Getting started with Emacs

Marc van der Sluys
Nikhef/Utrecht University
The Netherlands
http://pub.vandersluys.nl

April 20, 2022

Contents

I treat GNU Emacs [1] here, not XEmacs, Aquamacs or any other flavour. I used *Effective Emacs* [2] and *How to learn Emacs* [3] for inspiration and information. See also the official Emacs Reference card and Survival card for more shortcuts [4].

1 Why Emacs?

If you are going to write code for the next 50 years or so, choosing a text editor should get some serious attention. These were my wishes when choosing mine:

1. It should be available on many platforms. At the very least Linux, UNIX, BSD, Mac OS and Windows.
2. There should be a reasonable chance that it will be available in the next decades.
3. Since I don't want to learn a new trick for every different thing I do, it should support many languages. It should also be easily extensible to new languages, since I don't know yet what I will be programming in 20 years from now. (Also, that would mean that many third-party plugins would be available.) Currently, for me that should include plain text, Org mode, Markups and -downs, reST, calendar, email, Bash, Python, Fortran, C, C++, Qt, Makefiles, git, ebuilds, PKGBUILDs, LaTeX, BibTeX, HTML, CSS, assorted Wikis, config files, and the odd man page, or JavaScript, Octave, MATLAB or PERL file.
4. Also, I want to be able to read and write my email with it, and it would be nice if it could provide an interface to *e.g.* my calendar, to-do lists, git, GitHub, et cetera.[1]

[1] Not to mention the astounding capabilities of Org mode [5].

5. I want the editor to operate with a GUI, but also on a text console (*e.g.* for slow connections or low-spec systems).

6. The editor should be usable without a mouse (since grabbing a mouse, finding the pointer and clicking through a menu is *so much* slower than typing a key combination or a word).

The two best known editors that fulfil these requirements are vi [6] and Emacs [1] (see [7] for a comparison). Both are free-and-open-source software (FOSS), which makes wishes 1 and 2 more likely to be met. Also, they have been around since the (beginning) of the 1970's, which makes it more likely that they will be around for *another* 50 years. Finally, Emacs fulfils wish 5, while the vi's most often used clone *vim* has a graphical version called *gvim*. While vi is available on even the most basic Linux system (*e.g.* one of the 32 explicitly installed packages on a bare Arch Linux ARM installation; note that this is vi, not the improved vim), Emacs can typically be installed in seconds (*e.g.* on the same Arch system).

I eventually chose for Emacs over vi, because it seemed to be slightly friendlier. While gvim has a menu that can be accessed with a mouse, vim does not. In contrast, Emacs has such a menu in GUI mode and in text mode. Another pro of Emacs is that its key combinations are often identical to those in the Bash shell (see [8]). Finally, in my experience, while you can do a number of important things with simple keystrokes in vi, you can do many things with relatively simple keystrokes in emacs, many more in more complex keystrokes and everything by typing (and tab-completing) the command.

2 Emacs basics

You can open a text file by specifying its name as an argument:

```
$ emacs hello.c
```

If a graphical version of emacs and graphical desktop are available, emacs will open in a new window and you may want to append an ampersand (&) in order to start it in the background. You can force the terminal version of emacs (in the foreground) with the option `-nw`.

A default emacs screen consists, from top to bottom, of:

1. The menu (if shown);
2. The main text-edit panel (called *buffer*);
3. A status line;
4. A *minibuffer*, where commands show up (usually one line).

The menu can be accessed (and exited) with F10, and navigated with the arrow keys and Enter. For most options, the keyboard shortcut is shown, so that you can quickly learn it. Once you know the shortcut, you'll probably never use the menu again. Note that only a small (but important) fraction of emacs's functionality is available through the menu!

The most important key strokes to know in order to start learning emacs are:

- F10 Access or exit the **menu**;
- C-/ **Undo** text edit;[2]
- C-g **Cancel** the current command (press more than once if needed);

[2]This was the most important keystroke for me when learning emacs; knowing this I could undo whatever stupid mistake I made.

- Esc Esc Esc **Cancel everything** and go back to edit mode;
- C-x C-c **Exit** emacs, prompting to save changed buffers.

Here, C-/ stands for Ctrl /, *i.e.* holding Ctrl whilst pressing /, then releasing both.[3] Similar combinations exist for Alt, which is written as M-.[4] Most commands have a (long) name as well as a shortcut, which can be typed using M-x, followed by the name of the command and Enter. Tab-completion works while typing the command. For example, *undo* can also be achieved by M-x `undo` Enter.

The undo function in emacs is different from its equivalent in many other programs. Consider for example the case where you type `a`, remove it, and then type `b`, then press `undo`, then type `c`. If you then start pressing `undo`, many programs will never recover `b` — that version of the document has been lost forever. Instead, when pressing C-/ repeatedly, emacs will "replay" in reverse order *exactly* what happened before, including undoing previous undo's. While perhaps unorthodox, this means that you will always be able to get back to any previous version by pressing `undo` (within the limits of your undo history).

