A Guide to GNU Emacs

Timothy A. Gonsalves

TeNeT Group
Dept of Computer Science & Engineering
Indian Institute of Technology, Madras - 600 036
E-mail: tag@ooty.TeNeT.res.in

September 1999

Contents

1 Introduction

GNU Emacs is a powerful text editor that is especially well-suited for programmers. Besides a wide range of useful features, it is customisable to your preferences. It is extendible. Most of its commands are written in *ELisp*, a LISP-like macro language. You can modify existing *Emacs* commands and writing your own commands in ELisp. *Emacs* is a part of Linux distributions. The source code is freely available and can be compiled and installed on most other operating systems.

This guide contains a brief summary of the more commonly used *Emacs* commands. If you want to get started editing *Emacs* without reading the whole guide, Section 2 gives the *essential Emacs* commands.

If this guide succeeds in whetting your appetite, you can find more details in the *GNU-Emacs Manual* by R.M. Stallman and the online *Info* pages (Section 3.2).

1.1 Typographical Conventions

Anything in `typewriter` font in this guide should be typed as-is. Words in *italics* are replaced by names of your choosing. The following conventions are used to refer to specific keys:

`Esc`	Press the "Escape" key.
`k`	Press the key labelled "k".
`Ctrl-k`	Hold down the "Ctrl" key and press "k".
`Esc` `k`	Press and release the "Escape" key, then press "k".
`save`	Type the word "save".

Note In the *GNU-Emacs Manual* and on-line help, the following conventions are used:

C-k	equivalent to `Ctrl-k`
M-k	equivalent to `Esc` `k` (pronounced "meta-k", since early terminals had a key labelled "meta" similar to the control key).
C-M-k	equivalent to `Esc` `Ctrl-k`

2 Essential *Emacs* Commands

Using only the commands in this Section, you can use *Emacs* to do basic editing of programs and other text files.

Starting and Stopping Type `emacs` *file* at the shell prompt to start editing *file*.

`Ctrl-x` `Ctrl-c` exit from *Emacs*. Offers to save any files that you have modified but not saved.

`Ctrl-g` cancels a command that you have started typing, or stops a command that is executing. Type `Ctrl-g` one or more times if necessary. Typing `Ctrl-g` never does any harm.

Files

`Ctrl-x` `Ctrl-s` save the changes you have made to the current file. *Use this frequently.*

`Ctrl-x` `Ctrl-f` find an existing file for you to read and/or edit. Or, create a new file.

Cursor Movement

`Ctrl-f` move right one character

`Ctrl-b` move left one character

`Ctrl-n` move down one line

`Ctrl-p` move up one line

`Ctrl-v` scroll forwards by one page (screenful)

`Esc` `v` scroll backwards by one page (screenful)

`Esc` `<` move to the start of the file

`Esc` `>` move to the end of the file

On many terminals, the arrow keys, `PageUp`, `PageDn`, `Home` and `End` keys work with *Emacs*. Try them.

Insertion and Deletion To insert text at the cursor position, simply type it.

`Ctrl-d` or `Del` delete the character under the cursor

`Backspace` delete the character to the left of the cursor

`Ctrl-k` *kill* from the cursor to the end of the line. Press `Ctrl-k` twice to delete the line completely.

`Ctrl-y` *yank* back the last killed text. Use with `Ctrl-k` to cut and paste lines of text.

`Ctrl-x` `u` undo the last insertion or deletion. Use repeatedly to undo recent changes.

Search and Replace

`Ctrl-s` Search forwards towards the end of the file for a string (prompts you for the string). *Emacs* searches *incrementally*. As soon as you type the first character in the search string, *Emacs* searches for it and moves the cursor to the character that it finds. This occurs for every character you type. To end the incremental search, press `Enter`.

`Ctrl-r` Search backwards towards the beginning of the file for a string, similar to `Ctrl-s`.

`Esc` `%` replace several occurrences of one string to another. You are prompted for the old string (for example, `vi`) and the new string, (for example, `emacs`). *Emacs* then searches from the cursor to the end of the file for `vi`. It places the cursor after each occurrence of `vi` and waits for you to type one of the following:

`y`	replace this occurrence and go on
`n`	skip this occurrence and go on to the next
`!`	replace all remaining occurrences without asking
`Enter` or `q`	exit from query-replace.

