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.

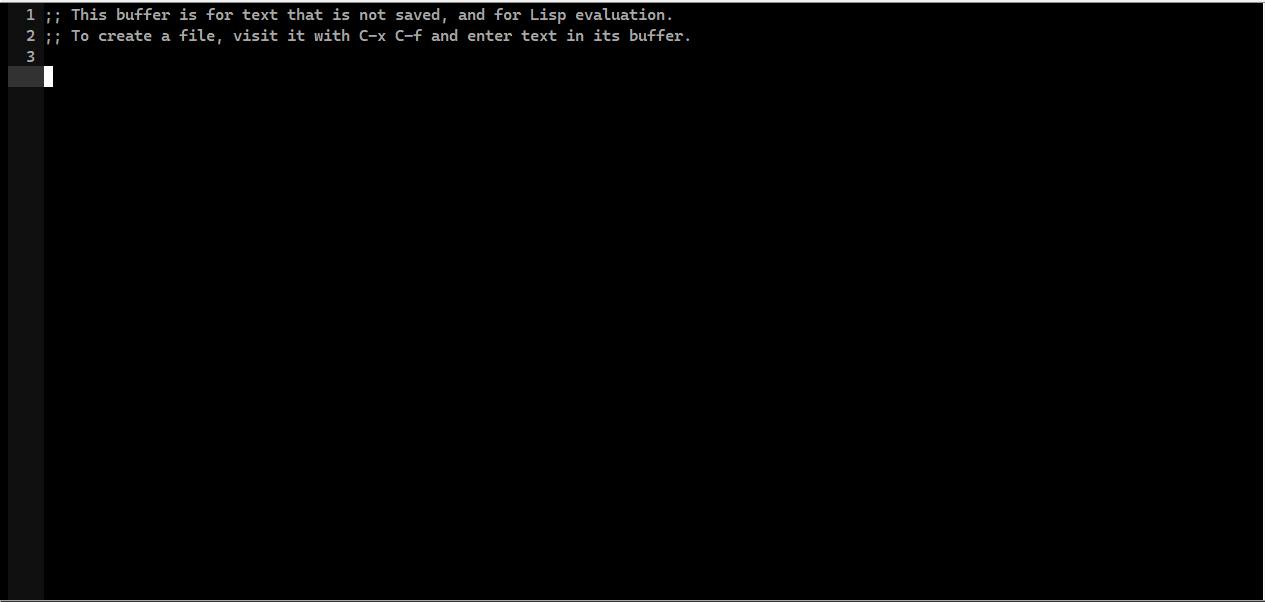
1. Packaging

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.

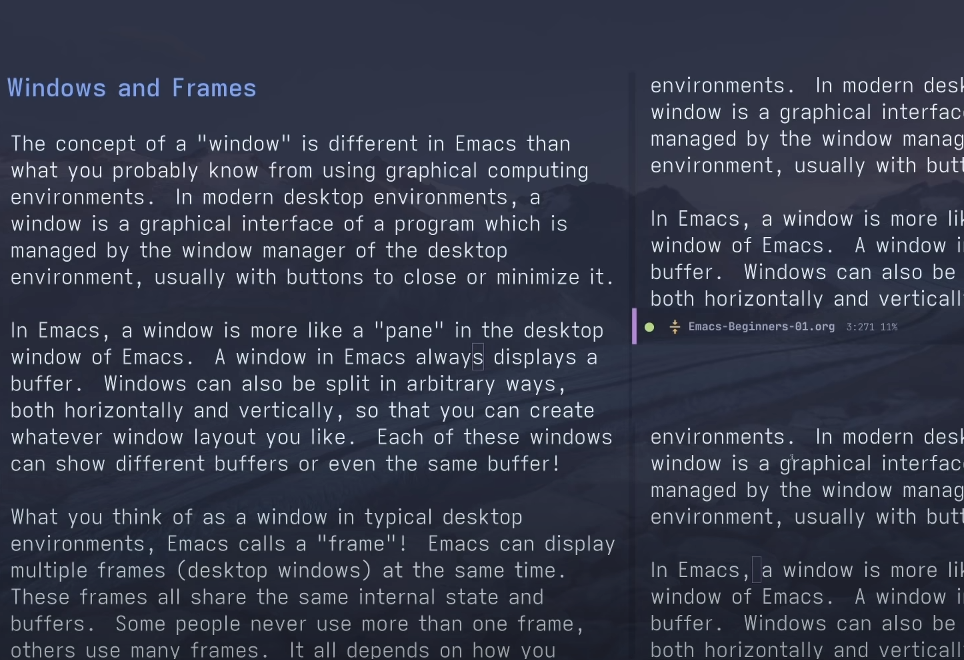
calendar

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.
  2. In doom emacs,

config.el

(global-visual-line-mode 0)

sets the characterwise selection mode for the replace mode (and not the insert mode).

