Emacs

Last Updated: Sept 2022

1. Defn

Emacs stands for Editor Macros, made in 1976. Highly Extensible, has features like debugger, terminal, music player etc. built-in as its basically an entire IDE. Unlike VSCode, a lot of setup is needed but it is much more configurable and has its own unique set of features including org-mode, a mode which is used to work with .org file format. Org-mode combined with org file format is a very comprehensive text editing solution, much more well-defined than markdown/HTML. This document itself should be in org-mode though idk how to use it effectively as of yet, which this doc aims to solve.

Emacs uses Lisp Emacs which is a modified version of Lisp language and hence almost all the IDE features can be easily modified/automated with Lisp code.

This document is for Emacs 27+ only, before Emacs 27 a lot of features were different.

This is more of keyboard-driven IDE, and it’s much faster to zip around using keybindings.

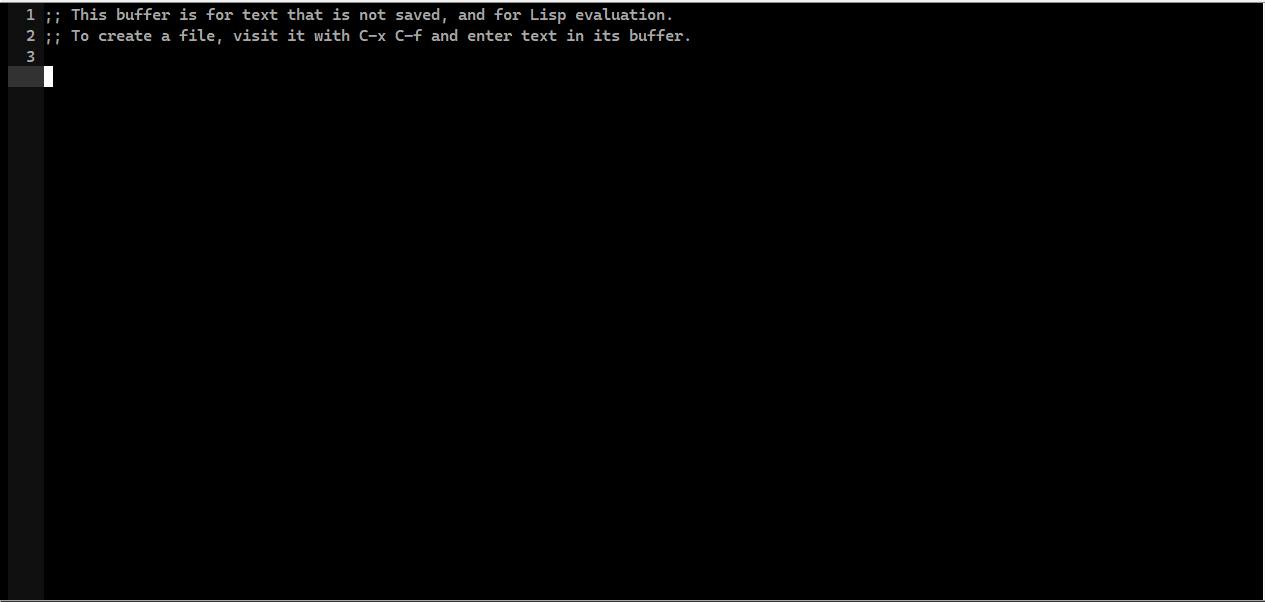
1. Packaging

Normal edition of emacs includes GUI which allows multiple fonts, colors etc. but the non-gui versions, emacs-no-x packages, have all the other features except these GUI features.

1. UI

Menu Bar: 

Tool Bar: 

Buffer: 

Mode Line: 

Shows the status of the buffer like buffer name, line number, file type etc.

Unlike the other bars, the mode line isn’t shared across windows and each window has its own mode line.

Echo Area/MiniBuffer: Just below status bar/mode line, it shows currently selected/executed commands, in which case it is called the echo area and allows running other commands/confirmations/options for commands, in which case it is called the minibuffer.

