Introduction to the Emacs text editor

Skylar; Jeffrey Fisher

March 22, 2023

Outline

What is Emacs?

What is Emacs?

Emacs is:

- a text editor
- customizable
    - There are many settings available without writing actual code, and small changes require little code.
- infinitely extensible
    - Thousands of plugins/extensions ("packages").
    - Emacs is configured with a full programming language, can easily add your own functions.

Emacs has a long history, but nowadays the most widely used version is GNU Emacs.

Dispelling the CMSC216 myth

If you attend the University of Maryland and take CMSC216, you will use Emacs.

But, it will be running on a (somewhat slow) shared server in a remote-desktop-like scenario.

Running graphical programs over a network is slow and unresponsive. Emacs is comfortably fast when it is running on your computer.

Also the version of Emacs installed on those servers is old.

About this presentation

Emacs is infinitely customizable. We can only cover a very small part of that infinity, and that part is biased by our experiences.

At the end of the day this presentation is really about how we use Emacs and how we think you *might* use Emacs.

What to expect from this presentation

- Hopefully, with the base config and knowledge from this presentation you can comfortably use Emacs at a basic level to edit code and other text.
- Learn how to get help from Emacs and external sources.
- Basic knowledge of a few advanced Emacs features.
- Awareness of interesting options to dive deeper, if you wish to do so.

Basic Emacs usage

First things first: Keyboard shortcut notation

`Ctrl+Alt+Shift+x`

When talking about keyboard shortcuts, Emacs would write the above as `C-M-S-x`.

- 'C' stands for control/ctrl.
- 'M' stands for "meta". For historical reasons Emacs talks about a

"meta" key. Nowadays this usually means "Alt".

- 'S' stands for shift.

First things first: Stopping/quitting

`C-g` key command quits the current action.

- Make a mistake while typing a key command? `C-g` will quit the partially entered command.
- Run a command that takes a while and want to stop it? `C-g`
- Is there a prompt open that you want to close? `C-g`

Note: `C-g` means `Control+g`, which means hold control/ctrl and press 'g'. We will use the shorter form because it is what Emacs uses, to help you get accustomed to it.

First things first: Running commands

`M-x` (that's hold Alt+x) lets you execute commands by their name.

Everything action in Emacs executes a command, even when you press simple letter keys. And you can always run that command manually by using `M-x`.

Mess around in a buffer

Create a new buffer.

1. `C-x b`
    - This means `Control+x b`, which means:
        1.1 Hold the Control/Ctrl key and press 'x'.
        1.2 Release the control key.
        1.3 Press 'b'.
    - Note: This is different from `C-x C-b`, which means hold control, press x then b, then release control.
2. At the bottom of your Emacs window there should be a prompt "Switch to buffer", with your cursor at the end so that you can type.
3. Type a name for the buffer, such as "new", then press enter.

You should now be in a blank buffer.

Notepad-level stuff works. Type stuff, backspace, arrow keys, selecting text and moving with the mouse.

Hold up

- Buffer = A place where text is stored that you can edit.
  - When you open a file it is loaded into a buffer. When you "save a file", the contents of the buffer are written to the file.

Learning new shortcuts takes time. The menu bar at the top of the screen has many common commands, and will also tell you the keyboard shortcut.

Opening files

So many options!

- You can click on the toolbar to open files. File > Open File...
- If you set Emacs as your default editor, then you can open files in Emacs from your operating system's file manager / file explorer.
- Open a file browser in Emacs: `M-x dired`.
  - Click on a file or folder name to open it, and click on `..` to go back 1 directory.
  - You can do other things like copy/move/rename/delete files in Dired too
- If you know the path of the file, type `C-x C-f`. You can hold Control while pressing x and f. Then you can type the path and open it.

Common shortcuts

- Select text: `C-SPC` (Ctrl+spacebar)
  - Using movement shortcuts will select more text. Press `C-g` to stop selecting.
- `C-z`, `C-x`, `C-c`, `C-v` for undo/cut/copy/paste
- Ctrl + left/right arrow to move by word
- Right click for context menu
- Save file: `C-x C-s`
- Close file: `C-x k`, `M-x kill-buffer`
- Save as: `C-x C-w`, `M-x write-file`
- Find / search file contents: `C-s`
- Go to start/end of file: `M-<`, `M->`

Viewing multiple buffers at the same time

1. Split the window: `C-x 2` splits horizontally, `C-x 3` splits vertically
2. Scroll down one window. The view changes in that window only.
3. Use `C-x b` to switch buffers. You can also left/right click on the buffer name to cycle buffers, or use the menu bar.

Window shortcuts

- Meta (Alt) plus arrow keys: move between windows
- Meta and Shift plus arrow keys: swap the position of windows
- `C-x 2`, `C-x 3` = Split horizontally and vertically
- `C-x 0` = Delete/close window

Beginning our configuration

Emacs Lisp code

```
;; Initialize package manager
(require 'package)
(setq package-archives
      '(("gnu" . "https://elpa.gnu.org/packages/")
        ("nongnu" . "https://elpa.nongnu.org/nongnu/")
        ("melpa" . "https://melpa.org/packages/")
        ))
(package-initialize)
(unless package-archive-contents
  (package-refresh-contents))
```

- semicolon (;) = Comment that continues until the end of the line.
- 'package = A "symbol". Human-readable constant. Can think of it like an enum value.

