Appendix: The `emacs` Editor

The **emacs** editor originally was developed at the MIT Laboratory for Computer Science. As **emacs** gained popularity, it was ported to UNIX and Linux so that today, many Linux systems offer some version(s) of **emacs** in addition to the standard **vi**/**vim** editors. Some people prefer **emacs** over **vi** for general purpose text editing as well as writing C/C++ programs.

Emacs is a highly customizable terminal-based editor. Thus, any general description of **emacs** is likely to be inadequate since commands you enter for the same operations can vary from version to version. Here is a brief introduction to **emacs**.

The four major features of **emacs** are

1. Rather than operating in distinct input and command modes like **vi**, **emacs** operates in only one mode: Printable characters typed are inserted at the cursor position. Commands are given as control characters or are prefixed by ESC or CTRL+X. In this appendix, we use the notation ^*C* as a shorthand for CTRL+*C* for any character *C*.

2. The **emacs** editor can manage multiple display windows by dividing a CRT screen into two or more horizontal sections. You can cut and paste between windows.

3. The **emacs** editor allows for user definition of editing functions and command keystrokes. Any built-in or user-defined editing function can be *bound* to any keystroke. For example, you can bind the function `delete-character` to the keystroke ^D to invoke that function. A command can also be invoked by its full name.

4. Some versions of **emacs** have built-in features that help in editing C, LISP, Pascal and other languages.

`Emacs` Commands

Each command of **emacs** has a full name and a one or two character *key sequence* used to invoke the command. A key sequence is either a single control character or a two-character combination whose first character is ^X or ESC. The key sequence for any **emacs** command can be changed to a different key sequence to suit the preference of an individual user. Following conventional **emacs** notations, a two-character key sequence will be denoted in this section as two characters separated by a hyphen. Thus,

ESC-*char*   (stands for ESCape followed by any character)
^X-*char*   (stands for ^X followed by any character)

In **emacs** command listings, a key sequence is usually followed by the full **emacs** function name normally associated with it. A full function name in **emacs** often consists of several words connected by hyphens (next-line for example). The full name usually gives a pretty good idea of what a command does.

There are two ways to invoke the same **emacs** command

1. Typing the key sequence of the command

2. Typing `ESC-X` followed by the full command name

Entering and Exiting `Emacs`

To invoke **emacs**, you simply type

emacs *filename*

where *filename* is either an existing file or a new file. You can exit **emacs** by

```
^C                      (exit-emacs, no write)
```

Since this command does not write out the edited file(s), you have to issue

```
^S                      (save-buffer)
```

to save the current buffer. On some versions of **emacs**, the following also will work

```
^F                      (write-file-exit)
```

`Emacs` Windows

The CRT screen may be divided into one or more *windows*. A buffer is associated with each window, and a part of the buffer is visible through the window. At the bottom of a window is a *mode line*. Below the mode line is a *message line*. The message line is used for two purposes: to display messages to the user and to prompt for necessary input from the user.

The mode line, which varies from version to version, contains information about the editing activity in the window. Here is a sample mode line.

```
Buffer: file.c  File: /user/fac/pwang/file.c   27%
```

On a terminal window, the mode line is displayed in reverse video (black characters on white), or some other highlight mode. The word `Buffer:` is followed by a buffer name that is normally the last part of a Linux file pathname. The `27%` shows the position of the window relative to the whole buffer. In this example, the cursor is 27 percent of the way down from the beginning of the buffer. Sometimes, **emacs** will set up a *help window* that has no file associated with it.

Panic Stop in `Emacs`

Sometimes you may want to stop the currently executing command, either because it is taking too long, because you typed something wrong, or any other reason. In this circumstance in **emacs**, use

```
^G                      (abort)
```

which immediately stops all activity and returns you to the **emacs** top (command) level.

`Emacs` Cursor Movement

In **emacs** the editing position is always relative to the current position which is known as the *point*. The point is not shown on the screen, but it is between the cursor and the character before the cursor. Moving the cursor, and therefore the point, allows you to view the file and move to the location where modifications are needed.

For the reverse search command `^R`, **emacs** prompts you with

```
Search for:
```

in the message line and waits for you to type in a character string. (In some versions, **emacs** will carry out the search as you type the search string). To terminate the search string press `ESC`. When responding to a prompt, you may correct typing mistakes with Linux command line editing. To repeat a previous search, enter `^R` with an empty search string.

Insert and Delete in `Emacs`

There is no need for an *insert* command in **emacs**. Characters insert themselves at the point as you type them.

To insert a control character or the character `ESC` precede the character with `^Q` (the quote-character command). Two control characters are exceptions to this rule and they insert themselves: `^I` inserts a TAB, `^J` inserts a new line and tabs it appropriately. The delete commands are self-explanatory. For the delete word commands, the word deleted is from the point (cursor) to the end of the word. Thus, if the point is in the middle of a word and one of the delete word commands is typed, only part of the word will be deleted.

Frequent `emacs` Commands

The key sequence is followed by the full command name in the tables here.

Cursor motion			
`ESC-<`	beginning-of-file	`ESC->`	end-of-file
`^N`	next-line	`^P`	previous-line
`^A`	beginning-of-line	`^E`	end-of-line
`^F`	forward-character	`^B`	backward-character
`^T`	scroll-one-line-up	`ESC-z`	scroll-one-line-down
`^V`	next-page	`ESC-v`	previous-page
`^X-l`	goto-line	`ESC-]`	forward-paragraph
`^S`	search-forward	`^R`	search-reverse

Deletion			
`^I`	insert TAB	`^Q`	quote-character
`^J`	newline-and-indent	`^O`	newline-and-backup
`^D`	delete-next-character	`^?`	delete-previous-character
`ESC-d`	delete-next-word	`ESC-DELETE`	delete-previous-word

Cut and paste			
`^Y`	yank-from-killbuffer	`^W`	delete-to-killbuffer
`^@`	set-mark		(from mark to point)
`^K`	kill-to-end-of-line	`ESC-w`	copy-region-as-kill

Window Operations			
`^X-n`	next-window	`^X-p`	previous-window
`^X-1`	delete-other-windows	`^X-2`	split-current-window
`^X-z`	enlarge-window	`^X-^Z`	shrink-window

File I/O			
`^X-^R`	read-file	`^X-^I`	insert-file
`^X-^F`	write-file-exit	`^X-^M`	write-modified-files
`^X-^S`	write-current-file	`^X-^V`	visit-file
`^X-^W`	write-named-file		

Miscellaneous			
`^L`	redraw-display	`^G`	abort-command
`ESC-x`	full-command-name	`^X-^U`	undo
`ESC-?`	apropos	`^Z`	pause-emacs
`^U`	argument-prefix	`ESC-`*number*	repeat-command

For More Information

The newsgroup for **emacs** is `comp.emacs` and questions and answers regarding text editors can be found in the `comp.editors` newsgroup including the FAQs for **vi** and **emacs**.

Books on **emacs** include: *Learning GNU Emacs*, by by Debra Cameron and Bill Rosenblatt (O'Reilley and Associates), and *GNU EMACS Manual* by Richard Stallman, (Free Software Foundation).