Joining The Church Of Emacs

A summary of how I configured my Emacs

Published on 03 October 2022
Last updated on 03 October 2022

All your computing are belong to us

Emacs, which stands for "Eight Megabytes and Constantly Swapping" is a 46 year-old text editing program.

Everything about Emacs, from its UI to the action performed when you press the `Enter` key, is hackable. You are *encouraged* to extend the behavior of the Emacs program by writing functions in the programming language it speaks: **Emacs Lisp**.

In practice, this means you have over `2000` commands at your disposal, out of the box.

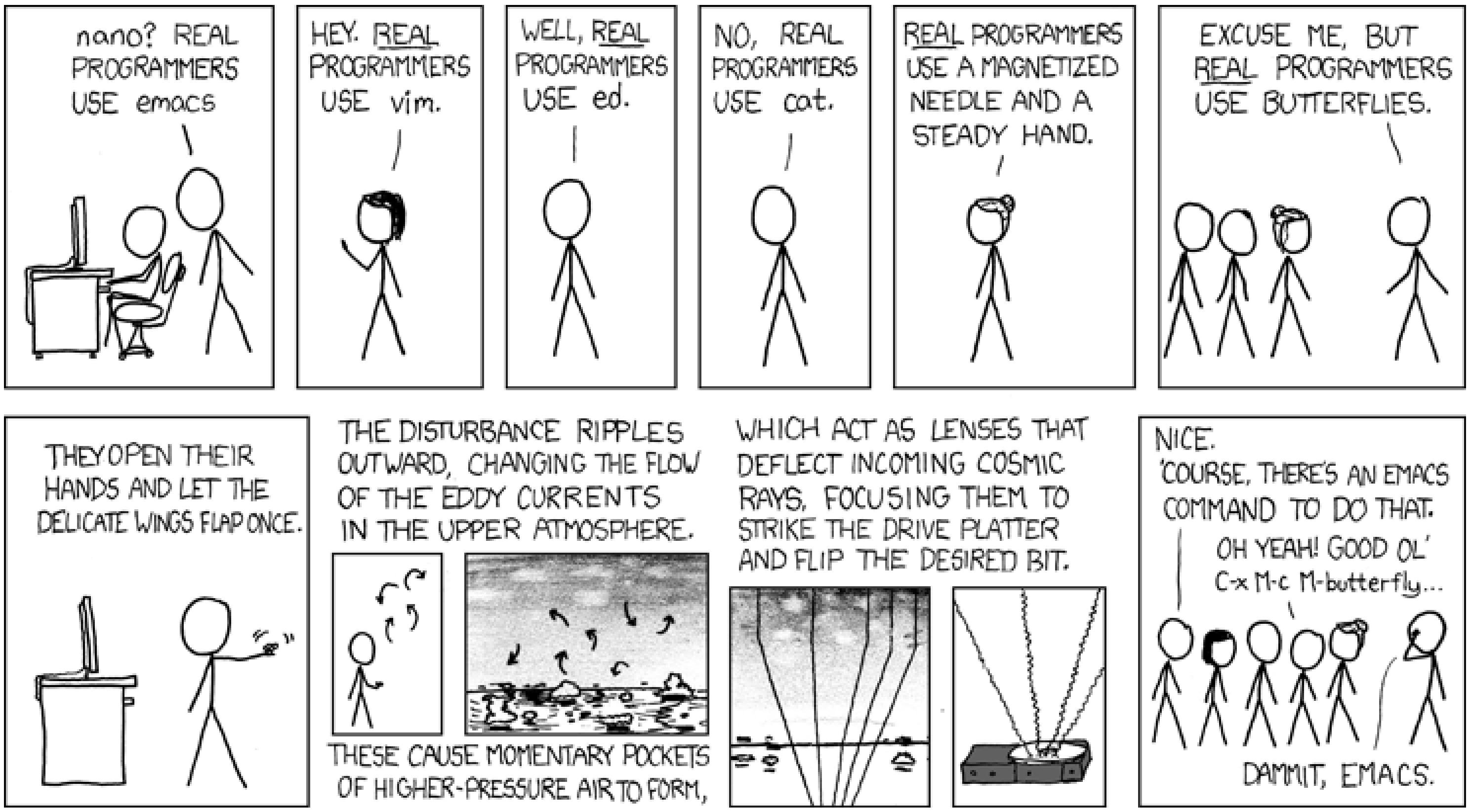
For example, the `align-regexp` command allows you to vertically align a region of text based on a regular expression rule. Say you have the following badly-formatted mess:

```
one = 1
fortyTwo = 42
elite = 1337
```

Calling `align-regexp` `=` after selecting the above snippet will yield:

```
one      = 1
fortyTwo = 42
elite    = 1337
```

This makes for a long-running Emacs joke; the idea that typing `M-x literally-anything` will result in exactly what you want is very amusing. In fact, while writing this article I got tired of typing out the markdown syntax for links, so I searched in the list of commands for "markdown link" and, lo and behold, `markdown-insert-link` exists and is even bound to `SPC c l` (more on this later). The following comic sums this up perfectly:


nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMACS.

Real Programmers, by xkcd

Ultimately, Emacs was built around empowering *end users*, not just developers and plugin authors, to do any kind of computing they want, without ever leaving Emacs[1].

> I'm using Linux. A library that Emacs uses to communicate with Intel hardware.
>
> — Erwin, #emacs, Freenode.

Emacs Lisp

At this point, I've been using Emacs for a little over two months. During that time, I haven't written much Emacs Lisp because I haven't needed to *program* Emacs myself; almost everything that I needed to do was available in the form of a package.

