The Emacs Editor

Tim Love

July 1, 1998

Contents

1 Emacs Terminology **2**

2 Help, short cuts and common operations **2**
2.1 Help 2
2.2 Error recovery 3
2.3 Common Commands 3
2.4 Completion 3
2.5 Manipulating blocks of text 4
2.6 Numeric Arguments 4
2.7 Multiple files 4

3 Concepts **4**
3.1 Modes 4
3.2 Buffers 5
3.3 Windows 5
3.4 Rings 5
3.5 Registers 6
3.6 Abbreviations 6
3.7 Variables 6
3.8 Macros 7
3.9 Tags 7

4 Customising **8**
4.1 Arguments and Xdefaults 8
4.2 Using the `.emacs` file 9
4.3 Saving macros 10
4.4 `emacsclient` 10
4.5 Big `.emacs` files 11

5 Key bindings **11**
5.1 Interactive bindings 11
5.2 Using the `.emacs` file 11

6 Miscellaneous **11**
6.1 Sorting 12
6.2 Compiling 12

Introduction

`Emacs` is a self-documenting, customisable, extensible text editor which is freely available for most types of computers.

A crib sheet is available for `emacs` and there is extensive on-line documentation within `emacs` as well as a tutorial. Further documentation (plus a copy of this document) is available from the help system's Emacs page[1]

This document only looks at the under-used or undocumented features. Contributions to this document will be welcomed. Mail `tpl@eng`.

1 Emacs Terminology

The `emacs` documentation uses some terms that it's useful to understand. This document will use terms like *window* in the same way as the `emacs` documentation does, rather the the way these terms are usually used.

Buffers - `emacs` loads files into internal *Buffers*; it loads *Files* but saves *Buffers*.

Frames - One `emacs` process can produce a number of `X` windows. `Emacs` calls each of these windows a *frame*.

Windows - Each frame can be split into sections that `emacs` calls *windows*.

Control Keys - '`M-`' means '*hold the* `Extend char` *key down*' (this key is called `Meta` in the emacs documentation) and '`C-`' means '*hold the* `CTRL` *key down*'. In general, `CTRL` key functions do standard things while `Meta` key functions do something extra. For example, `C-f` advances by a letter and `C-d` deletes a letter whereas the `M-` equivalents deal with word units. Using `CTRL` and `Meta` together usually calls a more specialised activity.

point - The *point* is where the text cursor is.

regexp - A *regexp* is a Unix Regular Expression; *i.e.* a string for pattern matching. Type `man ed` for more details on these, or look in `Emacs`' own help system.

Lisp - `Emacs` is not a self-contained binary. Many of `emacs`' commands are written in a language called Lisp. To call one of these functions by name, type `M-x` *function name*. The files containing these functions are in the directory `/usr/src/editors/emacs/lisp`. You can write your own lisp functions, but nearly all users will be able to do what they want without having to do this.

2 Help, short cuts and common operations

2.1 Help

- emacs' `Help` has various options for you to discover more about `emacs`. Some of these option split the window into 2. Use the `Just One Window` option at the bottom of the help menu to return to having a single window.
- `Info` is the definitive source of `emacs` documentation, containing far more information than the online manual page. Move the pointer around and use the scrollbars to see the text. If a word becomes highlighted when the pointer passes over it, click with the middle-mouse-button to display more information on that subject. Type '`q`' to quit.

[1] http://www-h.eng.cam.ac.uk/help/tpl/emacs.html

- `Command Apropos ...` tells you which `emacs` commands have something to do with a keyword that you're invited to type in.
- `List Keybindings` lists the control-key sequences that run `emacs` commands.
- `Describe Key ...` tells you what a particular control-key sequence does.
- `Describe Function ...` gives a full description of an `emacs` function.
- `Emacs Tutorial` runs a tutorial.

2.2 Error recovery

To reverse the effect of the last editing command, use the `Undo` menu option or type `C-x u`.

Whenever you save on top of any existing file, the old file is backed up into a file whose name has an appended '`~`'. `emacs` also backs up your files automatically every so many key presses. The name of this auto-backed-up file is created from the original by adding a `#` to each end of the name.

2.3 Common Commands

In a window environment, the common commands are available via the menus. At CUED there are 2 sets of menus to choose from, one more extensive than the other. The bottom item of the `Edit` menu lets you flip between them. Many of the menu items also have control-key equivalents which are displayed in the menu. For instance the `File` menu contains an item `Save Buffer (C-x C-s)`. It's worth learning the short cuts for common operations, even if you're at a graphics terminal. Here's a list of some useful ones.