Emacs Lisp code

```
;; Initialize package manager
(require 'package)
(setq package-archives
      '(("gnu" . "https://elpa.gnu.org/packages/")
        ("nongnu" . "https://elpa.nongnu.org/nongnu/")
        ("melpa" . "https://melpa.org/packages/")
        ))
(package-initialize)
(unless package-archive-contents
  (package-refresh-contents))
```

Translated to familiar syntax:

```
import package

# Initialize package manager
package.archives = [["gnu", "https://elpa.gnu.org/packages/"],
                    ["nongnu", "https://elpa.nongnu.org/nongnu/"]]
package.initialize()
if (not package.archive_contents) {
   package.refresh_contents()
}
```

Emacs Lisp code

In a C-like language:

```
sqrt(1 + 2 + 3)
```

In Emacs Lisp:

```
(sqrt (+ 1 2 3))
```

- Move the function name inside the parentheses.
- All operators (like +) are called using function syntax, so the plus goes at the start (prefix) instead of in the middle (infix).

Setup the package manager

Emacs has many features built-in, but we will want some third-party packages. Here, "packages" just means additional code for Emacs.

You can generally think of them like plugins/extensions.

```
;; Initialize package manager
(require 'package)
(setq package-archives
      '(("gnu" . "https://elpa.gnu.org/packages/")
        ("nongnu" . "https://elpa.nongnu.org/nongnu/")
        ("melpa" . "https://melpa.org/packages/")
        ))
(package-initialize)
(unless package-archive-contents
  (package-refresh-contents))
```

Package archives

The `package-archives` variable specifies where to download packages from.

- MELPA = Widely used third-party package repository.
- ELPA = Emacs Lisp Package Archive. This is the official Emacs package archive.

```
(setq package-archives
      '(("gnu" . "https://elpa.gnu.org/packages/")
        ("nongnu" . "https://elpa.nongnu.org/nongnu/")
        ("melpa" . "https://melpa.org/packages/")
        ))
```

use-package

We will be using use-package, a tool for declaratively specifying package configuration.

```
(unless (package-installed-p 'use-package)
  (package-install 'use-package))
(require 'use-package)
;; Download and install configured packages if they aren't already installed.
(setq use-package-always-ensure t)
```

Pretty colors

```
(use-package ef-themes)
;; A nice dark theme. 'modus-operandi' is the light theme version.
;; You can change the theme while Emacs is running with `M-x load-theme`.
(load-theme 'modus-vivendi)
```

- Selecting a theme with `M-x consult-theme` will interactively preview what the theme will look like.
- You may get a minibuffer prompt asking you to approve a theme. Themes can run arbitrary Lisp code, so for security only themes you have approved can be loaded.
- The `ef-themes` look nice and colorful. There are many more themes out there, you just have to find a package that has one you like.

Binding keys

You can always run a command with

Getting help

Self-documenting

Emacs is *self-documenting*. It can tell you information about itself. This feature is dynamic; if you rebind a key, or define your own function/variable, that info will also be shown.

What does that (variable|function|command|...) do?

Commands starting with `describe-`.

- `describe-command` (bound to `C-h x`. Mneumonic: `x` because `M-x` runs commands.)
- `describe-variable` (bound to `C-h v`)

`C-h ?` will tell you about all the help functions bound under the `C-h` prefix.

Advanced Emacs

Advanced Emacs

In this section, keep in mind that this is optional.

There are many Emacs users out there who don't use any third-party packages, don't do much customization, or don't use advanced text editing features.

Advanced text editing

Completions!

- Tab complete works as you expect
- `M-/` shows a window for completing by substrings (dabbrev)
    - complete words that are already in the buffer
- The cape package provides extra completions

Advanced Emacs: Macros

What are Emacs macros?

Basics

- `<f3>` = Start recording
- `<f4>` = Stop recording
- `C-x (`, `C-x )` = Start/stop recording
- `C-x e` = Execute last macro

(optional) Adventures

Adventures that await you, if you wish

- TRAMP : Transparent Remote (file) Access, Multiple Protocol
  - Similar to VS Code's Remote SSH plugin.
- evil-mode: Emulates Vim keybindings.
- org-mode
  - "keeping notes, authoring documents, computational notebooks, literate programming, maintaining to-do lists, planning projects", spreadsheets
  - This slideshow and the sample configuration we provide you were both created from the same Org document.
- Magit, the magical Git interface
  - A high-quality interface for the Git version control system.
- eww, the Emacs web browser

If you love Emacs so much, why don't you marry it?

Here are just a few examples of things Emacs can do beyond editing text.

- Shells / terminals: `M-x shell`, `eshell`, `term`, `ansi=term`
  - Running Shells and Terminal Emulators in Emacs
- Email: GNUS, mu4e, and more
- Instant messaging client for Slack, IRC, Matrix, etc.

(extra) Plain fun

Plain fun

- artist-mode lets you draw text-based art.
- `M-x strokes-help` : Control Emacs with mouse gestures.
- `M-x follow-mode` : Enable this and open multiple copies of a buffer side-by-side with `C-x 3` to read a buffer across multiple columns.

(extra) Other resources

Whet your appetite

- Emacs Rocks!: Series of short videos demonstrating cool and useful things you can do with Emacs.
- Batteries included with Emacs : Underrated built-in features.
    - More batteries included with Emacs

Reference

- GNU Emacs manuals
  - Also available inside of Emacs. `M-x info-emacs-manual` or `C-M-h r`
- EmacsWiki
- `https://www2.lib.uchicago.edu/keith/emacs/`

(extra) Glossary