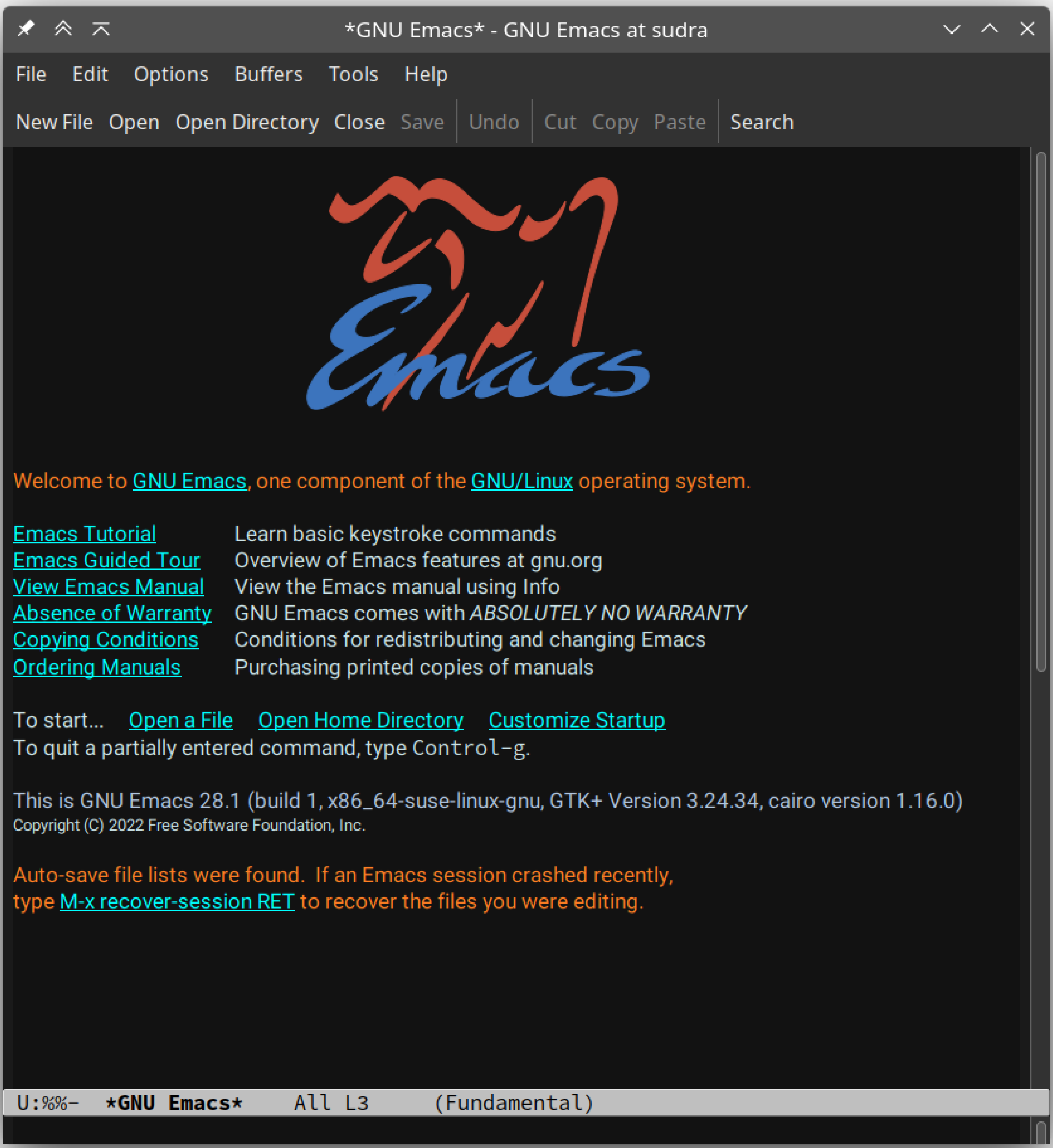
MELPA, Milkypostman’s Emacs Lisp Package Archive, is a repository of over 5000 automatically updated and curated Emacs Lisp packages. With access to so many extensions, configuring my Emacs consisted of customizing a few built-in properties and aggressively yanking other people's code — just like I've done with VS Code.

In a way, the "real power" of Emacs is lost on me. There are many aspects of my workflow which would benefit from automation. However, as with all automation endeavors, how do you know if the cost of writing that script is less than the total time spent doing the task manually?

Still, using Emacs means you *most probably can* program that niche script, if you really want to. And even better, someone might've come up with a snippet to do it already.

Eye Candy

Honestly, I'm not a very good hacker. Because I care about the way user-facing programs look and feel. And while Emacs is very attractive to the tinkerer in me, its UI philosophy is in stark contrast to anything made in the last few decades.


GNU Emacs - GNU Emacs at sudra
File Edit Options Buffers Tools Help
New File Open Open Directory Close Save Undo Cut Copy Paste Search

Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Emacs Tutorial	Learn basic keystroke commands
Emacs Guided Tour	Overview of Emacs features at gnu.org
View Emacs Manual	View the Emacs manual using Info
Absence of Warranty	GNU Emacs comes with *ABSOLUTELY NO WARRANTY*
Copying Conditions	Conditions for redistributing and changing Emacs
Ordering Manuals	Purchasing printed copies of manuals

To start... Open a File Open Home Directory Customize Startup
To quit a partially entered command, type `Control-g`.

This is GNU Emacs 28.1 (build 1, x86_64-suse-linux-gnu, GTK+ Version 3.24.34, cairo version 1.16.0)
Copyright (C) 2022 Free Software Foundation, Inc.

Auto-save file lists were found. If an Emacs session crashed recently,
type M-x recover-session RET to recover the files you were editing.

```
U:%%-  *GNU Emacs*    All L3     (Fundamental)
```

The **default** Emacs greeting window

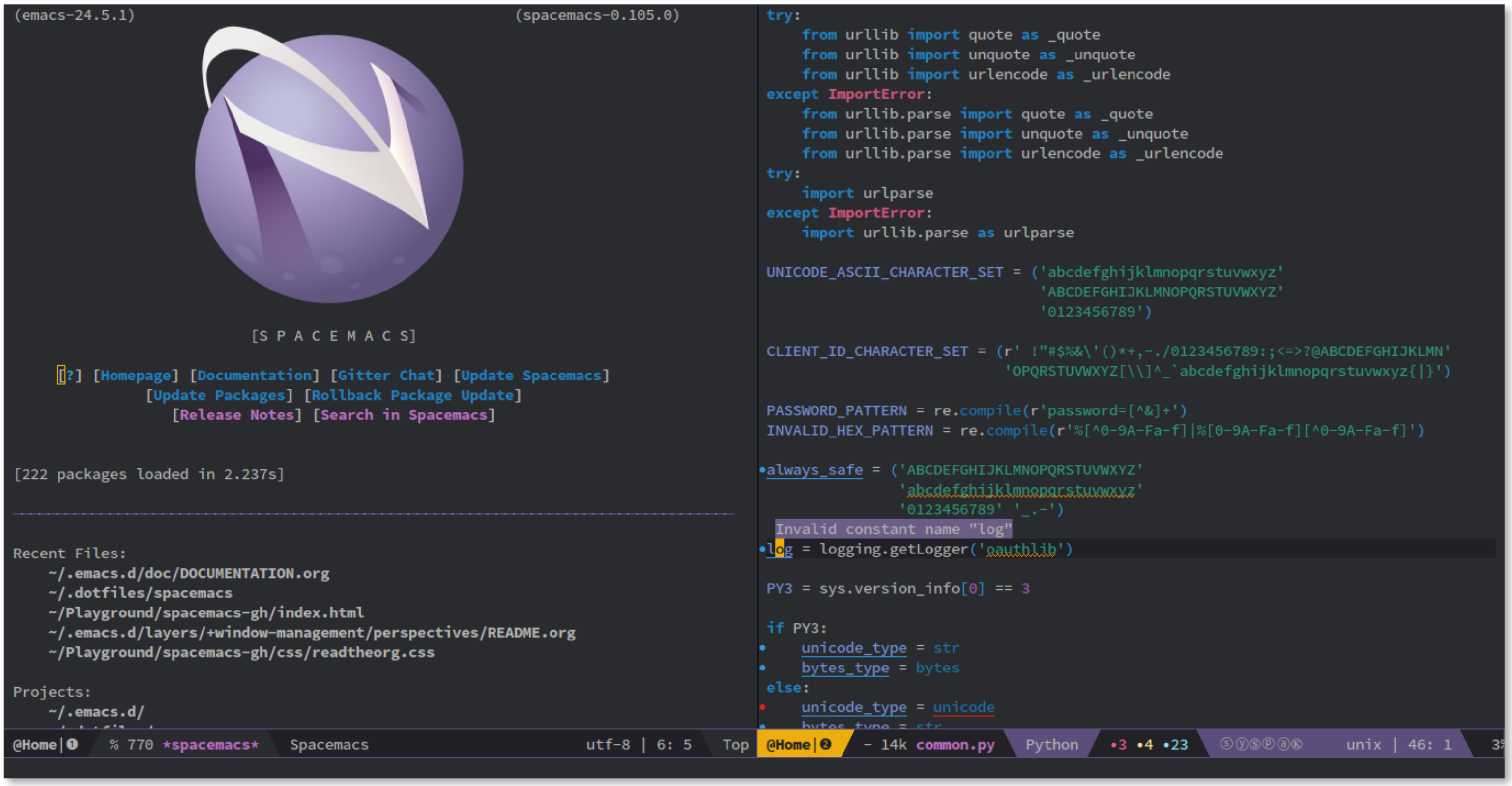
Back in 2020, a lively thread was started on the `emacs-devel` mailing list in reaction to a Redditor asking "Why is Emacs so square?". Somewhere in the chain of replies, one can find the following message, by the original author of GNU Emacs:

> Perhaps we should implement a mode that puts cosmetics on Emacs so it will appeal to those who judge by the surface of things.

