Very brief introduction to Emacs

C. D. Buckingham
Aston University

October 9, 2017

GNU Emacs is a free and extremely powerful *text editor* from the Free Software Foundation. Text editors are those that work on and generate straight text with no hidden formatting commands, such as Notepad, the one supplied with Microsoft operating systems. Emacs is much more powerful and can be modifed to carry out specific tasks easily and quickly. This document briefly explains how to use Emacs.

1 Interpreting the way Emacs commands are written in this document

For all these commands, CTRL stands for the Control key, ESC stands for the ESCAPE key, and a hyphen linking keys means hold them both together. So `CTRL-x u` means press CONTROL then x, keeping the control key still pressed, then release them both before pressing u. CTRL-xs means hold down the CONTROL key then x, then release x before pressing s, keeping the CONTROL key held down throughout (the Emacs manual writes this as CTRL-x CTRL-s but the effect is the same).

2 Starting Emacs

- Open a shell window and type `emacs &` which will invoke Emacs in its own window but run it in the background, which means you can still use the shell window for other commands.
- List the directory that Emacs is running in by typing `CTRL-xf .  <RET>` which will show all the files in your directory. Move to a file name and then press the `ENTER` key. This will load the file into Emacs.
- Loading files is also easily done using `CTRL-xf <filename>`.

Files loaded for editing are placed in a buffer and it is possible to have many buffers open at any one time. You can also see more than one buffer if the frame is split into more than one "window", which is useful for comparing different files. In fact, Emacs is extremely powerful and can actually be used as a complete operating system. It lets you send email, surf the web, create directories, play games, you name it, all within the Emacs program: 'nuff said: it's the best. To find out more about it, see:

- The emacs manual at `http://www.gnu.org/software/emacs/manual/emacs.html`
- `http://www.gnu.org/` for the GNU home page, with information on the Free Software Foundation, its licensing policy, and links to all its software and documentation.
- Links to all GNU software at `http://directory.fsf.org/`

3 Fundamentals

Some commands use a command buffer, which is right at the bottom of the frame, under the file name. If the cursor disappears down there, read what it says and follow the instructions. Alternatively, type CTRL-g to cancel the command (see below).

Undo, CTRL-x u Undoes the previous command

Cancel, CTRL-g Cancels whatever command is in progress. Useful when you get stuck somewhere strange....like the command buffer by mistake. It also cancels the selection of a region (see below).

Read a file into Emacs, CTRL-xf Takes you to the command buffer and asks for the file name. You can press the tab key while writing the file name, and emacs will complete the name, as far as is possible without ambiguity.

View the current directory, CTRL-xf . Shows all files in the current directory and you can load a file into the buffer by moving the cursor onto it and pressing the RETURN key.

Save file, CTRL-xs Saves the buffer contents to the directory file.

Leave emacs, CTRL-xc Will take you to the command buffer if any buffers have not been saved.

Go to command buffer, ESC x Takes you to the command buffer and waits for an Emacs command. If you get stuck or don't want to provide an Emacs command, cancel it (see above).

Getting help The File, Edit, Options, Buffers, Tools, and Help menu bar at the top of the frame will provide information if you click on them.

4 Moving around files

Cursor movement The following keyboard commands move the cursor:

up one line `CTRL-p`

down one line `CTRL-n`

back one character `CTRL-b`

forward one character `CTRL-f`

to the beginning of the line `CTRL-a`

to the end of the line `CTRL-e`

back one page `ESC v`

forward one page `CTRL-v`

beginning of document `ESC <`

end of document `ESC >`

Searching text Backwards and forwards:

search forward CTRL-s

repeat the same search forward keep pressing CTRL-s

search back CTRL-r

repeat the same search back keep pressing CTRL-r

5 Opening several files at once

windows You can have more than one window open within a frame, where the frame is split between two files. These commands manage the windows and files. Note that "o" means the letter o and "zero" means the number 0.

split the same file into two windows CTRL-x 2

move between windows CTRL-x o

close current window CTRL-x zero

open a file in a new window CTRL-x 4 f

Revert to a single window, the one with the cursor in it CTRL-x 1

6 Cutting and pasting text

To cut and paste text, you need to mark it first, then cut or copy it, then move to where you want to place it, and paste it in. The commands are as follows:

6.1 Marking text

1. Go to the start of the text and press CTRL-space
2. Move to the end of the text and press ESC-w

6.2 Cutting text

1. If you want to cut marked text, then press CTRL-w instead of ESC-w after marking it

6.3 Pasting text

All text that has been cut is put on something Emacs calls the "kill ring". You can paste any member of this kill ring by the following command:

1. CTRL-y brings back the last piece of text cut
2. ESC-y that immediately follows a CTRL-y will bring back the second to last piece of text cut.
3. Subsequent ESC-y commands will go further back on the kill ring, pasting in the associated text until there are no more left.

The "y" stands for "yank": that is, you yank back what has previously been cut. Note that ESC-y must follow a CTRL-y otherwise you will get an error in the command buffer. You may find that the amount of text the paste commands bring back and paste into your document causes you to be disorientated and confused between what is original and what was pasted. If so, you can get back to the previous state by the undo command, CTRL-x u.

7 Lisp and Emacs

The reason for using Emacs is that it makes Lisp programming much more easy. You need to ensure that your Lisp programs always have the `.lisp` extension so that Emacs goes into Lisp mode. You know the mode because it is on the information bar at the bottom of the Lisp window: the file name will be followed by the mode in brackets, which should be `(lisp)`.

You can use the Lisp mode of Emacs to ensure that your functions have the correct structure and number of brackets. When you type a closing bracket, the Emacs cursor will jump to the bracket it matches, so to complete a function definition, for example, you keep typing brackets until it matches the one before the `defun` command.

The most useful function of Emacs is to display the structure of your functions. If you press the tab key on a line of Lisp, it will put that line in the correct hierarchical position compared to the lines of lisp above. For example, if the following code was displayed as shown:

```
(defun get-room (room house)
(assoc room
(second
(get-rooms house))))
```

it is very difficult to see how the different parts of the function relate to each other. We need to see easily that the `second` function call is part of the `assoc` function and that the `get-rooms` function call is part of `second` and so on. Emacs will automatically put the cursor in the right place after carriage returns when you type in the function. However, if you do get the alignment wrong, then simply putting the cursor on the line next to the code and pressing the TAB key will cause Emacs to line the code up. Often this is enough to show where there are errors in your Lisp.

This method will cause the above code to be laid out as follows:

```
(defun get-room (room house)
  (assoc room
        (second
         (get-rooms house))))
```

and you can easily see the relationship between the different elements. It has lined up the two arguments for `assoc`, which are `room` and `second`. It has also indented the argument for `get-rooms` within `second`.

7.1 Emacs interprets Lisp

Emacs is written in Lisp and can interpret any lines of Lisp directly within itself. This can be useful to check whether your Lisp code has been correctly written without even having to load the program into the Clisp interpreter. For example, if the following line was written in any file (doesn't have to be a lisp file) within Emacs:

- `(* 2 4 (/ 4 2))`

and you type CTRL-xe at the end of the line, Emacs will run the Lisp code and show the result in the command buffer at the bottom of the window. If the Lisp has an error in it, then the error message will show up in the command buffer.