1. Emacs launch modes
   1. Terminal mode

Launched with emacs -Q -nw

In this mode we still get menu bar, mode line and MB by default and pressing f10 brings focus to the menu bar.

1. Cool functions/commands

We can write our own too but these are bundled together.

Describe-symbol: Describes a function/keybinding

Describe-bindings: Describes a keybinding

Describe-Key: Just like the function above.

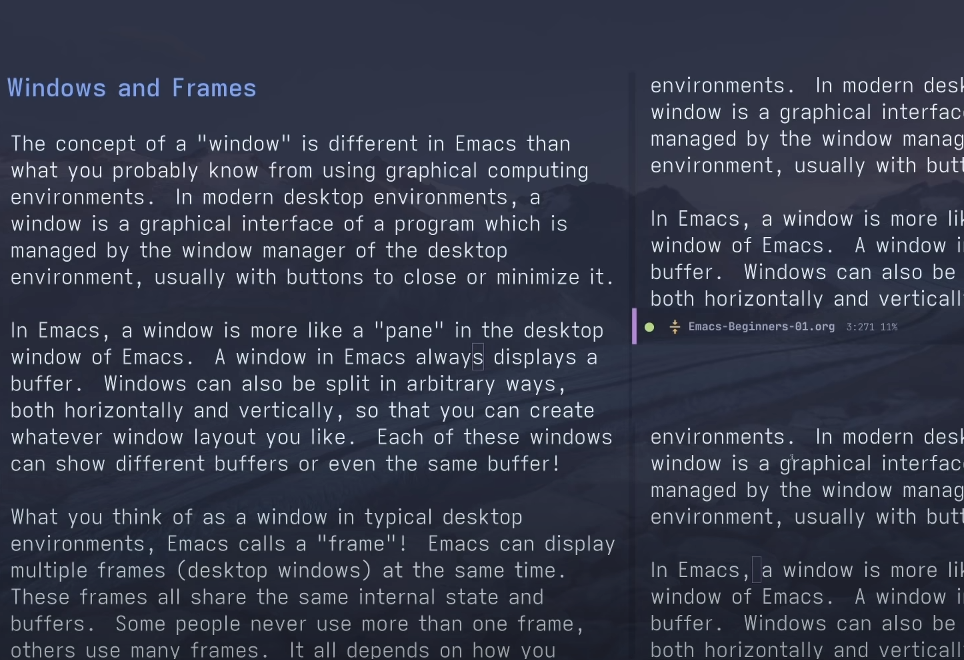
calendar

tetris

eshell : Run emacs shell, it is basically bash+ emacs lisp so we get best of both worlds.

1. Windows and Frames

A ‘window’ in emacs is a pane, it simply holds one or more frames, and each frame has exactly 1 buffer in it. A frame is the part that holds a buffer, where a buffer is an editable textpad.



Here we have 3 windows, each with 1 frame and 1 buffer.

Each frame can display the same buffer or different buffers. We can change what buffer is open in a frame too.

The use of multiple frames per window is that we can bundle related frames together and use them around.

1. Buffer

Contains data (usually text) and allows viewing/editing it. Every emacs window has at-least 1 frame and every frame has 1 buffer open.

Buffers aren’t just there as textpads, they can be be interactive like a terminal too and hold custom functionality.

There are 3 buffers that always open up by default

* 1. \*scratch\*

Temporary buffer that allows us to use it as temporary text storage.

* 1. \*Messages\*

Prints all emacs debug messages.

* 1. \*Warnings\*

Prints all warnings.

1. Major and Minor Mode

Every buffer has a single major mode, this defines what type of file the buffer is treating it as. Like .py files or .js files and so the major mode for them would be the Python mode or JS mode. These allow different types of syntax highlighting, code completion etc. By default, if no major mode is specified then the “fundamental mode” is activated for a buffer, which is a plain text editor major mode. We can always change the major mode for a buffer at any time.

A buffer can have multiple minor modes, these affect how a single line is formatted/highlighted. Like hl-line-mode, the highlight line mode, when enabled highlights current cursor line. There are global minor modes too, which enable a minor mode for all buffers.