— Richard Stallman

You can read more about this drama, the view of Emacs developers around UI/UX, and the challenge of making Emacs accessible to a wider audience in the excellent article Making Emacs Popular Again, by the LWN.net computing news site.

No big deal. Emacs is ridiculously malleable and I'm not the first person to attempt revamping the UI.

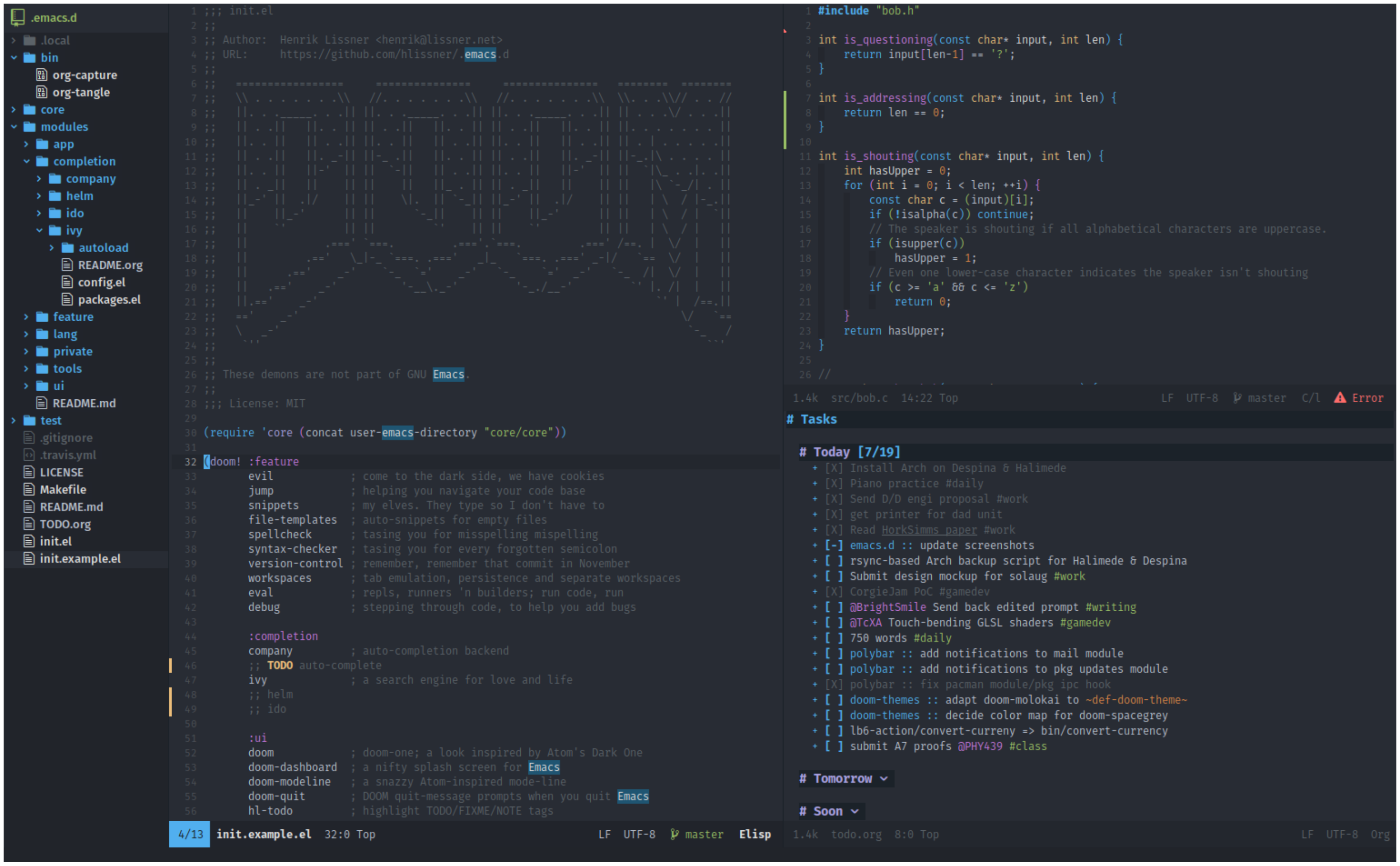


```
(emacs-24.5.1)                                                     (spacemacs-0.105.0)

                                  [S P A C E M A C S]

     [?] [Homepage] [Documentation] [Gitter Chat] [Update Spacemacs]
                 [Update Packages] [Rollback Package Update]
                    [Release Notes] [Search in Spacemacs]

[222 packages loaded in 2.237s]

Recent Files:
    ~/.emacs.d/doc/DOCUMENTATION.org
    ~/.dotfiles/spacemacs
    ~/Playground/spacemacs-gh/index.html
    ~/.emacs.d/layers/+window-management/perspectives/README.org
    ~/Playground/spacemacs-gh/css/readtheorg.css

Projects:
    ~/.emacs.d/

@Home|1   % 770 *spacemacs*   Spacemacs                              utf-8 | 6: 5   Top
```

```python
try:
    from urllib import quote as _quote
    from urllib import unquote as _unquote
    from urllib import urlencode as _urlencode
except ImportError:
    from urllib.parse import quote as _quote
    from urllib.parse import unquote as _unquote
    from urllib.parse import urlencode as _urlencode
try:
    import urlparse
except ImportError:
    import urllib.parse as urlparse

UNICODE_ASCII_CHARACTER_SET = ('abcdefghijklmnopqrstuvwxyz'
                               'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
                               '0123456789')

CLIENT_ID_CHARACTER_SET = (r' !"#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN'
                           'OPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}')

PASSWORD_PATTERN = re.compile(r'password=[^&]+')
INVALID_HEX_PATTERN = re.compile(r'%[^0-9A-Fa-f]|%[0-9A-Fa-f][^0-9A-Fa-f]')

always_safe = ('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
               'abcdefghijklmnopqrstuvwxyz'
               '0123456789' '_.-')
 Invalid constant name "log"
log = logging.getLogger('oauthlib')

PY3 = sys.version_info[0] == 3

if PY3:
    unicode_type = str
    bytes_type = bytes
else:
    unicode_type = unicode
```

```
@Home|2  - 14k common.py   Python   •3 •4 •23   ⓈⓨⓈⓅⓐⓚ   unix | 46: 1
```

Spacemacs, a popular Emacs distribution.



