Emacs

November 17, 2017

Contents

1 The Basics **2**
- 1.1 Opening Emacs 2
- 1.2 Terminology 2
  - 1.2.1 M-x All the things! 2
  - 1.2.2 Regions, Buffers, Windows, and Frames, oh my! 3
  - 1.2.3 Killing and Yanking! 3
- 1.3 Commands 4
  - 1.3.1 Quitting and Canceling 4
  - 1.3.2 Navigation 4
  - 1.3.3 Opening, Saving, and Writing 5
  - 1.3.4 Marking 5
  - 1.3.5 Cut, Copy, and Paste 5
  - 1.3.6 Searching 6
  - 1.3.7 Undo and Repeat 7

2 Buffering **8**
- 2.1 Basic Buffer Commands 8
- 2.2 Viewing Multiple Buffers 8
- 2.3 Major and Minor Modes 9

3 Customization **10**
- 3.1 Easy Customization 10
- 3.2 Customizing .emacs 10
- 3.3 Packages 11
- 3.4 Themes 11

4 More **13**
- 4.1 Help 13
- 4.2 Tools 13
- 4.3 Lisp 14
- 4.4 Games 14

5 Further Reading **15**

Chapter 1

The Basics

GNU Emacs is an "extensible, customizable, self-documenting, real-time display editor" based on the Lisp programming language. A powerful text editor, Emacs allows users to edit multiple files at the same time and offers several neat features.

1.1 Opening Emacs

Emacs can be opened in several different ways. The most common way in a graphical environment is simply clicking the Emacs icon. Alternatively, you can launch it from the command line by issuing the command **emacs**. If you would like to run Emacs in a terminal, you can add the argument **-nw** which means "no window". It is also possible to include the file name you wish to open on the command line as well.

1.2 Terminology

1.2.1 M-x All the things!

When Emacs was invented there were entire computers dedicated to running Lisp. The keyboard that this computer used was known as the Space Cadet keyboard, which had four modifier keys: Control, Shift, Meta, and Super. While all of these keys continue on today, we don't necessarily refer to them by that name anymore. Control and Shift kept their names, but Meta is more commonly known as Alt on PC keyboards or Option on Apple keyboards. Super would go on to become the Windows key or the Command key on Apple keyboards. Frequently you will see Emacs commands referred to by key sequence, such as C-x C-s. This means holding the control key and pressing the x key, then while holding the control key pressing the s key. Here's a few examples, followed by their command name, and how to type them:

```
C-x C-c save-buffers-kill-terminal
```

Hold control and press c, followed by hold control and press c.

```
M-x execute-extended-command
```

Hold alt and press x. `M-x` is special because it will expect you to type the name of a command afterwards. Something to note is that *some* window managers and terminals will “steal” the Alt key for their own use. In this case you can press and release the Escape key followed by the next key in the sequence. Emacs will realize what you meant and interpret the Escape as the Alt-key being pressed. For the example above this would equate to hitting Escape, then x.

```
C-S-backspace kill-whole-line
```

Hold control and shift, then press backspace.

1.2.2 Regions, Buffers, Windows, and Frames, oh my!

Emacs comes from a time when there was no standard terms in place to describe many actions and environments we use today. Because of this, the creators decided on their own terms to describe everything. Starting from the top to the bottom, we have the following:

- Frame — This is the main visual window that displays everything. This is what gets the title bar on top with the minimize, maximize, and close buttons.
- Window — A frame has one or more windows inside of it. This is a view port to a buffer. Running emacs in a terminal means you still have one or more windows visible.
- Buffer — A buffer holds text.
- Region — A region is a selected block of text.

Many commands will operate on one of the above items. For example the command `other-window` moves the cursor to the next window.

1.2.3 Killing and Yanking!

We’ll go into detail on how to perform these actions later, but for now you should know that killing text is simply cut/copy, and yanking is pasting.

1.3 Commands

1.3.1 Quitting and Canceling

You can press C-x C-c to exit emacs. This will prompt you to save any open files or warn you if any buffers are still processing something. If you just want to exit emacs without any prompts you can press M-x and then type kill-emacs.