`C-g`	cancel
`C-s`	incremental search forward
`C-r`	incremental search backwards
`C-k`	kill line
`C-y`	yank line
`C-x C-f`	load in a new file
`C-x C-s`	save a file
`C-x C-w`	write to a new file

With 'incremental searching', searching commences as soon as you begin to type in the target string. Hitting the `Backspace` key will remove the last letter from the current target string unless you have repeatedly used a search command with the same target, in which case `emacs` will go back to the last instance of the target found.

2.4 Completion

Whenever you type in a filename or an `emacs` command, these options are available to you

- `TAB` - this will complete the name if there's a unique completion.
- `Space` - like `TAB` but it will only complete up to a space or hyphen.
- `?` - lists possible completions

2.5 Manipulating blocks of text

Using the Mouse - The Cut option in the Edit menu lets you remove blocks of text. To delete a block of text, place the cursor at the start of the text that you wish to remove and drag over the text holding the left mouse button down. Then click on Cut. A faster way to cut text is to click at one end with the left mouse button, then double-click at the other end with the right mouse button.

If you want to restore the text, just click on the new destination with the middle mouse button.

To copy some text, place the cursor at the start and hold the left mouse button down while dragging the pointer over the text you want to copy. Release the mouse button when you reach the end of the text. Now don't select Cut, just move the pointer where you want the text to be copied and press the middle mouse button. This facility also works between most kinds of text windows.

Using Keys - Mark one end of the region by moving the cursor there and using C-<Space>. Move the cursor to the other end of the region. Use C−w to pick up the text and remove it; M−w to just pick up. Move the text cursor where you want the text to be pasted and type C−y

If you want to manipulate rectangles of text, mark one corner of the region, move the cursor to a diametrically opposite corner and use these commands:-

```
C-x r k
C-x r y
```

2.6 Numeric Arguments

All emacs commands can take a numerical argument. For instance, to kill 5 lines you can do M−5 C−k or C−u 5 C−k. If no number follows C−u then a default numerical argument of 4 is used. Where it makes sense, negative numerical arguments are accepted. Some commands don't interpret the numerical argument as a request for repetion but as a flag to modify behaviour. E.g. C−<Space> sets a mark whereas C−u C−<Space> goes back through previous marks.

2.7 Multiple files

You *don't* need to have one emacs running for each file you're editing! emacs is designed to be kept running. This is easily done in at a graphics terminal. On a non-graphics terminal you can do C−z to suspend emacs without killing it, then bring it back into the foreground using fg.

3 Concepts

To customize emacs further it helps to know about some underlying ideas.

3.1 Modes

So that each type of text file can be treated appropriately, emacs supports various major modes of operation. In the status line you will see something like

```
--:**  emacs.tex    (TeX)--L1-----16%---------
```

The major mode of the buffer is in brackets. emacs tries to pick the most appropriate mode by looking at the file name, but you can explicitly set modes. For example, you can type M-x C-mode to enter a mode with support for those writing C programs.

Minor modes are mutually independant and independant of major modes. For instance, if you type M-x auto-fill-mode, the status line will display something like

```
--:**  emacs.tex    (TeX Fill)--L1-26%-------
```

and emacs will create a new line for you when you reach the right edge of your window.

To find out which modes are available and what they provide, look in emacs' Info files.

3.2 Buffers

All the commands you use to move text around within a file can be used to move text from one buffer to another. If you select the Buffers menu you can see the available buffers and pick which you want to visit.

If you type C-x C-b you will see something like

```
-----------------------------------------------------------
MR Buffer         Size  Mode              File
-- ------         ----  ----              ----
.  xcursors.1     303   Fundamental       /tmp/xcursors.1
   comp_sci       329   Fundamental       /tmp/notes/comp_sci
   *scratch*      0     Lisp Interaction
   *Buffer List*  0     Buffer Menu
-----------------------------------------------------------
```

This shows which mode each buffer has and what file, if any, it's associated with. To jump between buffers, use C-x b. By default you are invited to go to the last buffer you visited. You can save all the buffers by typing C-x s (rather than C-x C-s which just saves the current buffer).