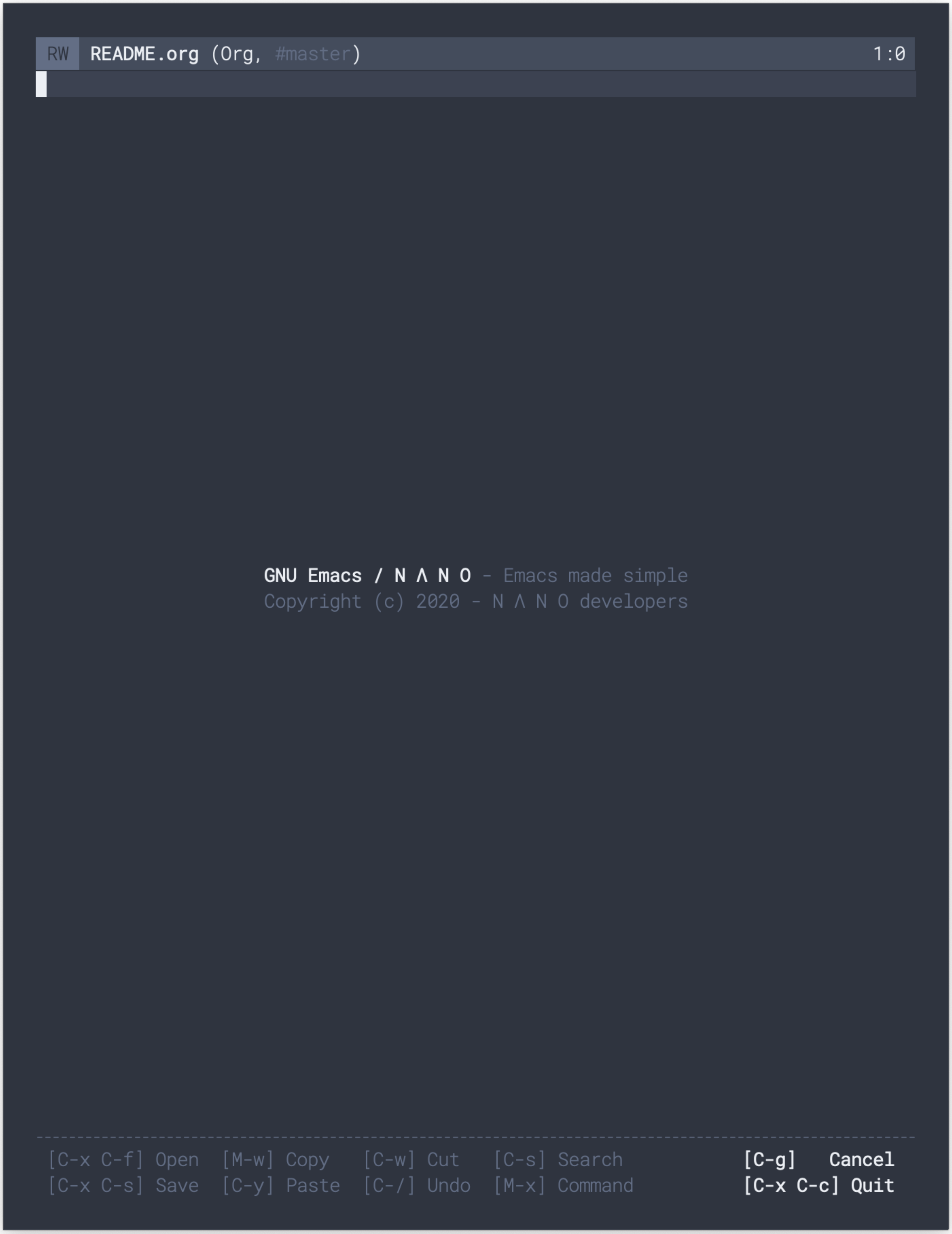
```
.emacs.d
› .local
⌄ bin
    org-capture
    org-tangle
› core
⌄ modules
  › app
  ⌄ completion
    › company
    › helm
    › ido
    ⌄ ivy
      › autoload
        README.org
        config.el
        packages.el
  › feature
  › lang
  › private
  › tools
  › ui
    README.md
› test
  .gitignore
  .travis.yml
  LICENSE
  Makefile
  README.md
  TODO.org
  init.el
  init.example.el
```

```
;;; init.el
;;
;; Author:  Henrik Lissner <henrik@lissner.net>
;; URL:     https://github.com/hlissner/.emacs.d
;;
;;   =================     ===============     ===============   ========  ========
;;   \\ . . . . . . .\\   //. . . . . . .\\   //. . . . . . .\\  \\. . .\\// . . //
;;   ||. . ._____. . .|| ||. . ._____. . .|| ||. . ._____. . .|| || . . .\/ . . .||
;;   || . .||   ||. . || || . .||   ||. . || || . .||   ||. . || ||. . . . . . . ||
;;   ||. . ||   || . .|| ||. . ||   || . .|| ||. . ||   || . .|| || . | . . . . .||
;;   || . .||   ||. _-|| ||-_ .||   ||. . || || . .||   ||. _-|| ||-_.|\ . . . . ||
;;   ||. . ||   ||-'  || ||  `-||   || . .|| ||. . ||   ||-'  || ||  `|\_ . .|. .||
;;   || . _||   ||    || ||    ||   ||_ . || || . _||   ||    || ||   |\ `-_/| . ||
;;   ||_-' ||  .|/    || ||    \|.  || `-_|| ||_-' ||  .|/    || ||   | \  / |-_.||
;;   ||    ||_-'      || ||      `-_||    || ||    ||_-'      || ||   | \  / |  `||
;;   ||    `'         || ||         `'    || ||    `'         || ||   | \  / |   ||
;;   ||            .===' `===.         .==='.`===.         .===' /==. |  \/  |   ||
;;   ||         .=='   \_|-_ `===. .==='   _|_   `===. .===' _-|/   `==  \/  |   ||
;;   ||      .=='    _-'    `-_  `='    _-'   `-_    `='  _-'   `-_  /|  \/  |   ||
;;   ||   .=='    _-'          '-__\._-'         '-_./__-'         `' |. /|  |   ||
;;   ||.=='    _-'                                                     `' |  /==.||
;;   =='    _-'                                                            \/   `==
;;   \   _-'                                                                `-_   /
;;    `''                                                                      ``'
;;
;;
;; These demons are not part of GNU Emacs.
;;
;;; License: MIT

(require 'core (concat user-emacs-directory "core/core"))

(doom! :feature
       evil             ; come to the dark side, we have cookies
       jump             ; helping you navigate your code base
       snippets         ; my elves. They type so I don't have to
       file-templates   ; auto-snippets for empty files
       spellcheck       ; tasing you for misspelling mispelling
       syntax-checker   ; tasing you for every forgotten semicolon
       version-control  ; remember, remember that commit in November
       workspaces       ; tab emulation, persistence and separate workspaces
       eval             ; repls, runners 'n builders; run code, run
       debug            ; stepping through code, to help you add bugs

       :completion
       company          ; auto-completion backend
       ;; TODO auto-complete
       ivy              ; a search engine for love and life
       ;; helm
       ;; ido

       :ui
       doom             ; doom-one; a look inspired by Atom's Dark One
       doom-dashboard   ; a nifty splash screen for Emacs
       doom-modeline    ; a snazzy Atom-inspired mode-line
       doom-quit        ; DOOM quit-message prompts when you quit Emacs
       hl-todo          ; highlight TODO/FIXME/NOTE tags
4/13  init.example.el  32:0 Top                      LF  UTF-8  master  Elisp
```

```
#include "bob.h"

int is_questioning(const char* input, int len) {
    return input[len-1] == '?';
}

int is_addressing(const char* input, int len) {
    return len == 0;
}

int is_shouting(const char* input, int len) {
    int hasUpper = 0;
    for (int i = 0; i < len; ++i) {
        const char c = (input)[i];
        if (!isalpha(c)) continue;
        // The speaker is shouting if all alphabetical characters are uppercase.
        if (isupper(c))
            hasUpper = 1;
        // Even one lower-case character indicates the speaker isn't shouting
        if (c >= 'a' && c <= 'z')
            return 0;
    }
    return hasUpper;
}

//
1.4k  src/bob.c  14:22 Top                LF  UTF-8  master  C/l  Error
```

```
# Tasks

 # Today [7/19]
  + [X] Install Arch on Despina & Halimede
  + [X] Piano practice #daily
  + [X] Send D/D engi proposal #work
  + [X] get printer for dad unit
  + [X] Read HorkSimms paper #work
  + [-] emacs.d :: update screenshots
  + [ ] rsync-based Arch backup script for Halimede & Despina
  + [ ] Submit design mockup for solaug #work
  + [X] CorgieJam PoC #gamedev
  + [ ] @BrightSmile Send back edited prompt #writing
  + [ ] @TcXA Touch-bending GLSL shaders #gamedev
  + [ ] 750 words #daily
  + [ ] polybar :: add notifications to mail module
  + [ ] polybar :: add notifications to pkg updates module
  + [X] polybar :: fix pacman module/pkg ipc hook
  + [ ] doom-themes :: adapt doom-molokai to ~def-doom-theme~
  + [ ] doom-themes :: decide color map for doom-spacegrey
  + [ ] lb6-action/convert-curreny => bin/convert-currency
  + [ ] submit A7 proofs @PHY439 #class

 # Tomorrow ⌄

 # Soon ⌄
1.4k  todo.org  8:0 Top                                  LF  UTF-8  Org
```

