 and "b" is number 98. Do "man ascii" to see a table of the whole ASCII character set. All the control codes are in the beginning, so "control-a" is number 1, "control-b" is number 2 and so forth.

command, hold down the Alt key and type a "b." That command would be documented as "M-b" (analogously to "C-b"). The "M-b" command backs the cursor up by one word.

What if your keyboard doesn't have a meta key, not even one that is assigned? (This often happens if you are using Emacs via a modem and a terminal emulator; for example, you're logged into a Linux machine using telnet or SSH.) You can always use the Escape key (marked ESC) as a prefix character to mean *meta*. Note that ESC is not a modifier key, so you first type "ESC" then type "b" but don't hold down the ESC key. Meta keys are not in the 7-bit ASCII character set, but the ESC character is (it's number 27).

By the way, sometimes these modifier keys are combined. If you hold down *both* meta and control and type a "b" you get the "C-M-b" command, which goes backword by one parenthesized expression. If you don't have a meta key, type ESC followed by C-b.

- Extended commands. Emacs has so many commands that there are not enough keystrokes for them all, so you have to specify their names. This is done with the less commonly used commands, so you won't need this much until you become more experienced. First, you have to type a prefix, which is "M-x" and then Emacs prompts you for the name of the command in the *minibuffer*[3] After you type the name of the command,[4] you press the return key, which is documented as "RET," just as if this were the unix command line. An example is "M-x man RET" which prompts you for some input and runs the UNIX "man" command on that input and puts the result into a buffer for you, which then allows you to search it, edit it, save pieces, and so forth.

  Note that you can get *any* command via M-x; it's just that the more common commands are *bound* to keys. When the documentation talks about "key bindings," they're talking about the connection between keystrokes and commands. You can modify those key bindings if you want to.

Finally, when you start Emacs you'll notice some menus and icons, so you can also issue commands by using the mouse. Power users spend most of their time using keyboard commands, so that's what I'll describe in this document.

One other distinction between Emacs and `vi` is that Emacs allows you to edit several files at once. The window for Emacs can be divided into several windows each of which contains a view into a buffer. Furthermore, on a windowing system, you can start up multiple windows (Emacs calls them *frames*) each of which seems to have Emacs running in it, but there's really only one Emacs process behind all of it. Each Emacs buffer typically corresponds to a different file. Many of the commands listed below are for reading files into new buffers and moving between buffers. This distinction results in a very big difference in how the two editors are used. Most `vi` users enter and exit `vi` many times, as they edit files and save them. Consequently, `vi` has been built to start very quickly. Most Emacs users, on the other hand, start up Emacs once when they login and only exit it when they logout. Instead, they just save buffers and switch to another window where they run the compiler or whatever. Consequently, very little effort has gone into make Emacs start quickly.

3 Using Emacs, Getting Started

If you're logged into the console, so you have a window manager available, the easiest way to start Emacs is from the menu. (TODO: Describe where for KDE and Gnome.) It will come up and have a bunch of information about getting help and running the tutorial. The initial buffer will be `*scratch*`. You can then use commands to read in files as necessary.

Alternatively, you can start Emacs from the command line. This is useful when you're logged in remotely via telnet or SSH. To use Emacs on a file, type `emacsfilename`.[5] If the file exists, then the first screen's worth of the file will be displayed; if it doesn't exist, a help message will be displayed. Alternatively, you can start up Emacs from the command line without mentioning a file, in which case it comes up just as described in the previous paragraph.

[3] The minibuffer is a line at the very bottom of the Emacs window, right below the status line, (which is the one that tells you what file you're editing and information like that). The minibuffer is used anytime Emacs needs more information from you in order to execute a command; and it's used whenever Emacs to tell you something short in response to a command.

[4] Emacs has what is called *completion*, meaning that as you type in a command, you can type a space or tab at any time and Emacs will either fill out the rest (if there is only one possible completion) or it will list the possible completions. This tremendously reduces the amount of typing and prevents mistakes.

[5] If you want Emacs to run in its own window, which you might not if you're logged in remotely, but sometime you might (It's too complicated to get into now. TODO: expand this.) you would typically would, follow the command with an ampersand (&) and it will run "in the background" in its own window.

To give you a head start, the following lists some basic commands you will need to know to use Emacs to edit a file. An exclamation point to the left of the commands indicate those to learn immediately.