Some types of buffers have their own special commands. For instance, the *Buffer List* buffer can't be written into but you can move the cursor around and select buffers for deletion (type d) or saving (type s) then type x for everything to happen. See the emacs Info section called *Several Buffers* for more info.

3.3 Windows

Emacs can split the screen into two or many windows, which can display parts of different buffers, or different parts of one buffer. You can use the mouse to move the text cursor into a window, or use C-x o

C-x 2	Split the selected window into two windows, one above the other
C-x 5	Split the selected window into two windows side by side
C-x 0	Get rid of the selected window
C-x 1	Get rid of all windows except the selected one
C-x ^	Make the selected window taller, at the expense of the other(s)
C-x }	Make the selected window wider

3.4 Rings

Each point that you mark is saved in a circular list or ring. Each buffer has its own *Mark Ring*. You can go back to these old marks by repeatedly typing C-u C-<Space>

The *Kill Ring* contains blocks of killed text. Emacs has only one kill ring, common to all buffers, allowing you to delete text from one buffer and insert it in another. Old deletions are saved so that you can, for instance, recover text 3 deletions back by doing `C-u 3 C-y`.

3.5 Registers

You can temporarily store away text or information in registers. Each register has a single character label denoted here by `<r>`.

`C-x / <r>`	to save position in register
`C-x j <r>`	to go to position saved in `<r>`
`C-x x <r>`	to save region
`C-x g <r>`	to get region saved in `<r>`
`C-x r <r>`	to save rectangle
`C-x g <r>`	to get rectangle (if a rectangle was saved in that register).

Use `M-x view-register` to see what a register contains.

3.6 Abbreviations

There's an `Abbrev` minor mode (set it using `M-x abbrev-mode`) which allows automatic expansion of abbreviations. If you find that you are often typing 'at this moment in time', you can easily arrange things so that typing 'waffle`<space>`' reproduces the phrase. You can define abbreviations interactively or from a file.

Interactive - To define a new abbreviation for a word, type the word, then type `C-x a g`. You'll be prompted for the abbreviation you want to use.

If you want to create an abbreviation for a phrase of (say) 3 words, use the standard emacs numerical argument facility. *E.g.*, typing `Save our Souls C-u 3 C-x a g sos` creates an abbreviation `sos`.

Use `M-x list-abbrevs` to list and `M-x edit-abbrevs` to edit abbreviations. Note that there are mode-specific abbreviations as well as global ones.

Using abbreviation files - `M-x write-abbrev-file` will put into a file the currently active abbreviations. `M-x read-abbrev-file` will read a file of abbreviations.

Dynamic abbreviation - If you type `M-/` `emacs` will try to complete the word you're on by looking back in the file and finding the first match. This facility is available even if the `Abbrev` minor mode isn't set.

3.7 Variables

`Emacs` keeps a list of variables for its own use. Many of them are switches to control `emacs'` behaviour. Find out about them by typing

```
M-x list-options
```

Change their values using one of the following –

```
M-x set-variable RET <var> RET <val> RET
M-x edit-options
```

3.8 Macros

To avoid repetitive key sequences you can use macros. Suppose you wanted to append '`.old`' to the next 50 words of text. Type

```
C-x (
```

to start defining a macro, then perform what you want the macro to do on the first word,

```
M-f .old
```

(`M-f` goes to the end of the current word) then end the macro definition using

```
C-x )
```

`C-x e` will execute this, the last defined macro, once. To perform it 48 more times, use `C-u 48 C-x e`.

To give this macro a name, use

```
M-x name-last-kbd-macro RET <macroname> RET
```

You can now use `M-x <macroname>` to call the macro, or use `global-set-key` to bind a key to the macro (see page 11).

3.9 Tags

To assist movement within a multi-file source, you can set up tag files which stores the location of functions definitions, chapter headings etc that you might want to jump to by name. The syntax of `C`, `Fortran` and LaTeX files are understood.

From the command line go to the directory where you'll be working from and type

```
etags <files>
```

(or `etags -t <files>` to record typedefs in C code too). This will create a file called `TAGS`. It doesn't matter if this file gets a bit out of date unless you move functions from one file to another. Just run `etags` again every so often. To load this tag file in, use

```
M-x visit-tags-table
```

which will prompt for a file name (the default is `TAGS`). Now you can use

```
M-. <tag>
```

to jump to the file and position of the definition of `<tag>`.