Windows Sometimes, *Emacs* divides the screen into two windows. You can also do this yourself and edit different files in each window.

`Ctrl-x` `1` make the current window occupy the entire screen.

`Ctrl-x` `2` split the current window into two windows.

`Ctrl-x` `o` switch to the other window.

3 Getting Started

3.1 In Case of Difficulty

`Ctrl-g` cancels a command that you have started typing, or stops a command that is executing. Type `Ctrl-g` one or more times if necessary. Typing `Ctrl-g` never does any harm.

`Ctrl-l` redraw the screen if it gets messed up because, for example, of messages written by some other user. The line with the cursor is centred.

`Ctrl-x` `u` use repeatedly to undo recent changes.

`Ctrl-x` `Ctrl-c` exit from *Emacs* permanently. Offers to save any files that you have modified but not saved. Normally do this only when you want to logout.

`Ctrl-h` gets you help (see Section 3.2)

On some terminals, when you type `Ctrl-s`, the screen may freeze with subsequent keys being ignored. This happens because the terminal is using `Ctrl-s` for flow control. Typing `Ctrl-q` may solve this problem. If this happens, avoid using `Ctrl-s` and `Ctrl-q` until you can get some administrator to turn off the flow control.

3.2 Getting Help

An online tutorial is available that gives an interactive introduction to basic editing with *Emacs*. To use the tutorial, start *Emacs* by typing `emacs` at the shell prompt, then press `Ctrl-h` `t` and follow the instructions.

For comprehensive coverage of *Emacs*, read the *Info* pages. Type the *Emacs* command `Ctrl-h` `i` to start *Info*. When *info* starts, type `memacs` `Enter`. The Info pages are even more comprehensive than the *GNU-Emacs Manual*.

`Ctrl-h` `a` *topic* for a list of commands that deal with a particular topic, use the *apropos* command. For example, to find out how to save a file, type: `Ctrl-h` `a` `save` `Enter`.

`Ctrl-h` `k` *keys* describes what *keys* does.
Type `Ctrl-h` `k` `Esc` `a` to learn what `Esc` `a` does.

`Ctrl-h` `f` *command-name* `Enter` gives a detailed description of what *command-name* does.
Type `Ctrl-h` `f` `backward-sentence` `Enter` for a description of the command (or function) *backward-sentence*.

`Ctrl-h` `m` Lists special commands that are available in the current buffer. E.g., Pascal-mode commands, C-mode commands, and so on.

3.3 Useful General Commands

`Esc` *num* repeats the next commands *num* times. `Esc` `7` `Ctrl-f` moves forwards 7 characters (just as though you had pressed `Ctrl-f` 7 times).

`Ctrl-u` the same as `Esc` `4` — repeats the next command 4 times. `Ctrl-u` `Ctrl-u` repeats the next command 16 times.

`Ctrl-o` insert a blank line to the right of the cursor.

`Ctrl-z` suspends *Emacs* and returns you to the shell prompt. *Emacs* is not terminated. The shell command `jobs` will show suspended jobs including *Emacs*. To resume *Emacs* where you left off, type `fg` at the shell prompt. You will be back exactly where you were when you pressed `Ctrl-z`. This is the recommended method of leaving *Emacs* to run shell commands. Use `Ctrl-x` `Ctrl-c` only when you want to logout.

`Esc` `x` *cmd-name* execute any command by typing its full name. `Esc` `x` `suspend-emacs` `Enter` is equivalent to typing `Ctrl-z`.

4 Basic Editing

4.1 Files and Buffers

When you edit a file, *Emacs* makes a temporary copy of the file in a *buffer* in memory. Any changes you make affect only the copy in the buffer. To make your changes permanent, you must explicitly save the file to disk using Ctrl-x Ctrl-s.

Emacs has one buffer for each file you are editing. Every buffer has a name. This is usually related to the name of the file. *Emacs* may also create buffers which are not associated with any files. Examples are `*Help*` which the Help command Ctrl-h uses to display text; `*mail*` in which you compose a mail message; `*shell*` in which you can issue Shell commands.

Ctrl-x Ctrl-f find an existing file for you to read and/or edit. Or, create a new file.

Ctrl-x i insert the contents of a file after the cursor.