1. Configuration

There are 2 ways to configure/customize emacs, the first is through the usage of customize command which presents a buffer with all the customization options and we can give them values.

The other is through the usage of an .el file.

Emacs automatically picks up a .el (emacs lisp) file at the start and then runs it to apply configuration each time it starts. We can define our file like so and emacs will customize itself with it. By convention, this file is named init.el.

However, the file must be at the HOME directory, which is determined through a few steps. But basically, this location will always work, ~/.emacs.d/

In windows, ‘~’ is %appdata%/Roaming/

or we can set a “HOME” environment variable and give it the location of anywhere, generally its C:/Users/%USERNAME/

So we can create an init.el file here.

Sample init file,

;;t is true, nil is false

(setq inhibit-startup-message t

visible-bell t)

(tool-bar-mode 1)

(global-display-line-numbers-mode 1)

(load-theme 'modus-vivendi t)

(set-frame-font "Cascadia Mono SemiBold 12" nil t)

(require 'cua-base)

(cua-mode t)

;;enables cua mode each time emacs opens

* 1. In doom emacs, doom/find-file-in-private-config shows the config files that are being used.

1. Keybindings

Over 1000 keybindings. All can be customized/removed and more can be added too.

Most of the important keybindings can be seen through the Menu items in the Menu bar, shortcut F10.

All keybindings are shortcut to run a function.

* 1. Modifiers

M: Alt key (called Meta key, as Alt didn’t exist back then)

C: Ctrl Key

S: Shift Key

s: Super Key (windows key)

RET: Enter Key

SPC: Space Key

for ex.

C-x is Ctrl+x

C-x c is Ctrl+x then leave both keys and press c

C-x C-c is hold Ctrl and press x then press c.

Some keybindings are known as prefix keybindings, like C-x and C-c. When pressed, emacs waits for us to define the rest of the keybinding. C-x is prefix for all of Emacs’ primary key bindings and C-c is prefix for chaning major modes. After pressing the prefix we can take our time to enter the rest of the keybinding.

* 1. Basic keybindings

F10: Open the Menu bar.

M-x: Brings focus to MB and allows typing a command. Press tab after something or nothing to get list of available commands.

C-x C-s: Saves a file

C-x C-f: Opens a file

C-x C-w: Saves a file into different location and then opens it.

C-x b: Switch buffer, requires buffer name to switch to which can be retrieved with tab key.

C-x C-b: List open buffers.

C-x <right/left arrow>: Quick switch to next/prev buffer.

C-SPC: Start selecting a region, or just hold shift and use arrow keys.

C-w: Cut a selected region. Called ‘kill region’ as the region is sent to a ‘kill ring’.

C-y: Paste killed region. Called yank.

Alternatively, enable CUA mode, which enables C-x, C-c and C-v for Cut/Copy/Paste. The cost of this mode is that if we select a region and use C-x or C-c then if we don’t want to cut/copy but instead use other keybinding like C-x c, we only have 0.2s to enter the combination or alternatively we can use C-S-x and then enter c normally. To enable it, either do it through Options > CUA Mode or execute the function cua-mode.

C-\_ or C-/: Undo key.

C-g C-\_: Redo. Undo/Redo in emacs work a bit differently as if we undo after redoing we will redo what we undo’ed. And then it will start undo’ing again after reaching redo history.

C-g: Just like C-c in windows terminal, it interrupts running commands.

C-h k: Runs the describe-key function.

C-x 2 Creates a horizontal split in the open window making an extra window

C-x 3 Same but a vertical split

C-x o Switch to next window

C-x 0 Close the current window (not the buffer)

C-x i Close all windows except the current one

C-x b Change the current window’s buffer after specifying a name for the other buffer, use tab to autocomplete names

C-x C-b Change the current window’s buffer but presents a selection

C-x left or right Quick change the buffer to next/previous one in the buffer list

C-x k kill the current buffer

* 1. M-x Tab

Tab after any command or even without any shows auto-completion, so the names of commands that are visible to emacs and also completes the command if we press tab on one selected command.