DOOM, a faster alternative to Spacemacs.



```
RW  README.org (Org, #master)                                    1:0


                 GNU Emacs / N Λ N O - Emacs made simple
                 Copyright (c) 2020 - N Λ N O developers


----------------------------------------------------------------------
[C-x C-f] Open  [M-w] Copy   [C-w] Cut   [C-s] Search       [C-g]     Cancel
[C-x C-s] Save  [C-y] Paste  [C-/] Undo  [M-x] Command      [C-x C-c] Quit
```

N Λ N O, a very lightweight and modular configuration by **Nicolas P. Rougier**.

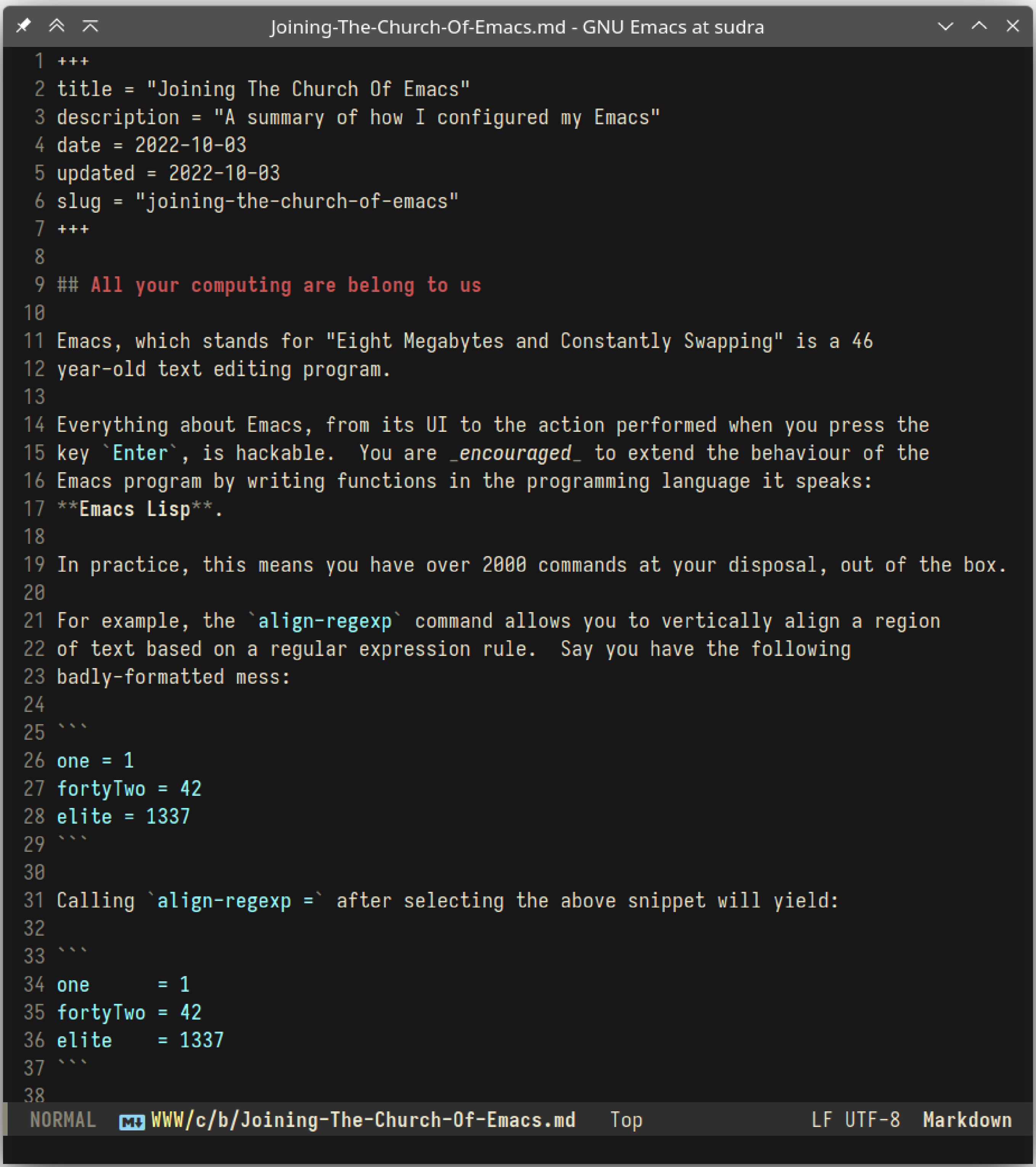
With a good idea of what's possible, I sought to customize Emacs to my liking. I was determined to make it look *OK*. First, I removed the title and menu bars, the scroll bar, and any cruft that cluttered my 14" laptop screen.

Next, I wrote a custom dark color scheme, which was surprisingly simple. In a nutshell, every UI element has a corresponding *face*; a set of properties that controls its background color, its font-weight, its size and so on. Consequently, writing an Emacs theme felt a lot like writing CSS.

Finally, I needed to get rid of the ugly status bar, or in Emacs-speak, the *modeline*. However, handling how the modeline renders each piece of information available user-facing was a complex undertaking and I didn't have enough knowledge of how Emacs works to accomplish what I wanted.

For example, N Λ N O's modeline code alone is more than 500 lines. For comparison, my entire configuration is under 400 lines of Emacs Lisp. Writing a fancy and functional modeline would shatter my complexity budget.

