

Introduction to the Emacs text editor

Jeffrey Fisher; Skylar

March 15, 2023

TODO

What is Emacs?

Basic Emacs usage

Beginning our configuration

Getting help

Advanced Emacs

(optional) Adventures

(extra) Plain fun

(extra) Other resources

(extra) Glossary

TODO

Before publishing this document, search for the keyword TODO, and make sure nothing shows up.

Configure a file tree sidebar

Many people like it, popular in other code editors.

Window management commands

The defaults are not very mnemonic.

Jeff: I need to look at windmove.

What basics do we need to cover?

Do we even need to cover "arrow keys move in the text"? I guess we could mention it.

Jeff: As a short introduction at a hackathon I feel we should introduce interesting features instead of the basics. The basics of movement are best learned with sustained practice, and Emacs has a good tutorial for that already. Plus common shortcuts actually seem to be default, like Control plus arrow keys for movement by word.

What to do for unfamiliar notation/terms?

At least two options:

1. Cover them upfront.
 - Pro: Helpful for using Emacs themselves.
2. Use familiar terminology, and mention that there is a glossary that they can look at. Put the glossary in the Git repository or in the slideshow.

Jeff: Temporarily I am leaning towards covering them upfront. Will wait for feedback from Skylar. Also will see how much time it uses and confusion it creates.

Display line numbers by default, or at least mention how to in the config.

- Emacs keyboard shortcut notation

Goal: Reasonable language-specific setups

We can't achieve perfect language-specific setups because it will depend on personal preference and there are a lot of languages.

Jeff: Currently I am not using that many popular languages. My personal config doesn't have much language-specific config to pull from. All I can do is recommend modes for syntax-highlighting and Tuareg mode for OCaml.

Jeff: If eglot LSP is reasonably easy to setup, then that'd be great. However I don't think installing the actual language servers easily is a solved problem. I haven't used LSP though so I don't actually know.

Goal: Configuration results in an Emacs usable at a Notepad++ level

Should paper over main annoyances that new users face.

cua-mode does a lot here.

Spellcheck this document.

Should we enable pixel-scroll-mode in the config?

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

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.

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'.
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.

- 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.

Emacs is definitely keyboard-focused, but it has pretty good mouse support.

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

Viewing multiple buffers at the same time

Beginning our configuration

Emacs Lisp code

```
;; Initialize package manager
(require 'package)
(setq package-archives
  (list ("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
  (list ("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()
}
```

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
  (list ("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
  (list ("gnu" . "https://elpa.gnu.org/packages/")
        ("nongnu" . "https://elpa.nongnu.org/nongnu/")
        ("melpa" . "https://melpa.org/packages/")
  ))
```

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.

Follow "Common User Access" conventions.

- C-z = Undo
- C-x = Cut
- C-c = Copy
- C-v = Paste

```
(use-package cua-base
  :custom
  (cua-keep-region-after-copy t)
  :init
  (cua-mode))
```

Getting help

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. Mnemonic: 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

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.

Useful features

(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.

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

- `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](#)

- [GNU Emacs manuals](#)
 - Also available inside of Emacs. `M-x info-emacs-manual` or `C-M-h r`
- [EmacsWiki](#)

(extra) Glossary

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.