If you start entering a command and wish to cancel it, press C-g, multiple times if needed.

1.3.2 Navigation

Emacs provides many options for navigating your text files. While getting your feet wet, arrow keys and mouse cursor selection will work like other editors, but there are many more efficient shortcuts. By mastering these shortcuts you can improve your editing speed by reducing the amount of time your fingers are off of home row.

Instead of arrow keys, you can use C-f to move your cursor right, C-b to move your cursor left, C-p to move your cursor up, and C-n to move your cursor down. These can be easy to remember with the mnemonics **f**orward, **b**ack, **p**revious, and **n**ext. Additionally, the commands M-f and M-b will allow you to move your cursor an entire word at a time.

You can press C-e to move your cursor to the **e**nd of the current line. Likewise, you can press C-a to move your cursor to the start of the current line. These are both very useful commands while programming.

To move through your buffer quickly, you can use C-v to move forward a page and M-v to move back a page. The behavior is similar to scrolling with the page up and page down keys, but much easier to reach. To jump to the very beginning of the buffer you can press M-<. Likewise, to jump to the end of the buffer press M->.

While moving your cursor around, you may find that your window is not providing a good view of the surrounding text. To center your view around your cursor, press C-l. If you press C-l a second time, your cursor will now be at the top of the window. A third press of C-l will, you guessed it, move your cursor to the bottom of the window. Subsequent presses of C-l will cycle through these three views.

If you wish to go to a specific line, press M-g then g followed by the line number you wish to go to and finally enter to finalize your selection. All of these new shortcuts may seem inconceivable, but stick with them and they will quickly become muscle memory.

1.3.3 Opening, Saving, and Writing

The following commands all take a sequence of key presses and can be challenging to remember when starting out. Fortunately, all of the following commands start by pressing C-x . When you press C-x C-f you will be prompted to open a file. The starting directly will be relative to either the current working directory when emacs was started, or the same directory that the currently selected window was opened from. While typing the name of the file to open, pressing tab will auto-complete your selection similar to most shell interfaces. You are not required to enter the name of a file that exists, if you don't then the buffer will be created with the provided file name.

Saving and writing files is a straightforward process in comparison. Pressing C-x C-s will **s**ave the current window. If you opened a file that did not exist, it will be created at this point and saved with the provided file name. If you press C-x C-w you will get a prompt to define the file name the buffer will be **w**ritten as.

1.3.4 Marking

Marking is synonymous to selecting in emacs. One of the most common uses for marking is copying or cutting a region of text. You can click and drag a selection with your mouse if you are in the graphical environment, but getting used to the keyboard shortcuts will enhance your productivity. The most common way to mark a region of text is to press C-space to start your mark point and then use the standard navigation keys to grow your marked area. Likewise, you can press C-@ to start your mark point if you prefer. You may cancel marking text by pressing C-g . You can swap where the cursor is (The cursor is known as point) with the mark by pressing C-x C-x . This will allow you to select prior text that you mave have missed when starting your selection.

1.3.5 Cut, Copy, and Paste

Not only is the key bindings for cut, copy, and paste different in emacs, but, as mentioned earlier, so is the terminology. Pressing C-w will kill the selected text. Killing text is equivalent to cutting, but it is saved into the kill ring instead of a just clipboard. Pressing M-w is the equivalent of copy, the selected text is copied to the kill ring and is not removed from the buffer.

To paste text: you can press C-y to yank the top item in your kill ring into your buffer. After yanking, you can press M-y to cycle through the previous text that has been saved into the kill ring. This is especially useful when you are pasting text into multiple locations and accidentally kill a section of text into the top location of the kill ring.

Additionally, you can kill to the end of a line by pressing C-k . Pressing M-k will kill until the end of a sentence. Furthermore, C-M-k will kill to the end of the word. Finally, to kill an entire line, you can press C-S-backspace .