Instead, I "borrowed" DOOM's modeline, which was luckily published as a standalone package on MELPA. With that, I had something that indeed looked *OK*, at least to me.



```
+++
title = "Joining The Church Of Emacs"
description = "A summary of how I configured my Emacs"
date = 2022-10-03
updated = 2022-10-03
slug = "joining-the-church-of-emacs"
+++

## All your computing are belong to us

Emacs, which stands for "Eight Megabytes and Constantly Swapping" is a 46
year-old text editing program.

Everything about Emacs, from its UI to the action performed when you press the
key `Enter`, is hackable.  You are _encouraged_ to extend the behaviour of the
Emacs program by writing functions in the programming language it speaks:
**Emacs Lisp**.

In practice, this means you have over 2000 commands at your disposal, out of the box.

For example, the `align-regexp` command allows you to vertically align a region
of text based on a regular expression rule.  Say you have the following
badly-formatted mess:

```
one = 1
fortyTwo = 42
elite = 1337
```

Calling `align-regexp =` after selecting the above snippet will yield:

```
one      = 1
fortyTwo = 42
elite    = 1337
```

```

NORMAL WWW/c/b/Joining-The-Church-Of-Emacs.md Top LF UTF-8 Markdown

The source document of this article opened in my Emacs.

## Modal editing

I don't like Emacs' strong dependence on modifier keys. By default, opening a file is `C-x C-f` (i.e. `Ctrl+X` followed by `Ctrl+F`) which I find to be very awkward. Instead, I appreciate (Neo)Vim's modal editing approach better.

Modal text editing means for example that instead of typing `C-n` to move the cursor down one line, you only type `j`. The downside of this is the introduction of distinct *modes*. In "normal" mode, pressing `j` would indeed move the cursor down one line, but in "insert" mode the "j" character would be typed instead. In (Neo)Vim, switching to "normal" mode is done by pressing `ESC` while `i` sends you back into "insert" mode. There are other modes as well, such as "visual" (for making selections) and "terminal" (for terminal emulation).

There is a popular Emacs package called `Evil`, which provides Vim[2] emulation. This makes Emacs an **almost** drop-in replacement for (Neo)Vim, which was important to me as I was a very avid Neovim user before deciding to switch teams.

Evil worked well for a while. Then I started to notice its keybindings weren't working consistently across all of Emacs. I had my *leader* key set to Space which meant I could configure it to open a file by typing `SPC f` from anywhere. A big improvement over `C-x C-f`. This approach of using the Space key for keybindings was popularized by Spacemacs. Yes, the word "space" in "Spacemacs" supposedly refers to the Space key.

It turns out there are many solutions to my problem with Emacs modifier keys, other than Evil. God Mode for example works by straight out removing `Ctrl` and `Alt` from all keybindings. So that `C-x C-f` becomes `x f`. This was nice and all, but I wanted something more Vim-like. And I was in luck.

Meow is yet another modal editing mode for Emacs. Here is why I decided to go with it:

- Inspired by God Mode, any keybinding of the form `C-a C-b C-c` is by default routed[3] to `SPC a b c`. This meant I wouldn't have to re-bind every single Emacs command in order to avoid modifiers.

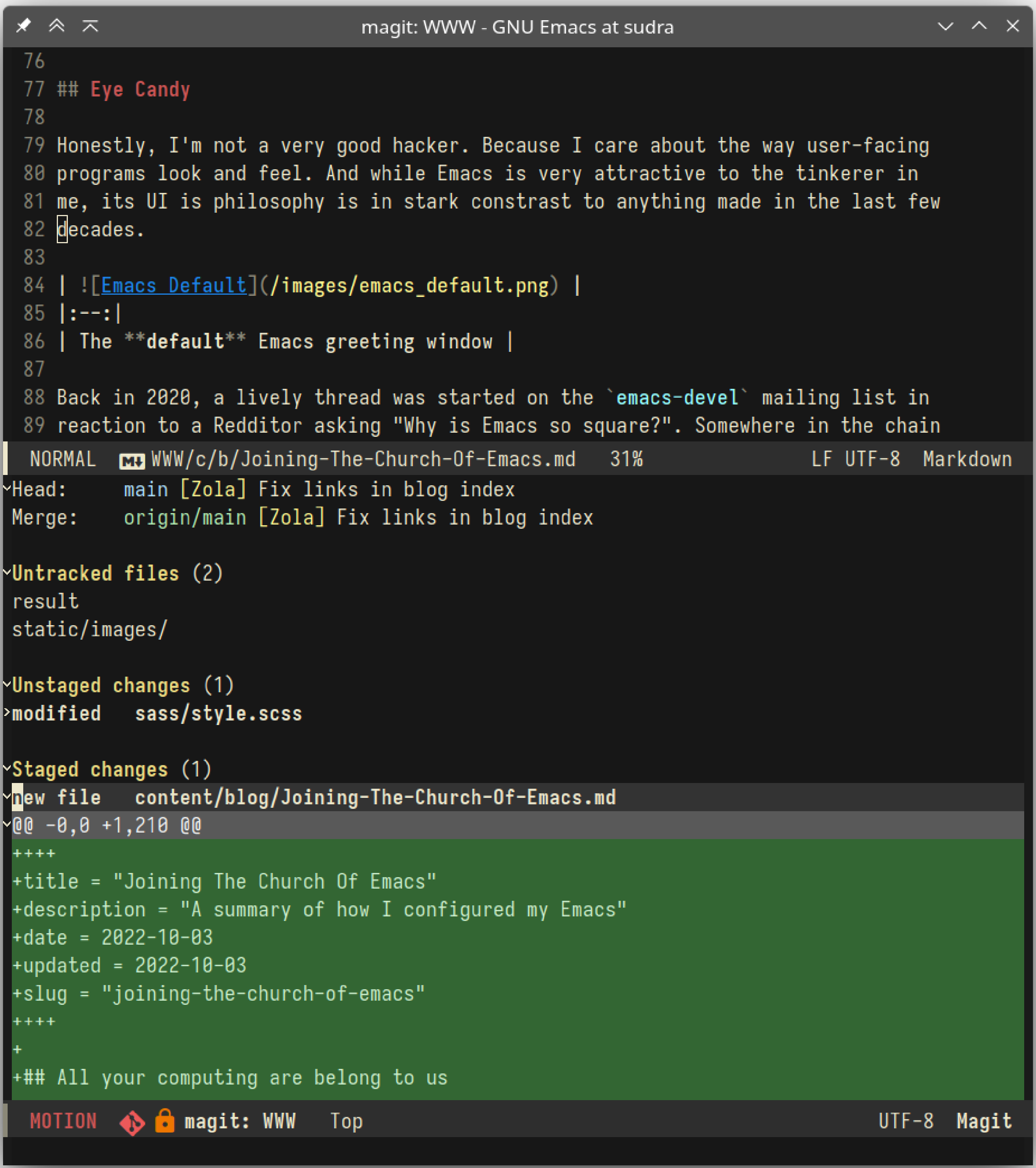
- Meow's modal editing model is very much an improvement over classic Vim. It applies many of the ideas recently introduced by Kakoune. Read this blog article for a more thorough comparison of Vim and Meow.

- Meow was designed for consistency with Emacs. Unlike Evil, it provides built-in support for using Space as the leader key. Now my `SPC f` keybinding works everywhere.

What was lost in this little transition is my Vim muscle memory. On the upside, I found myself to be using fewer keystrokes to achieve the same results. And in any case, one could not expect to use Vi modal editing *everywhere*, and where it is supported, it's not the default.