Since the `TAGS` file contains a list of files that you're interested in, you can use it to perform multi-file versions of emacs commands. Eg

```
M-x tags-search    (then M-, to resume the search)
M-x tags-query-replace
```

If you know you're going to visit a succession of files, use

```
C-u M-x next-file
```

to look at the first file mentioned in `TAGS` and

```
M-x next-file
```

to move to the next.

```
M-x list-tags
```

asks for a file and tells you the tags defined in it.

```
M-x tags-apropos
```

asks for a regexp and displays the matching tags.

4 Customising

Just about everything in `emacs` can be customised. The rest of this document concentrates on how this can be done.

4.1 Arguments and Xdefaults

Here's a list of the more useful arguments that emacs can take. See the man page or `emacs'` own help for details.

-f function Execute the lisp function called `function`.

-l file Load the lisp code in the file file.

-font font Choose a font.

-b pixels Set the Emacs window's border width to the number of pixels specified by pixels.

-ib pixels Set the window's internal border width to the number of pixels specified by pixels. Defaults to one pixel of padding on each side of the window.

-w =[WIDTH][xHEIGHT][+-XOFF[+-YOFF]] Set the Emacs window's width, height, and position on the screen. The `[]`'s denote optional arguments,

-fg color On color displays, sets the color of the text.

-bg color On color displays, sets the color of the window's background. See the file `/usr/lib/rgb.txt` for a list of valid color names.

-bd color On color displays, sets the color of the window's border.See the file `/usr/lib/X11/rgb.txt` for a list of valid color names.

-cr color On color displays, sets the color of the window's text cursor. See the file `/usr/lib/X11/rgb.txt`.

-ms color On color displays, sets the color of the window's mouse cursor. See the file `/usr/lib/X11/rgb.txt`

You can also set up emacs defaults in your `~/.Xdefaults` file. For instance, adding this to `~/.Xdefaults` will set many of the things that could have been set from the command line

```
emacs.font:             6x10
emacs.geometry:         80x40+360+215
emacs.borderColor:      VioletRed
emacs.cursorColor:      SkyBlue
emacs.pointerColor:     Yellow
emacs.borderWidth:      1
emacs.softButtons:      on
emacs.foreground:       PaleGreen
emacs.background:       DarkSlateGrey
emacs.bitmapicon:       on
emacs.iconName:         Emacs
```

4.2 Using the `.emacs` file

If you have an `.emacs` file in your `HOME` directory, this will be read when `emacs` starts up. This file contains Lisp calls. Here are some examples of things you can do

- Make searches case-sensitive

  ```
  (setq-default case-fold-search nil)
  ```

- Sets autofill on in text mode automatically.

  ```
  (setq text-mode-hook
        '(lambda() (auto-fill-mode 1)))
  or
    (setq text-mode-hook 'turn-on-auto-fill)
  ```

- Set `text` rather than `Lisp` to be the default mode.

  ```
  (setq default-major-mode 'text-mode)
  ```

- Arrange for the syntactic elements of C, fortran, pascal and LaTeX files to be displayed in different colors that display ok in a window with a *light* background (the CUED default is a dark background).

  ```
  (setq inhibit-default-init 1)
  (cond (window-system
         (setq hilit-mode-enable-list  '(not text-mode)
               hilit-background-mode   'light
               hilit-inhibit-hooks     nil
               hilit-inhibit-rebinding nil)

         (require 'hilit19)
         ))
  ```

- Disable coloring

  ```
  (setq inhibit-default-init 1)
  ```

- Use matlab-mode for matlab files

  ```
  (autoload 'matlab-mode "matlab-mode" "Enter Matlab mode." t)
  (setq auto-mode-alist (cons '("\\.m$" . matlab-mode) auto-mode-alist))
  ```

- Stop `C-s` freezing the screen when using `emacs` in a windowless environment

  ```
  (enable-flow-control)
  ```

- Don't show the startup message

  ```
  (setq inhibit-startup-message t)
  ```

- Set colors (as in `~/.Xdefaults`).

```
(set-border-color "lightblue")
(set-border-color "blue")
(set-cursor-color "red")
(set-foreground-color "black")
(set-mouse-color "red")
```

- Display the time in the status line

```
(load "time" t t)
(display-time)
```

- If you want to customise menus, look at the example files in the directory `/usr/src/editors/emacs/site-lisp`. Putting the following into your `.emacs` file will give you long menus by default

```
(load "/usr/src/editors/emacs/site-lisp/.long_menus.el")
(setq inhibit-default-init 1)
```

4.3 Saving macros

To save a macro into `~/.emacs` so that you will always be able to use it, make `~/.emacs` the current file and do

```
M-x insert-kbd-macro RET <macroname> RET
```

This will convert the macro into Lisp for you. Doing

```
M-x 4 insert-kbd-macro RET <macroname> RET
```

will save the relevent key bindings too.