Ctrl-x Ctrl-s save the changes you have made to the current file. *Use this frequently.*

Ctrl-x s for each file that you have modified but not saved, *Emacs* offers to save it. Answer y or n as you wish. *Use this often!*

Ctrl-x b switch to another buffer. *Emacs* prompts you for the buffer name. The default is the buffer you were last in.

Ctrl-x Ctrl-b display a list of all the buffers *Emacs* has. The list includes the number of characters in each buffer and the associated file name, if any.

Ctrl-x k close a buffer. *Emacs* prompts for the buffer name. Just type Enter to close the current buffer. If the buffer has been modified but not saved, *Emacs* offers to save it. If you close a buffer without saving it, the changes are lost forever.

4.2 Moving About

Ctrl-b move left one character

Ctrl-f move right one character

Esc b move left one word

Esc f move right one word

Ctrl-a move to start of line

Ctrl-e move to end of line

Ctrl-p move up one line

Ctrl-n	move down one line
Esc v	scroll backwards by one page (screenful)
Ctrl-v	scroll forwards by one page (screenful)
Esc <	move to the start of the buffer (file)
Esc >	move to the end of the buffer (file)

These commands are mnemonic. Often, the Ctrl commands work over shorter units while the Esc commands work over longer distances, such as characters versus words (Ctrl-f versus Esc f). In other cases, the Ctrl commands operate forwards, while the Esc commands operate backwards (Ctrl-v versus Esc v).

See also Section 4.3 "Searching" for a convenient and fast way of moving large distances.

4.3 Searching

Ctrl-s	Search forwards towards the end of the file for a string (prompts you for the string)[1]
Ctrl-r	Search backwards towards the beginning of the file for a string (prompts you for the string)

Emacs searches *incrementally*. After you type Ctrl-s or Ctrl-r, as soon as you type the first character in the search string, *Emacs* searches for it and moves the cursor to the character that it finds. This occurs for every character you type. Typing Ctrl-s or Ctrl-r searches for the next occurrence of the string. To end the incremental search, press Enter. After ending the search, if you wish to go back to where you were when you started searching, press Ctrl-x Ctrl-x.

While searching, Ctrl-g will quit and put the cursor back where it was when you started the search. To repeat the previous search, type Ctrl-s Ctrl-s or Ctrl-r Ctrl-r.

4.4 Editing Text

Backspace	delete the character to the left of the cursor.
Ctrl-d or Del	delete the character under the cursor
Esc Backspace	delete the word to the left of the cursor.
Esc d	delete the word to the right of the cursor.
Ctrl-k	delete from the cursor to the end of the line. Press Ctrl-k twice to delete the line completely.
Esc k	delete from the Cursor to the end of the sentence.

[1] On some older terminals, Ctrl-s locks the keyboard until you type Ctrl-q. If this happens, use Ctrl-\ for forward search.

4.5 Replacing

To change several occurrences of one string to another, for example, to correct a common typo, use Esc %, the *query-replace* command. You are prompted for the old string (for example, `vi`) and the new string, (for example, `emacs`). *Emacs* then searches from the cursor to the end of the file for `vi`. It places the cursor after each occurrence of `vi` and waits for you to type one of the following:

Space or y	replace this occurrence and go on (press the space bar)
,	replace this occurrence but do not move the cursor immediately
Del or n	skip this occurrence and go on to the next
!	replace all remaining occurrences without asking
Enter or q	exit from query-replace.

Emacs tries to preserve the case of the word being replaced: `Vi` is replaced by `Emacs` and `VI` by `EMACS`. As in incremental search, *Emacs* sets a mark when you start the replace command. To get back to the starting point, just type Ctrl-x Ctrl-x.

4.6 Cutting, Copying and Pasting – the Region

To copy or move text from one place to another, you define a *region* of text by setting a mark at one end and moving the cursor to the other end. Once a region has been defined, you can perform various operations on the region, such as delete, copy or print the region.

Ctrl-2	Set the mark at the cursor location. The message: Mark set appears at the bottom of the screen. Note: on some keyboards, you may have to use Ctrl-@ or Ctrl-Space to set the mark.
Ctrl-x Ctrl-x	Set the mark here and move the cursor to the other end of the region. The region is unchanged. Repeating this causes the cursor to jump from one end of the region to the other so you can see what the region is.
Ctrl-w	delete the region. The text is placed in a *kill buffer* from which it can be pasted back later.
Esc w	copy the region to the kill buffer without deleting it.
Ctrl-y	yank back the last killed text. Note that the commands that delete words, lines and sentences described in Section 4.4 also place the deleted text in a kill buffer.
Esc y	immediately after Ctrl-y, use this to retrieve an earlier deleted text. Repeat this several times till you get the text you want inserted.