## It's Magit!

Magit is an Emacs package that provides a text-based interface to Git. Unlike most GUI interfaces to Git, it exposes everything from the common to the most advanced Git features with a few keystrokes. I can type `SPC v` (for **V**ersion control) to bring up the Magit interface:

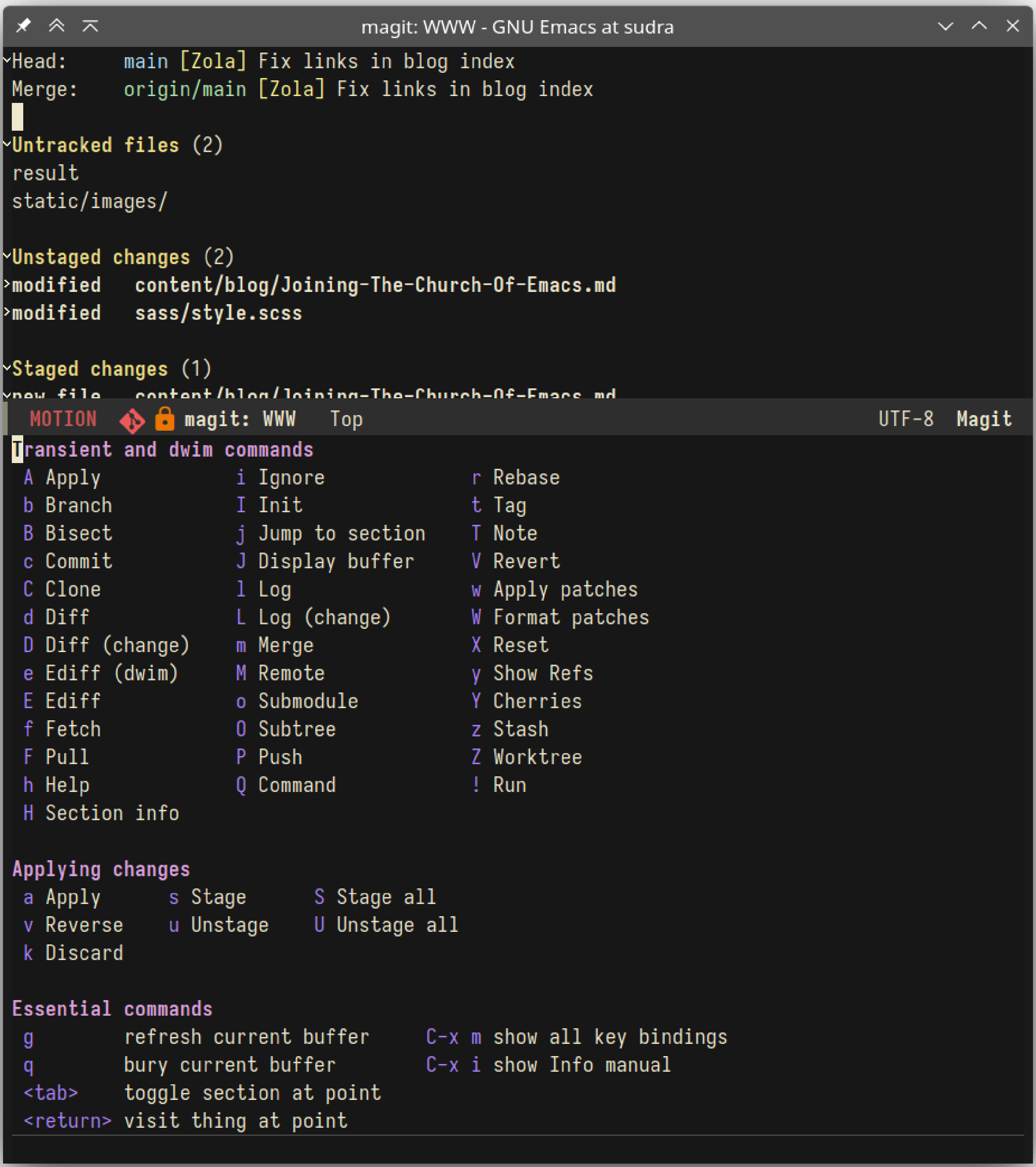


```
Eye Candy

Honestly, I'm not a very good hacker. Because I care about the way user-facing
programs look and feel. And while Emacs is very attractive to the tinkerer in
me, its UI is philosophy is in stark constrast to anything made in the last few
decades.

![Emacs Default](/images/emacs_default.png)
The **default** Emacs greeting window

Back in 2020, a lively thread was started on the `emacs-devel` mailing list in
reaction to a Redditor asking "Why is Emacs so square?". Somewhere in the chain
 NORMAL WWW/c/b/Joining-The-Church-Of-Emacs.md   31%   LF UTF-8 Markdown
Head: main [Zola] Fix links in blog index
Merge: origin/main [Zola] Fix links in blog index

Untracked files (2)
result
static/images/

Unstaged changes (1)
modified   sass/style.scss

Staged changes (1)
new file   content/blog/Joining-The-Church-Of-Emacs.md
@@ -0,0 +1,210 @@
++++
+title = "Joining The Church Of Emacs"
+description = "A summary of how I configured my Emacs"
+date = 2022-10-03
+updated = 2022-10-03
+slug = "joining-the-church-of-emacs"
++++
+
+## All your computing are belong to us
 MOTION   magit: WWW   Top   UTF-8 Magit
```

A Magit buffer open to the bottom of my window

As you can see, it displays the latest local and remote commits, untracked files , and (un)staged changes. You can stage individual sections of a file by simply scrolling to them and pressing s; something that is much less ergonomic in command-line Git and is done through Interactive Staging.

Next, press h to bring out a view of all possible actions supported by Magit:



```
Head:     main [Zola] Fix links in blog index
Merge:    origin/main [Zola] Fix links in blog index

Untracked files (2)
result
static/images/

Unstaged changes (2)
modified   content/blog/Joining-The-Church-Of-Emacs.md
modified   sass/style.scss

Staged changes (1)
new file   content/blog/Joining-The-Church-Of-Emacs.md
```

```
 MOTION   magit: WWW   Top                                        UTF-8  Magit
Transient and dwim commands
 A Apply              i Ignore              r Rebase
 b Branch             I Init                t Tag
 B Bisect             j Jump to section     T Note
 c Commit             J Display buffer      V Revert
 C Clone              l Log                 w Apply patches
 d Diff               L Log (change)        W Format patches
 D Diff (change)      m Merge               X Reset
 e Ediff (dwim)       M Remote              y Show Refs
 E Ediff              o Submodule           Y Cherries
 f Fetch              O Subtree             z Stash
 F Pull               P Push                Z Worktree
 h Help               Q Command             ! Run
 H Section info

Applying changes
 a Apply       s Stage       S Stage all
 v Reverse     u Unstage     U Unstage all
 k Discard

Essential commands
 g          refresh current buffer      C-x m show all key bindings
 q          bury current buffer         C-x i show Info manual
 <tab>      toggle section at point
 <return>   visit thing at point
```

The Magit help buffer

This is a high-level view of everything you can do in (Ma)git. Pressing any of the suggested keys will bring out further menus where Magit will narrow down your action all while showing you all available options, at every step.