By default this is set to 1 meaning when we select characters, it follows visual line rather than characterwise selection, which leads to the selection jumping to the next line when nearing the last few chars in the line. This disables that for replace mode.

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. The most important keybindings  
     :   
     which is basically Shift + ; key.  
     And  
     M-x  
     which is basically Alt + x key.  
       
     Pressed in overview mode ‘:’ brings the command-mode, which is where we can type any command. Most commands here are from Vim world, so they are there to provide easy functionality. Commands are basically functions, these are from the Vim world but they are adapted to emacs.  
       
     Similarly, M-x brings the emacs-command-mode where the commands/functions are made specifically for emacs and use full emacs’ functionality.   
       
     All other keybindings are a shortcut to subcommands of these commands.
  2. 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-x h Select whole buffer (Select All)

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

M-w: Copy a region. (Doom emacs)

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

In doom emacs, we can enable :ui workspaces in init.el, then we can use

SPC Tab n Create a blank new workspace

SPC Tab s Save a workspace to a file

SPC Tab l Load a workspace from a file

SPC Tab 1-9 Switch to the given int workspace

SPC Tab d delete a workspace

Workspaces allow saving and restoring frame layouts quickly.

* 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.

* 1. There are 2 primary modes in working with text files, one is the insert mode and the other is the replace mode. Keybindings such as save file work only in Replace mode and mean something different in Insert mode. The insert mode is where we can actually enter text in the text file. This is only true in evil mode in emacs.

Toggle between the 2 modes with the Insert key (Doom emacs).

Similarly there’s 2 more modes, active mode (which enters edit or insert mode in a file) or overview mode (opinion). keybindings such as SPC Tab … only work in overview mode.

When we switch to a buffer, or press any normal key then we enter active mode and can edit the file (as it is in replace mode and can go into insert mode) or do something with it using other keybindings. Pressing escape key pulls us out into the overview mode where we act on the buffer itself so keybindings are different there.

To check which mode is being used, check the color dot on the left side inside the status bar.

For ex.:

In doom emacs,



is the overview mode

and



is the active mode.

* + 1. Undo

When we undo in emacs, it undoes a step. Each ‘step’ is basically the changes in a file from the point we entered insert mode to right now, and each time we enter insert mode a new ‘step’ is created.

So if we don’t exit insert mode and keep editing text in it and undo, it will undo everything from the point we entered it.

Thus it is recommended to frequently toggle between insert and replace mode or insert and overview mode. Think of each time we exit insert mode as a checkpoint. And undo will bring us to the last checkpoint.

* 1. Searching for text (Find text) in current buffer  
     In the overview mode, pressing ‘/’ will begin searching for text, after writing the text press Enter then pressing ‘n’ will go to the next occurrence and pressing ‘S-n’ will go to the previous occurrence.  
     This is the Vim-way of searching for text in emacs.  
       
     The emacs way is using the `isearch-forward` command, which has a shortcut of `C-s` whilst in active mode. After typing text it is highlighted, pressing `C-s` or ` C-r ` will cycle searches, pressing RET will put cursor on the current item in the buffer and pressing Backspace twice will allow editing search text.
  2. Searching and replacing  
     Vim way: Go into Vim command mode with S-;, then type %s/<search text>/<to replace with text>/g  
     Here %s is the search command and g is the global tag.  
       
     Emacs way: We can use the `query-replace` function or the `replace-string` function.  
     query-replace can be invoked with `M-%` (Alt+Shift+5) then we enter search text, then replace text then for each occurence `y` to confirm replacing, `n` to skip, `!` to replace all and `q` to exit.  
     replace-string is in the Emacs command (M-x) and is a simpler version of query-replace.