Emacs has a ring of kill buffers. You can traverse the kill buffer using Esc y repeatedly.

4.7 Windows

Emacs treats the screen as a window through which you look at a buffer. Usually you see only part of the buffer. You can split the screen vertically and/or horizontally into 2 or more windows. Each window can be focussed on a different buffer or different parts of the same buffer. Windows are divided vertically by highlighted mode lines and horizontally by columns of vertical bars.

`Ctrl-x 1` make the current window occupy the entire screen.

`Ctrl-x 2` split the current window into two windows, one above the other.

`Ctrl-x o` switch to the next window. Repeat this to go to each window in turn.

`Ctrl-x ^` make the current window one line larger. `Ctrl-u Ctrl-x ^` makes it 4 lines larger.

`Ctrl-x 3` split the current window into two windows, side by side. This is useful for comparing two files line by line.

`Ctrl-x {`, `Ctrl-x }` make the current window narrower or wider respectively.

`Ctrl-x >`, `Ctrl-x <` scroll the current window right or left, respectively. Useful if you have side by side windows and the lines of text are truncated (indicated by a "$" at the left and/or right edge of the window.

4.8 Keyboard Macros

Emacs has commands to do most things you would want to do, and many more besides. However, there are times when you find yourself typing the same series of *Emacs* commands repeatedly. In such cases, you can create a special command called a *keyboard macro* that has the same effect as the whole series. You create the command by showing *Emacs* what to do. *Emacs* memorizes your keystrokes. Then, you merely type this special command and *Emacs* repeats all the keystrokes it has memorized.

`Ctrl-x (` tells *Emacs* to start remembering your keystrokes. The mode line changes to include `(....Def)` indicating that you are defining a command.

`Ctrl-x )` tells *Emacs* to stop remembering your keystrokes. You may now use the keyboard macro that you have defined.

`Ctrl-x e` execute the keyboard macro you have defined. To execute it several times, give a repeat prefix, for example, `Esc 7 Ctrl-x e` repeats it 7 times.

You may have only one such keyboard macro at a time. When you define a new one, any earlier ones are lost. When you exit from *Emacs* permanently, the keyboard macro is lost.

It is usually a good idea to start defining the command with the cursor at the start of the line you are going to change and end it with the cursor at the start of the next line to be changed.

You can use any *Emacs* command except Ctrl-g in a keyboard macro. However, a command that encounters an error and rings the bell will probably cause *Emacs* to stop remembering your keystrokes. In this case, the `(...Def)` disappears from the mode line.

Example 1 Suppose you want to change Pascal declarations

```
txBuf PktType;
msg MsgType;
...
```

to C declarations

```
PktType txBuf;
MsgType msg;
...
```

Position the cursor at the start of the first declaration and type:
Ctrl-x (Esc f Esc t Ctrl-n Ctrl-a Ctrl-x)
This defines a keyboard macro that moves the cursor forward one word, transposes the two words before and after the cursor, then moves the cursor to the beginning of the next line. Now, execute the keyboard macro on the second declaration by typing Ctrl-x e.

Example 2 Where possible, use searches (Ctrl-s or Ctrl-r) to move the cursor. For example, given the two lines:

```
one two:  xxxx
three:  yyyy
```

Ctrl-s : Enter will move the cursor from the start of either line to the ":". Moving right 8 characters Esc 8 Ctrl-f or 2 words and 1 character Esc 2 Esc f Ctrl-f will do the same on line 1 but not on line 2.

4.9 The Minibuffer

The last line on the screen is called the *Minibuffer*. *Emacs* uses it to display messages and to prompt you for input such as file names. You can use the usual *Emacs* commands to move about in the minibuffer or switch to other windows on the screen. In addition, the following commands can save you a lot of typing and can aid your memory. Try them!

? give a list of all possible completions of what you have typed so far. For example, if *Emacs* asks you for file name, typing e? will get you a list of files beginning with `e`.