1.3.6 Searching

Emacs allows you to use incremental search, non-incremental search, regex search, and more. Pressing C-s will start an incremental search. Incremental search will move to the first instance that matches what you are typing as you type it. Searching starts at your cursor location, so if you wish to search an entire document it is best to move your cursor to the start of the file. Pressing C-s again will search for the next matching instance. You can use C-r to search backwards to the previous matching instance. Finally, press enter when you are done searching.

For non-incremental search, you simply press enter after pressing C-s. With non-incremental search, you specify a string before emacs will search for it. To perform a regex search, press C-M-s. You can now type in your favorite regex expressions and use C-s and C-r to traverse the results. Emacs actually remembers the searches you've run recently; when you start searching press M-p or M-n to see the recent searches.

To perform a search and replace, the key sequence is M-%. This is an odd key sequence, because in order to type a % you'll need to press Shift. So in reality the key sequence you press on your keyboard is Alt, Shift, 5. Once executed it will ask you for a word to search for, press enter and it will ask for the new word to replace it with. Emacs will then highlight the next occurrence of the word and you can press y to replace it or n to move to the next occurrence. You can also do a search and replace with a regex as a search string. To access this, press C-M-%. But what if you get to a case where you need to do more than just find and replace a word? While running the query-replace command you can press C-r to enter recursive-edit mode. This will give you full control of your editor to fix whatever you need. When you're done fixing the issue, you can resume your query-replace with C-M-c, which means you pickup where you left off in the search and replace.

1.3.7 Undo and Repeat

The first habit you may find yourself having to break is pressing `C-z` to undo. If accidentally hit `C-z` and you are returned to your shell with emacs moved to a background process, you can enter the command **fg** to return the last process (emacs) back to the foreground. Note: emacs is highly configurable and you can define `C-z` to be mapped to undo if desired (see later sections).

To undo previous commands and input, you can press either `C-/`, `C-_`, or `C-x u`. Note that the action of undoing is saved into your undo history once you perform any edits. For more control over undo, you can mark a region of text to limit where to undo. Because all actions are recorded to the undo buffer, simply moving the cursor and repeating the undo command will start re-doing the previous undos. This is really un-intuitive when starting out. Don't worry, it never gets easier. You're not wrong. There are multiple emacs packages to try and address this, but none are included in base emacs.

Another useful tool in emacs is the ability to repeat actions. By pressing `C-x z` you will use whichever command was last activated, and subsequent presses of `z` will perform the last command additional times. If you know in advance how many times you wish to repeat a command, you can use `M-[0-9]`, the meta key combined with a digit, to specify how many times you want the next command repeated. For example, `M-5 C-n` will move your cursor 5 lines down.

You can also use `C-u` to repeat the next command four times. This can be chained with multiple presses, e.g. `C-u C-u` will perform the next command sixteen times. If you type a standard character after `C-u` it will repeat that character 4 times. If you type digits after `C-u` it will specify how many times you wish to repeat the next command. To repeat a digit a N times, press `C-u` followed by the digit N and then `C-u` and the digit you wish repeated N times.

Chapter 2

Buffering

2.1 Basic Buffer Commands

Buffers are containers for editing text or executing emacs lisp programs. You can imagine buffers as tabs in a modern web browser, they have their own content that you can easily switch between. Windows are the containers that actually display individual buffers. When you start up emacs, you are given multiple buffers including "*scratch*" and "*messages*". You will see the name of the currently active buffer in the bottom left corner of a window. The name of a buffer will be set based on the name of a file you open or create.

Like everything associated with emacs, there are multiple ways to navigate the active buffers. The simplest method is cycling through all of the active buffers by pressing `C-x Left`, pressing `C-x` and then the left arrow key, or `C-x Right`. These are shortcuts for the respective emacs commands `next-buffer` and `previous-buffer`.

To fetch a buffer by its name, you can press `C-x b` and then enter the name of the buffer you wish to open. Opening a buffer by name has tab auto-completion once you type enough characters to identify an open buffer. You can also specify a buffer that is not open to create a new buffer with the given name. To view a list of buffers (which are presented inside their own new buffer!) you can press `C-x C-b`. This will bring If you wish to get rid of a buffer, you can use the key sequence `C-x k Enter` to kill the active buffer. Before venturing into the other methods of selecting buffers, it is best to understand how to handle viewing multiple buffers at the same time.