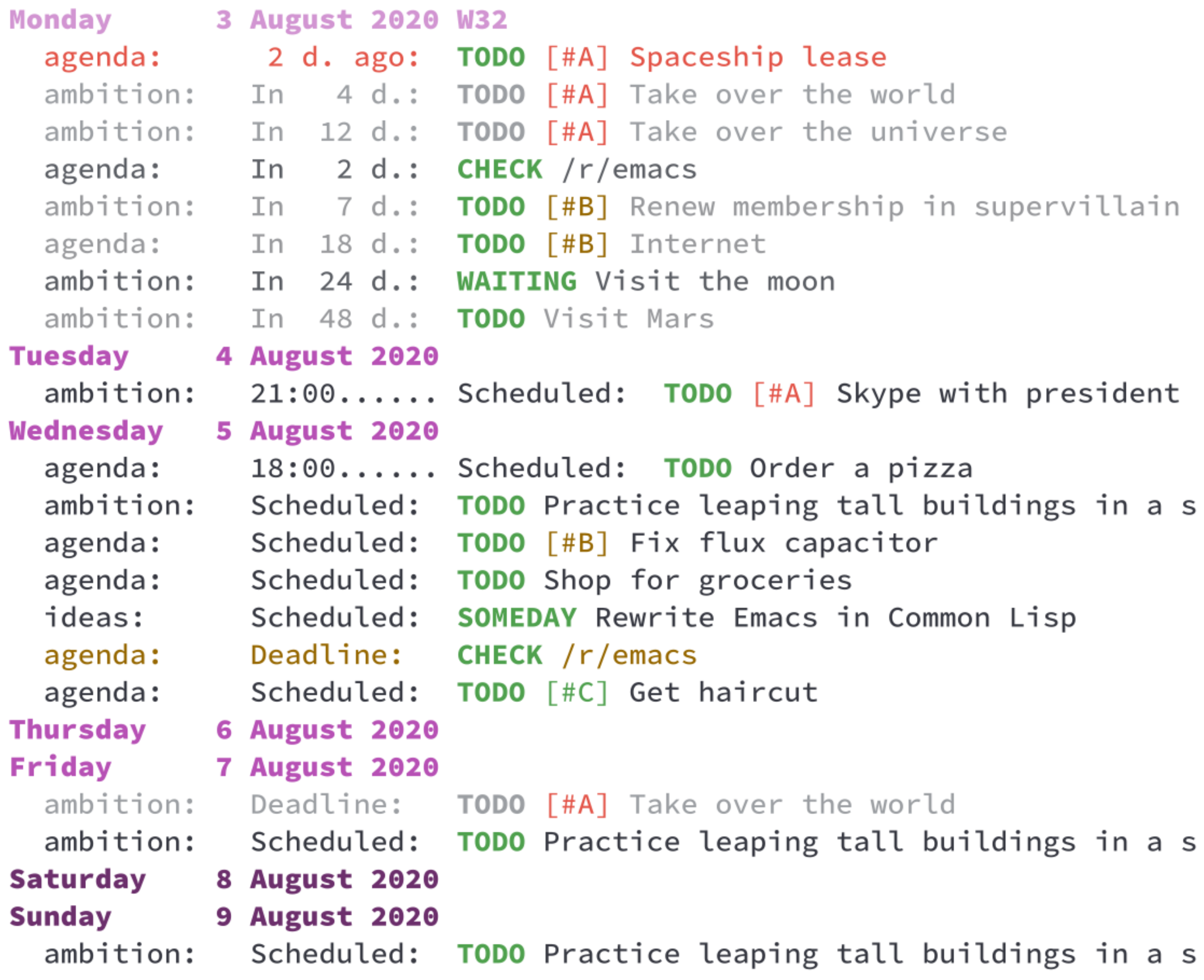
This transformed my Git workflow from reading the man pages and copying commands from Stack Overflow, to completing my way through the Magit buffer. It also helped me discover many cool features I wasn't aware of, such as Auto-squashing; exactly like one discovers functions by scrolling LSP auto-completions.

Org-anize all the things

Emacs comes with support for a document format called Org:

> **[Org is] a GNU Emacs major mode for keeping notes, authoring documents, computational notebooks, literate programming, maintaining to-do lists, planning projects, and more — in a fast and effective plain-text system.**

I use Org for managing notes and to-do lists. Interestingly, the Org file format includes syntax for scheduling to-do items, managing priorities, setting deadlines, enforcing habits, and so on. All in plain text. With this information, Org Mode can scan all your Org files to generate an Agenda view:



```
Monday       3 August 2020 W32
  agenda:       2 d. ago:  TODO [#A] Spaceship lease
  ambition:   In   4 d.:  TODO [#A] Take over the world
  ambition:   In  12 d.:  TODO [#A] Take over the universe
  agenda:     In   2 d.:  CHECK /r/emacs
  ambition:   In   7 d.:  TODO [#B] Renew membership in supervillain
  agenda:     In  18 d.:  TODO [#B] Internet
  ambition:   In  24 d.:  WAITING Visit the moon
  ambition:   In  48 d.:  TODO Visit Mars
Tuesday      4 August 2020
  ambition:   21:00...... Scheduled:  TODO [#A] Skype with president
Wednesday    5 August 2020
  agenda:     18:00...... Scheduled:  TODO Order a pizza
  ambition:   Scheduled:  TODO Practice leaping tall buildings in a s
  agenda:     Scheduled:  TODO [#B] Fix flux capacitor
  agenda:     Scheduled:  TODO Shop for groceries
  ideas:      Scheduled:  SOMEDAY Rewrite Emacs in Common Lisp
  agenda:     Deadline:   CHECK /r/emacs
  agenda:     Scheduled:  TODO [#C] Get haircut
Thursday     6 August 2020
Friday       7 August 2020
  ambition:   Deadline:   TODO [#A] Take over the world
  ambition:   Scheduled:  TODO Practice leaping tall buildings in a s
Saturday     8 August 2020
Sunday       9 August 2020
  ambition:   Scheduled:  TODO Practice leaping tall buildings in a s
```

Actionable Agendas, from the Org Mode website.

Next, I use the proprietary but very convenient Dropbox service to sync my Org files with my Android phone. And luckily for me, I don't have to run Emacs inside Termux. Because the Orgzly application provides a touch-screen-friendly interface to my Org files, and can send push notifications for upcoming to-do items.

This is *very* basic usage of Org Mode. If you're more curious about how some of the more demanding work schedules look like in Org Mode, I recommend reading Get Things Done With Emacs by Nicolas P. Rougier.

Closing words

This was a summary of *my experience* with configuring and using Emacs. I tried to focus on what sets Emacs apart from other text editors and IDEs. And what I found to be so special about it. Throughout the last two years, I have used Vim, Neovim, VS Code, Jetbrains IDEs, DOOM Emacs, Spacemacs and now *vanilla* Emacs.

It is certainly true that Emacs isn't a very popular IDE. Only 5.33% of respondents in the 2021 Stack Overflow Developer Surgery claimed to use the thing. Despite that, there is a rich community around Emacs, comprised of people determined to carry it through the 21st century. The fact that this software project existed continuously for nearly half a century is a testament to the stubbornness of the hackers involved with it. This gives Emacs a unique sense of timelessness.

> **It's a tale as old as time: a stubborn, shell-dwelling, and melodramatic vimmer -- tormented by Vimscript and his boundless productivity -- makes a formal request to the netherworld for a transfer. They agree. The terms? He must lure more unsuspecting souls into a life of eternal bikeshedding. Now he runs the place.**
>
> **— Henrik Lissner, author of DOOM Emacs**

[1] Assuming that your computation is expressible in the Emacs Lisp programming language. ↩

[2] Vi refers to the original version of the text editor commonly known as Vim. In fact, Vim stands for "Vi IMproved". Neovim is a more modern project which uses the programming language Lua instead of the often dreaded Vimscript for writing extensions. Neovim is compatible with Vim v8. ↩

[3] There are more rules to deal with keybindings that use `Alt`. For example `C-x M-t` is `SPC x m t` and `C-M-x` is `SPC g x`. ↩