4.4 `emacsclient`

Many other editors are designed to be started afresh each time you want to edit. You edit one file and then exit the editor. The next time you want to edit either another file or the same one, you must run the editor again. With these editors, it makes sense to use a command line argument to say which file to edit.

But starting a new `emacs` each time you want to edit a different file would be annoyingly slow and would fail to take advantage of `emacs`' ability to visit more than one file in a single editing session. If you

1. start `emacs` with the `emacsserver` running (this can be done by adding `(server-start)` to your `.emacs` file, or by typing `M-x server-start` from within `emacs`)
2. use `emacsclient` where you'd normally use `emacs` (as your editor in `pine`, as the value of the `EDITOR` environmental variable, etc)

you'll cause `pine`, etc to use the existing `emacs` rather than start a new one.

This facility goes wrong if you're in a windowless environment. Get round that by adding this to your `~/.profile`

```
# Sort out the terminal.
[ $TERM = network ] && TERM=xterm
# Set your default TERM here ^^^^^^^
if [ $TERM = "xterm" ]
then
     export EDITOR=emacsclient
else
     export EDITOR=emacs
fi
```

(thanks to `itc10@eng` for this tip)

4.5 Big `.emacs` files

If you have a big `.emacs` file you should compile the Lisp into a file (`estart.elc`, say) by copying `~/.emacs` into `~/estart.el` (the '`.el`' extension denotes 'Emacs Lisp') then using `M-x byte-compile-file` on this to create `~/estart.elc` (the '`.elc`' extension denotes 'Emacs Lisp Compiled') so that in your .emacs file you can just have `(load ``~/estart.elc'')`.

5 Key bindings

To make the most of the above customisation, you will need to set keys to perform particular functions.

Tables called keymaps store what keys do. There's a global keymap and one for each major mode.

5.1 Interactive bindings

The commands

```
M-x global-set-key RET <key> RET <cmd> RET
M-x local-set-key RET <key> RET <cmd> RET
```

are used to set key functions.

5.2 Using the `.emacs` file

Here you will need to know (or copy) a bit of Lisp. Here are some examples.

```
(global-set-key "\e#" 'quoted-insert)
(define-key c-mode-map  "\e#" 'quoted-insert)
```

These both set `M-#` to perform a quoted insert.

```
(global-set-key "\C-cg" 'goto-line)
(define-key ctl-x-map "g"    'goto-line)
```

Make `C-c g` and `C-x g` prompt for a line number to jump to.

The function keys can also be set to do things –

```
(global-set-key [f1] 'goto-line)  ;; goto line number
```

6 Miscellaneous

Some other commands you may like to investigate are :-

`M-h`	mark a paragraph
`C-x TAB`	indent a region
`C-x C-u`	make a region upper case
`M-x overwrite-mode`	to toggle insert/overwrite
`C-x f`	set right margin to point
`C-M-n`	move down, staying at same syntax level
`C-M-p`	move up, staying at same syntax level
`C-M-a`	move to start of function
`C-M-e`	move to end of function
`C-x C-q`	toggle the writeability of the file

You can add control characters to a file. To put a `C-m` character (which has `ascii` value 13) into a file, type `C-q` then type `C-m`. This ability to 'quote' control characters is useful if you have copied a file from another machine and find that each line has one of these characters on the end. Use query-replace (`M-%`), input `C-M`, (quoted as above), to be your target string and nothing as your replacement string.

6.1 Sorting

A region of text can be sorted within `emacs`. For example, if you want to sort the lines of a region so that the third words of the lines are in alphabetical order, define the region then type

```
M-x sort-fields 3
```

See the info documentation for more details and options.

6.2 Compiling

You can compile programs from within `emacs` by using `M-x compile`. This will prompt for a command line which will be used as the default for subsequent calls of `M-x compile`. If compilation fails, you can use ``C-x ` `` to move to the first reported error and then to subsequent errors.