Introduction to the Emacs text editor

Jeffrey Fisher; Skylar

March 10, 2023

Outline

TODO:

What is Emacs?

Beginning our configuration

(optional) Advanced Emacs

(extra) Other resources

(extra) Glossary

TODO:

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

TODO: Notation/terms glossary

- Emacs keyboard shortcut notation

TODO: Make links in the PDF slide output more obvious with underline and color.

What is Emacs?

What is Emacs?

Emacs is:

- a text editor
- customizable
    - There are many settings available.
- 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.

Beginning our configuration

Emacs Lisp code

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

Translated to familiar syntax:

```
import package

# Initialize package manager
package.archives = [["melpa", "https://melpa.org/packages/"],
                    ["elpa", "https://elpa.gnu.org/packages/"]]
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
      (list ("melpa" . "https://melpa.org/packages/")
            ("elpa" . "https://elpa.gnu.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 ("melpa" . "https://melpa.org/packages/")
            ("elpa" . "https://elpa.gnu.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 with ‘M-x load-theme‘.
;; You can pick from the ef-themes with ‘M-x ef-themes-select‘.
(load-theme ’modus-vivendi)
```

Conveniences

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))
```

(optional) Advanced Emacs

(optional) 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

Adventures that await you, if you wish

- 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) Other resources

Whet your appetite

- Emacs Rocks!: Series of short videos demonstrating cool and useful things you can do with Emacs.

Reference

- 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.