3.1 Help Commands

!	C-h	help-command, prefix character for lots of useful help commands
!	C-h t	help-with-tutorial, command to run the tutorial
	C-h i	info describes most of the emacs commands in man style pages
	C-h k	describe-key tells you what a particular keystroke does
!	C-h a	command-apropos, prompts for a string and then searches for all emacs commands that contain that string
!	C-h ?	help-for-help, describes how to use the help facilities

3.2 File Reading and Writing Commands

!	C-x C-f	find-file, first prompts for a filename[6] and then loads that file into a editor buffer of the same name
!	C-x C-s	save-buffer, saves the buffer into the associated filename
	C-x C-w	Write named file, prompts for a new filename and writes the buffer into it

3.3 Movement Commands

Each Emacs buffer has a location called "point," which is where new input will go. There are zillions of commands to move the point. In addition, the arrow keys and clicking with the mouse work too. You can change the window (but not the point) by using the scroll bars.

However, I'd like to encourage you to learn some of these movement keyboard commands. First of all, there are circumstances (some telnet connections), when the arrow keys and the mouse won't work. Secondly, the movement commands can be more efficient. Using the arrow keys takes no effort to learn, but you pay for that every time you have to move a long way. Some users don't seem to mind holding down an arrow key for 10 seconds when a few `C-v` commands would have the same effect, but you won't know you have a choice unless you try to learn `C-v` and other such commands.

!	C-a	put cursor at beginning-of-line
!	C-e	put cursor at end-of-line
!	C-b	go backward one char
!	C-f	go forward one char
!	C-n	go to next line
!	C-p	go to previous-line
!	C-v	scroll-up by one screenful
!	M-v	scroll back by one screenful
	M-<	go to beginning-of-buffer
	M->	go to end-of-buffer
	M-b	go backward one word
	M-f	go forward one word

3.4 Copy, Kill and Yank Commands

Word processing programs refer to "copy and paste," but Emacs calls them "copy and yank." (You might argue that Emacs should change, but Emacs has been calling it "yank" for over twenty years and it's a hard habit to break.) Anything that is deleted ("killed") is remembered (in the "kill-buffer") and can be "yanked" back at any location.

[6] Recall that Emacs has *completion*, meaning that as you type in a filename, you can type a space or tab at any time and Emacs will either fill out the rest (if there is only one possible completion) or it will list the possible completions. This tremendously reduces the amount of typing and prevents mistakes.

!	C-d	delete-char
	M-d	delete from cursor to end of word immediately ahead of the cursor
!	C-k	kill-line, delete the rest of the current line
	C-@	set-mark-command, the mark is used to indicate the beginning of an area of text to be killed, copied or whatever
	C-w	kill-region, delete the area of text between the mark and the current cursor position
	M-w	copy-region-as-kill, copy area between mark and cursor into kill-buffer so that it can be yanked into someplace else
!	C-y	yank, insert at the point whatever was most recently killed or copied
!	M-y	yank-pop, insert at the point whatever was killed or copied previous to the last yank or yank-pop

3.5 Search Commands

Emacs uses something called "incremental search," which I think is much better than search commands on other editors. Say you want to search for "administration." Rather than prompt you for the entire search string, Emacs lets you type it one letter at a time. Each letter you type causes Emacs to show you the next occurrence. So, you type `C-s` (incremental search) and the letter "a," and Emacs shows you the next "a" in the file. Then you type "d" and Emacs shows you the next occurrence of "ad." Then you type "m" and emacs shows you the next occurrence of "adm." By the time you've typed "admin," you've probably already found the occurrence of "administration" that you were looking for, and you had to type only a fraction of the search string. If you get to the end of the search string, just keep typing C-s and it shows you the next occurrence.

!	C-s	isearch-forward, incremental search prompts for a string and searches forward in the buffer for it. It starts searching immediately
	C-r	isearch-backward, like isearch-forward, but goes backwards
	M-%	query-replace, prompts for a search string and a string with which to replace the search string

3.6 Window and Buffer Commands

	C-x 0	deletes current window
	C-x 2	splits current window into two parts, so you can edit at two different locations in the same file or so that you can view two different files at the same time
	C-x b	switch-to-buffer, display a different buffer on the screen
	C-x o	move the cursor to the other window (assuming that you have two windows/buffers open at once)
!	C-x C-b	list-buffers, shows the buffers currently loaded into emacs

3.7 Exiting Emacs, Fixing Mistakes and Other Important Stuff

!	C-x C-c	save-buffers-kill-emacs, when you are finished editting, it offers to save editted but unsaved buffers and then exits
!	C-g	keyboard-quit, if while typing a command you make a mistake and want to stop, this aborts commands in progress
	C-u	universal-argument, if you want to do a command several times type this command followed by a number (for the number of times) followed by the command you wish repeated. It's also used to modify the behavior of some commands.
!	C-x u	undoes the last command typed, in case you made a mistake
!	M-x	execute-extended-command, prompts for the name of an Emacs command, allows you to execute commands if you know roughly what it is called but cannot remember the key strokes for it

4 Customizing Emacs

Perhaps the biggest difference between Emacs and `vi` (and other editors), is that Emacs is designed to be customizable. It has a built-in way to run programs written by users (in a programming language called Emacs-lisp or "elisp" for short) and loaded into Emacs. Many, many people have written extensions to Emacs to do all kinds of things. When you're more experienced, perhaps you'll be one of them.