Tab complete as much of what you have typed as possible. For example, if you are typing a file name, and have typed e, typing Tab will look in your directory for a files beginning with `e`. If only one is found, the entire file name is typed. If there are more than one, *Emacs* waits for you to type more of the name to indicate which one you want.

`Space` complete up to the end of the next word, if possible (similar to `Tab`).

`Enter` ends your response to *Emacs*' prompt.

5 Advanced Editing

5.1 Commands for Programs

When you edit a program source file, *Emacs* enters a mode in which several extra commands are available that are tailored for the particular language. *Emacs* decides which mode to enter depending on the file name: for example, c-mode for `.c` files, pascal-mode for `.pas` files.

Pascal Mode In Pascal mode, the string `(Pascal)` appears in the mode line. Some special commands (type `Ctrl-h` `m` for a complete list):

`Esc` `*` insert a comment block

`Esc` `Ctrl-a` go to the beginning of the current function or procedure.

`Esc` `Ctrl-e` go to the end of the current function or procedure.

`Esc` `Ctrl-h` mark the current function or procedure — useful to copy or delete the function or procedure.

`Ctrl-c` `Ctrl-c` comment out the current region.

`Ctrl-c` `Ctrl-u` uncomment the text commented by `Ctrl-c` `Ctrl-c`.

`Ctrl-c` `Ctrl-b` insert a `begin..end` pair.

`Ctrl-c` `Ctrl-d` go to a function/procedure (prompts for name in the minibufffer at the bottom of the screen).

`Enter` goes to the next line and indents it depending on the level of nesting of the code. If you are editing a comment block, inserts " `*`" at the beginning of the line.

`Tab` indents the current line.

C/C++/Java Mode In C, C++ and Java mode, the string `(C)` or `(C++)` or `(Java)` appears in the mode line. Some special commands (type `Ctrl-h` `m` for a complete list):[2]

`Esc` `;` insert a comment on the current line.

`Esc` `a` go to the beginning of the current statement.

[2]The commands listed here assume that you have loaded `DONc.el`.

Esc e — go to the end of the current statement.

Esc Ctrl-h — mark the current function — useful to copy or delete the function.

Ctrl-j, Ctrl-Enter — goes to the next line and indents it depending on the level of nesting of the code.

Tab — indents the current line.

Esc Ctrl-q — indent the block starting from the cursor.

To save on typing, Ctrl-c followed by various other keys inserts commonly-used code templates. This includes comments and proper indentation. You need only fill in the details.

Ctrl-c u — insert a template for a new function. Prompts for the function name and return type.

Ctrl-c t — insert a comment block at the top of the file. *Use this for every file.*

Ctrl-c w — insert a while loop.

Ctrl-c f — insert a for loop.

Ctrl-c d — insert a do-until loop.

Ctrl-c s — insert a switch statement.

Ctrl-c i — insert an if statement.

Ctrl-c e — insert an if-else statement.

5.2 Commands for Text

Emacs has several commands that are useful for editing and simple formatting of text files such as mail messages.

Changing Case

Esc c — capitalize the word to the right of the cursor

Esc l — change the word to the right of the cursor to lower-case

Esc u — change the word to the right of the cursor to upper-case

Spelling

`Esc` `$` check the spelling of the word under the cursor.

`Esc` `x` `ispell-region` `Enter` interactively check the spelling of the region (see Section 4.6 for setting the region).

`Esc` `x` `ispell` `Enter` interactively check the spelling of the buffer.

If *Emacs* finds an incorrect word, it highlights the word, gives several possible corrections and the top of the screen, and prompts for your choice. If none of these are correct, type `r` and *Emacs* will prompt for a replacement of the word in the minibuffer.

Miscellaneous

`Ctrl-t` while typing, transpose the two characters to the left of the cursor. While correcting, transpose the character under the cursor with that before the cursor. Typing `Ctrl-t` repeatedly drags a character forward.

`Esc` `t` transpose the words to the left and right of the cursor.

`Esc` `q` fill the paragraph under the cursor, making all lines approximately the same length.

`Esc` `g` fill all paragraphs in the region (see Section 4.6 for setting the region).