2.2 Viewing Multiple Buffers

Each emacs window displays a single buffer, so to view multiple buffers simultaneously you will need to open multiple windows. There are two different ways to split an active window. You can split a window across the horizontal axis by pressing C-x 2 . Likewise, you can split a window across the vertical axis by pressing C-x 3 . Each window can view any of the open buffers, and they can even view the same buffer from multiple locations. This is useful when you are editing a large file and need to reference code from multiple sections.

Navigating multiple active windows. The standard method of selecting windows is the key sequence C-x o which cycles through all of the windows. You can also click on a buffer inside of a window to make it the active window. More conveniently, if you run the command windmove-default-keybindings then you can press S-Arrow Keys to move to the window in the direction associated with the arrow key that was pressed.

Once you know how to create emacs windows, you will quickly realize that you will want to know how to remove them as well. Pressing C-x 0 will close the current active window. If you want to close all windows but the active window you can press C-x 1 . To close a window and remove its open buffer, use the key sequence C-x 4 0 .

You can further adjust the size of your active windows. Pressing C-x ^ will increase the vertical size of the active window. The sequences C-x { and C-x } will adjust the horizontal size of the active window. Using C-x z and subsequent z 's to repeat the last command helps adjust the size of a window quickly.

2.3 Major and Minor Modes

Each buffer that is open has a single associated major mode and can have multiple minor modes. Major modes determine the appearance and behavior of certain commands. For example, the fundamental-mode provides no syntax highlighting or special features. Conversely, the python-mode will provide python relevant syntax highlighting, change the behavior of indenting and unindenting blocks of code, and allow the execution of the current python script within emacs. Typically major modes will be decided automatically based on file type. Due to the nature of major modes, you cannot have multiple active at the same time, but you can have many minor modes active on a given buffer.

Minor modes can provide additional features to your active buffer. There are many minor modes distributed with emacs and you can download countless more. Entering the command auto-fill-mode will enable automatic line wrapping as you type such that any text on a line that exceeds a 70 character length will be wrapped to the next line. A popular minor mode is flyspell-mode which will highlight misspelled words in your active buffer. The command auto-revert-mode will automatically revert a file if it has been edited externally. If you turn on hl-line-mode then the active line will be highlighted. Another useful minor mode is show-paren-mode which highlights matching parenthesis.

Chapter 3

Customization

3.1 Easy Customization

Enter the command `customize` to open the easy customization program. From here you can select different sections, denoted by underlined text, by either clicking on them or navigating your cursor to them and pressing enter. Individual settings will have an arrow to the left of them, their name, and then a description below them. Clicking on the arrow will reveal a section to edit the value of the setting, often a toggle or numeric input. Once a setting has been changed, you will still need to go up to the top of the current buffer to select "Apply" or Apply and Save to activate the setting.

3.2 Customizing .emacs

When you customize values with the easy customization tool, the values are saved to the **.emacs** file in your home directory. In this file you can define emacs lisp commands to further customize your emacs experience. Lines starting with **;;** are treated as comments and are ignored. Any of the following commands can be copied into your .emacs file:

```
;; Enable line numbers for buffers that contain open files
(add-hook 'find-file-hook 'linum-mode)
;; Allows entering just y or n instead of yes or no
(fset 'yes-or-no-p 'y-or-n-p)
;; Indent each case in a switch statement in c-mode
(c-set-offset 'case-label '+)
```

Additionally, you can enable minor modes automatically in your .emacs file as follows:

```
(windmove-default-keybindings)
(add-hook 'text-mode-hook 'flyspell-mode)
```

```
(add-hook 'prog-mode-hook 'flyspell-prog-mode)
(global-hl-line-mode t)
(show-paren-mode t)
(global-auto-revert-mode t)
```

3.3 Packages

If you think that emacs didn't have enough customization, then we have the thing for you: packages! With packages you can easily download plugins for emacs to add even more features and change its behavior. Before installing new packages, you will need to specify where you are downloading packages from. The following lines can be added to your .emacs file to do this:

```
(require 'package)
(add-to-list 'package-archives
   '("melpa" . "http://melpa.org/packages/"))
(add-to-list 'package-archives
   '("gnu" . "http://elpa.gnu.org/packages/"))
(package-initialize)
```

Now you can enter the command `list-packages` to get a list of all available packages to install from the melpa and gnu databases. You can either click on the package's name and then choose install, or enter the command `package-install` and then type the name of the package you wish to install.