There are simpler ways to customize Emacs than writing code. Often it's as simple as setting the value of a variable. For example, suppose you would like Emacs to save backup versions of your files. You would set some variables to true by doing the following:

```
(setq make-backup-files t)
(setq version-control t)
```

Where do you put these variable settings? You put them into a file called `~/.emacs`. When Emacs starts up, it reads in and evaluates all the code in that file. There is a sample `.emacs` file; it should be in the same place where you got this file. It is called `dot-emacs` so that it'll be easily visible.

5 Backspace versus Delete

On PCs, the convention is that the Backspace key deletes the character to the left of the insertion point (the previous character), and the Delete key deletes the character to the right of the insertion point (the next character). On UNIX systems and other older operating systems (Multics, TOPS-20, ...), it's typically the case that the Delete key deletes the previous character and the Backspace key does something else, such as backing up but not deleting the character. [7] Consequently, the Backspace key wasn't used very often. Note, by the way, that Backspace and Delete are both ordinary ASCII characters: Backspace is the same as C-h, which is character 8; Delete is character 127.

Early in Emacs history, the implementors decided that since Backspace wasn't used much, and that C-h would be mnemonic for "help," they assigned C-h to be the prefix character for all of the built-in help facilities. Of course, this meant that the Backspace key also was a prefix character for help, since (at the time) the computer couldn't tell the difference between backspace and C-h. Since most people didn't use the Backspace key, this wasn't a problem.

Modern computer keyboards can tell the difference between C-h, Backspace and Delete. Most Linux machines make the Backspace and Delete keys act the same way, deleting the previous character. If you want to delete the next character (PC-style Delete), try to get used to typing "C-d," the Emacs command to delete forward. This will be more efficient than using the arrow key to go forward and then pressing the Delete key. (When you get more advanced, you can re-bind the Delete key to be the same as "C-d.")

6 XEmacs versus Emacs

There have been many implementations of Emacs over the years (the editor is over twenty years old) and so the word "Emacs" often means, generically, any one of those implementations. They are quite similar, although many features have been added over the years.

Currently, there are two major implementations: GNU Emacs, written by the Free Software Foundation (FSF) and XEmacs, written by the XEmacs organization. They are both free, both very good, and very similar. Thousands of organizations all over the world use one or the other, and sometimes both.

7 Flow Control

This section is aimed at people who may use Emacs via a modem and terminal emulator. If you always use Emacs from the computer's console, you can probably skip this section.

Terminals are typically slower than the computers they are communicating with, and so when the computer starts blasting a bunch of characters to the terminal, the terminal needs a way to say "Wait! I need to catch up." When it

[7] This makes it possible to actually underline something, by typing it and then backing up and typing underline characters.

catches up, the terminal says “Okay, go ahead,” and the computer sends some more data. This is called *flow control*, since the terminal needs to be able to control the rate that information flows in.

For many years, flow control was done by the terminal sending “control S” (stop) and “control Q” (resume) to the sender. In fact, you can still do that at most Unix prompts. To try it, just sent a large file or listing to the screen (say by `cat.login` or `ls-lR/`). Then type C-s and the output should freeze; type C-q and it resumes.

Notice that I used Emacs notation for C-s and C-q; I did so because those are legitimate Emacs commands that you might want to use (incremental-search and quoted-insert, respectively). If you use one of those commands (or even type one accidentally) and you get odd behavior, particularly if the screen seems to be frozen, that’s probably because the C-s or C-q is being interpreted by the computer as flow-control, instead of being passed on to Emacs.

If that happens, the easiest solution is to give the command `M-xenable-flow-controlRET` and you can then type C-s as C-\, and C-q as C-^. You can also put `(enable-flow-control)` into your `.emacs` file.

8 Parting Words

Emacs has many, many other useful commands. A partial listing, more compact than this document, is in the `emacs-refcard-letter` files that are in the same place where you got this file. As you get more proficient at it, try listing the key bindings (C-h b) to find other commands. There is also an enormous amount of on-line documentation, either via C-h m (help with this mode), C-h F (the FAQ), or C-h i (the info pages, which are hyperlinked manual pages).