5.3 Abbreviations

Emacs lets you define abbreviations for words and phrases that you type frequently. Whenever you type an abbreviation, it is replaced by the full form. For example, you define `csd` to be an abbreviation for `Computer Science Department`. Every time you type the word `csd` followed by a `Space`, `Enter` or punctuation mark, *Emacs* expands `csd` to `Computer Science Department`. Naturally, you would choose abbreviations that you do not normally type as words by themselves.

Abbreviations can also be used to automatically correct your common spell mistakes. For example, define `dealy` to expand to `delay`, and `itme` to expand to `item`.

`Ctrl-x` `a` `g` add a global abbreviation for the word to the left to the cursor. *Emacs* prompts you for the abbreviation. This has effect in all buffers.

`Ctrl-x` `a` `i` `g` (inverse of the above) define the word to the left to the cursor as a global abbreviation. *Emacs* prompts you for the expansion.

`Ctrl-x` `a` `l` add a local abbreviation for the word to the left to the cursor. *Emacs* prompts you for the abbreviation. This has effect only in buffers sharing the same *mode*, such as C-mode or Text-mode.

`Ctrl-x` `a` `i` `g` (inverse of the above) define the word to the left to the cursor as a local abbreviation. *Emacs* prompts you for the expansion.

`Ctrl-x` `,` undo the last expansion. Use this when *Emacs* expands a word that you do not want expanded.

`Esc` `x` `abbrev-mode` `Enter` toggles abbrev-mode — only when abbrev-mode is on (look for `...Abbrev...` in the mode line) does *Emacs* automatically expand abbreviations.

`Esc` `x` `write-abbrev-file` `Enter` *Emacs* prompts for the name of a file in which to save the abbreviations you've defined. `/.abbrevs` is a common choice.

`Esc` `x` `read-abbrev-file` `Enter` *Emacs* prompts for the name of a file from which to read saved abbreviations, such as `/.abbrevs`.

To have abbrev-mode automatically entered whenever you use *Emacs*, you need to edit your `~/.emacs` file (see Section 6.4).

5.4 Shuffling Files

Emacs has commands to let you print, copy, delete and rename files. These are equivalent to the shell commands of similar names, except that while typing a file name, you can type `?` to see a list of your files, and use the other completion keys (see Section 4.9).

`Esc` `x` `copy-file` `Enter` *file1 file2* `Enter` copy *file1* to *file2*. *Emacs* will ask for confirmation if *file2* already exists. Answer `yes` to overwrite *file2*, `no` to abort the command.

`Esc` `x` `rename-file` `Enter` *file1 file2* `Enter` change the name of *file1* to *file2*. Use this to move a file from one directory to another.

`Esc` `x` `delete-file` `Enter` *filename* `Enter` delete *filename*. To delete several files in a directory, see Section 5.5 Editing Directories.

5.5 Editing Directories

A convenient way to clean-up a directory is to edit it using *Emacs*' *Dired* mode. To clean-up, say, `work/src`, find it as you would a file: `Ctrl-x` `Ctrl-f` `work/src`. *Emacs* creates a buffer named `src` with `(Dired)` in the middle of the mode line. The buffer contains one line for each file, the result of running `ls -l` on the directory. For example:

```
/home/juniper/tag/work/src:
...
-rw-r--r-- 1 tag 10254 Dec 2 17:14 fileutils.c
...
```

This says that `fileutils.c` has 10,254 characters and was last edited at 17:14 on December 2^{nd}. It is owned by user `tag` who can read or write it. All other users can read but not write the file.

In Dired, you may type several single-letter commands to do things with the file on whose line the cursor is:

`?` display a list of commands at the bottom of the screen

`d` mark this file for deletion. A `D` in the first column of the line indicates that the file is to be deleted when you type `x`. The cursor automatically moves to the next file.

`u` undelete this file, that is, remove the `D` mark.

`v` view the contents of the file. You can use the usual cursor-movement commands to move around in file. `Space` scrolls forward one page. Type `Ctrl-c` to get back to Dired.

`x` expunge files marked with `D`. *Emacs* will show you a list of marked files and ask you to confirm with `yes` or `no` before the files are deleted. Once deleted the files cannot be recovered.

`c` copy the file to some other name. Works just like `Esc` `x` `copy-file`.

`r` rename the file or move it to another directory. Works just like `Esc` `x` `rename-file`.

`f` find the file in a buffer for editing. Works just like `Ctrl-x` `Ctrl-f`.