There are many packages that can greatly enhance your productivity and give emacs the functionality of many popular IDEs. The first of which is the `flycheck` package which can be configured to highlight syntax errors in real time as you are editing your code. You can also install `auto-complete` to get autocomplete suggestions. The following can be added to your .emacs file to automatically enable flycheck and autocomplete:

```
(add-hook 'after-init-hook #'global-flycheck-mode)
(require 'auto-complete)
(global-auto-complete-mode t)
```

3.4 Themes

Emacs has a variety of themes built in. You can select them with the command `customize-themes`. Likewise, if you know which theme you wish to apply you can use the command `load-theme` and then enter the name of the theme.

You can download additional custom themes, many of which are in the melpa archives. If the theme is available on melpa, installation is as simple as installing the associated package. Otherwise, you can download the themes into a folder and specify the path with the following line in your .emacs file:

```
(add-to-list 'custom-theme-load-path "path/to/themes")
```

Make sure you replace `path/to/themes` with the actual path to the directory with your custom themes. When you first try `load-theme` with a custom theme emacs will prompt you if you trust the theme as themes can execute their own lisp programs.

Once you find a theme you like, you might find out that not all of the colors are supported in the terminal. To work around this, you can use the following as a template in your .emacs file to load a different theme based on if your emacs is being run with GUI or terminal mode:

```
(if (window-system)
    (load-theme 'solarized-dark t)
    (load-theme 'afternoon t))
```

In the above example the theme ”solarized-dark” will be used in a GUI emacs session while the theme ”afternoon” will be used in a terminal emacs session. This assumes that the associated themes are installed.

Chapter 4

More

4.1 Help

The built in help system is a great resource to further your understanding of emacs. You can open up the manual by pressing the sequence: C-h r . Emacs also has a built in tutorial which is accessed with C-h t . By pressing C-h b you can see all of the current key bindings. If there is a command that you wish to know which key bindings are associated with it you can use C-h w and then type the name of the command, for example typing undo will list all of the keybinding associated with undo. If you are curious about what command is bound to a specific key press, enter C-h c and then press the associated key combination. Another useful help command is C-h m to list the major and minor modes that are currently active.

4.2 Tools

Emacs provides many tools that can further enhance your productivity. For example, you can run an interactive shell within a window by entering shell or eshell (eshell will provide additional features such as using arrow keys to select previous commands input into the shell). You can also compile your programs from within emacs with compile .

You can also debug your programs from within emacs by using the command gdb . You can use grep from within emacs by running the command grep or lgrep to run it asynchronously. You can kill all grep sub-processes currently running with kill-grep .

You can even view man pages from within emacs by invoking the command man . If you need to do a simple calculation you can use the emacs calculator . Another useful tool is ediff which allows you to compare the difference between two files.

4.3 Lisp

Emacs is a lisp interpreter, so anything that you can write in lisp can be executed within emacs. Emacs has its own flavor of lisp that is similar to MacLisp. For an introduction to programming with emacs lisp, goto:

```
https://www.gnu.org/software/emacs/manual/eintr.html
```

4.4 Games

If you ever need a break from being productive, emacs has you covered with built in games! You can never go wrong with a game of `tetris`. The classic game of `snake` is another fine choice. If you are feeling like a bit of a puzzle you can always try out the text based adventure `dunnet`. There's lots of games available to download for free as well. For a list of even more games check out:

```
https://www.emacswiki.org/emacs/CategoryGames
```

Chapter 5

Further Reading

Reference sheet (Very handy when printed)

```
www.gnu.org/software/emacs/refcards/pdf/refcard.pdf
```