3 Further keystrokes

Other often-used keyboard shortcuts include:

- C-x C-f **Open** (visit) a file in the current buffer;
- C-x C-w **Write** the current buffer to a new file;
- C-x C-s **Save** the current buffer to its file;
- C-x s **Save** all buffers to their files;

- ←/→ **Move** the point (cursor) one character — officially C-f/C-b;
- ↑/↓ **Move** the point one line — officially C-p/C-n;
- C-←/C-→ (or M-) **Jump** the point one word — officially M-f/M-b;
- C-↑/C-↓ **Jump** one paragraph;
- C-a **Jump** to the beginning of the **line**;
- C-e **Jump** to the end of the **line**;
- C-v/M-v **Jump** down/up one **screenfull**;
- M-</M-> **Jump** to the beginning/end of the **buffer**;
- C-x C-x **Jump** between the point and mark;

- C-l **Scroll** the buffer so that the current line is in the middle/top/bottom;

- C-t **Swap** two characters;

[3] For C-x C-c you can hold Ctrl while pressing X and then C.
[4] Pressing and releasing Esc has the same effect as pressing and holding Alt.

- `M-t` **Swap** two words;

- `C-SPACE` **Mark** the beginning of a region (selection; press twice to show the region);
- `M-w` **Copy** the region from the mark to the current point (cursor position);
- `C-w` **Cut** the region from the mark to the current point;
- `C-Backspace` **Cut** the previous word;
- `M-d` **Cut** from the point to the end of the word;
- `C-k` **Cut** from the point to the end of the line;
- `C-y` **Yank** (paste) the last copy/cut at the point;
- `M-y` **Yank** previous copy/cut at the point (after `C-y`);

- `C-s` **Search** forward (repeat for next match);
- `C-r` **Search** backward (repeat for previous match);
- `M-%` **Search and replace** (press `y`/`n` to do/do not replace, `q` to quit, `?` for more);

- `Tab` **Indent** messed-up code (for most programming languages);
- `C-x (` Start **recording a macro**;
- `C-x )` Finish **recording a macro**;
- `C-x e` **Execute a macro** — press `e` afterwards to repeat;

- `C-x 2` **Split** the screen vertically in two panels;
- `C-x 3` **Split** the screen horizontally in two panels;
- `C-x +` **Equalise** panels in size;
- `C-x o` **Jump** to another screen panel;
- `C-x 0` **Close** the current screen panel;
- `C-x 1` **Close** all other screen panels;

- `C-x b` **Move** to another **buffer** (create it if it doesn't exist);
- `C-x C-b` **Show** a list of **buffers**;
- `C-x ←`/`C-x →` **Move** to the previous/next **buffer**;
- `C-x k` **Kill** a buffer (the current by default);

- `C-h i` Documentation in **info** format; same as the `info` command in the shell;
- `C-h b` Show a list of **key bindings**;

- `C-h k` Show what a **keyboard shortcut** does;
- `C-h f` Get the documentation for a **function**;
- `M-x apropos` **Apropos**: search a command you only half remember;
- `C-h ?` More **help** options.

4 Configuration of Emacs

The configuration of your Emacs editor is stored in the *hidden* (note the leading dots) file or directory `.emacs.d/init.el` or `.emacs` in your home directory. An example configuration file can be found at [9] and [10].

5 References

[1] **Stallman, R.** *Emacs*. URL https://www.gnu.org/software/emacs/. Visited 2016-03-20.

[2] **Yegge, S.** *Effective Emacs*. URL https://sites.google.com/site/steveyegge2/effective-emacs. Visited 2017-11-12.

[3] **Röthlisberger, D.** *How to learn Emacs*. URL http://david.rothlis.net/emacs/howtolearn.html. Visited 2017-11-12.

[4] **GNU**. *GNU Emacs reference cards*. URL https://www.gnu.org/software/emacs/refcards/. Visited 2017-11-13.

[5] **Dominick, C.** *Org mode for Emacs*. URL https://orgmode.org. Visited 2022-04-08.

[6] **Moolenaar, B. *et al.*** *Vim*. URL http://www.vim.org. Visited 2016-03-20.

[7] **Wikipedia**. *List of text editors*. URL https://en.wikipedia.org/wiki/List_of_text_editors. Visited 2016-03-20.

[8] **van der Sluys, M.** *Efficient use of the Linux command line in the Bash shell*. URL http://pub.vandersluys.nl. Visited 2022-04-19.

[9] ——. *My terminal configuration*. URL https://github.com/MarcvdSluys/MyTerminalConfig. Visited 2022-04-19.

[10] ——. *Environment settings (bash, emacs, git)*. URL https://github.com/MarcvdSluys/han-ese-ops-env. Visited 2019-02-11.