`g` read the directory again.

When you are done with Dired, close the buffer (`Ctrl-x` `k`), or just switch to another file or buffer (`Ctrl-x` `Ctrl-f` or `Ctrl-x` `b`)

6 Customizing *Emacs*

Besides defining keyboard macros (Section 4.8), *Emacs* can be customized in several ways. For instance, you can bind commands to keys, set the value of variables that control the way *Emacs* behaves, and alter the behaviour of *Emacs* in various modes. When *Emacs* starts up, it reads the file `~/.emacs` and executes any commands contained it it. This is used to make customisations permanent.

6.1 Binding Commands to Keys

All commands in *Emacs* have a name and you can execute them using:
`Esc` `x` *command-name* `Enter`.
Commonly-used commands are bound to keys, for example, `insert-file` is bound to `Ctrl-x` `i`. Note that commands are also referred to as *functions*.

You can change the bindings of commands to keys. For instance, if you prefer that Ctrl-x Ctrl-i should insert a file, you can do this by typing:

Esc x global-set-key
Set key globally: Ctrl-x Ctrl-i
Set key C-x TAB to command: insert-file Enter[3]

To make bindings permanent, place them in `~/.emacs`. For example, the following line added to `.emacs` makes Ctrl-x Ctrl-i insert a file in the current buffer at the cursor position:[4]

```
(global-set-key "\C-x\C-i" 'insert-file)
```

6.2 Variables

The behaviour of *Emacs* is controlled by a number of variables. Some of these are global, some are specific to a mode and others are local to a buffer. To see the value of a variable and its description, type Ctrl-h v *variable-name*. Note that you can use completion keys while typing the variable name (see Section 4.9).

For example, to save you from losing very much work in case of power failures or other system crashes, *Emacs auto-saves* your file periodically. This is controlled by two variables: after every *auto-save-interval* keystrokes, or if you have been idle for *auto-save-timeout* seconds, *Emacs* saves your file in a temporary file with the name #*filename*#. The default values are 300 keystrokes and 10 seconds. If you wish to auto-save every 100 keystrokes, type:

Esc x set-variable Enter *Set variable:* auto-save-interval Enter
Set auto-save-interval to value: 100 Enter

To make this permanent, put the following line in your `.emacs`:

```
(setq auto-save-interval 100)
```

6.3 Hooks: Customising a Mode

When *Emacs* does certain operations such as finding a new file, or entering a special mode, it can be made to execute some commands by means of a *hook*. For instance, when *Emacs* enters C-mode (Section 5.1) to edit a C source file, you might want *Emacs* to automatically go to the next line when you type the ; at the end of a statement. To do so, add the following lines to your `.emacs`:

```
(setq c-mode-common-hook ; applies to C, C++ and Java
    '(lambda ()
        (setq c-auto-newline t)))
```

[3] Note that Ctrl-i and Tab are equivalent.

[4] Changes to `.emacs` take effect the next time *Emacs* is started. To have them take immediate effect, while you are in the `.emacs` buffer, type the command Esc x eval-current-buffer Enter.

6.4 Miscellaneous

Your `.emacs` can contain any sequence of *Emacs* commands that you want executed on start-up. For example, if you have defined some abbreviations in the file `~/.abbrevs` and want to always use abbreviation mode (Section 5.3), add the following lines to your `.emacs`:

```
(quietly-read-abbrev-file "~/.abbrevs")
(abbrev-mode 1)
```

Thereafter, if you add or change your abbreviations and try to exit, *Emacs* will ask you:

Save abbrevs in ~/.abbrevs? (y or n)

Press `y` if you want to save the new abbreviations. If you press `n`, your changes will be lost.

6.5 *Emacs* Lisp: Writing your own *Emacs* Commands

Emacs lets you write your own commands in a Lisp-like language called Emacs *Lisp*. In fact, most of the *Emacs* commands that you are now familiar with are written in *Emacs*Lisp. The source for these commands are kept in `.el` files in the `/lisp` subdirectory of *Emacs* (usually `/usr/share/emacs/`*nn.m*`/lisp`, where *nn.m* is the version number of *Emacs*, such as 20.3). Reading some of these files is a good way to learn to write your own commands. Also, see the *GNU-Emacs Manual* and the online information (`Ctrl-h` `i`).