916 days of Emacs

Table of Contents

<Expand> <Collapse>

> *Poof I made my free-time disappear*

  - Ellis Kenyő, on being called an "elisp mage"

Little did I know on the fateful day of **[2020-10-09 Fri]**, when I had installed GNU Emacs. I wasn't thinking about the ethical aspects of free software, the aesthetics of Lisp, or these other things with which an occasional layperson migh explain how an almost half a century old program can still be in active use.

In fact, when considering using software X for anything, the most important question to me was: can X provide a better user experience? For Emacs, the answer to most of these questions turned out to be yes.

So over time, Emacs has become my programming environment, email client, window manager, knowledge base, and a lot mor I think I ended up using Emacs for almost as many things as possible; I even authored a few packages that implement certain parts of my workflows that weren't readily available.

Among other things, the Emacs community is responsible for my introduction to Zettelkasten, RSS, Lisps… Perhaps even m English became slightly less broken because Emacs is so text-centered. A lot has changed over the course of these shor 2.5 years.

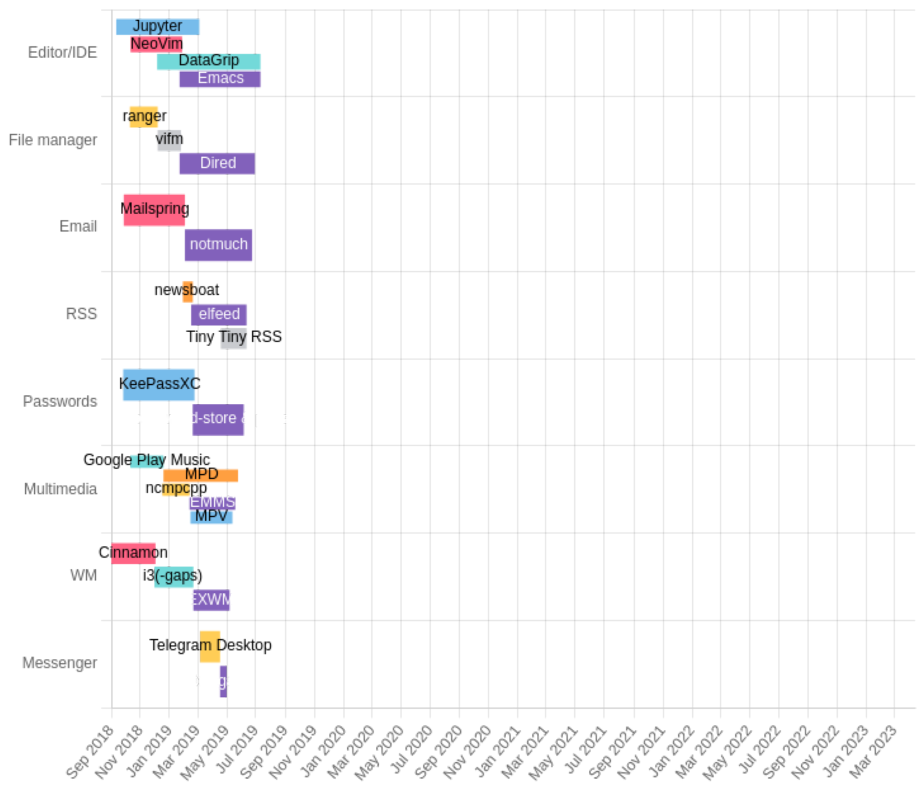
Anyway, this post is an attempt to quantify some aspects of that story. The numbers mostly come from projects called ActivityWatch and WakaTime.

Mostly I'm curious myself, but also every now and then I see Emacs people discussing their journeys through the Elisp-land, or a potential convert wondering whether this rabbit hole is worth investigating. If any of this applies to you, you might find something interesting in this document.

Everything goes into Emacs

As I mentioned earlier, I use Emacs for a lot of things, which are described in my Emacs config. Fig. 1 shows how Emac replaced various programs over time.

Figure 1. Everything goes into Emacs



As you can see, I used Neovim for a little over a year. We'll get into some numbers on that later.

The process of moving from knowing nothing about Emacs to using EXWM took about 13 months.

Fig. 2 shows the dynamics of the direct screen time ratio spent in Emacs per month, i.e. the average number of non-AFK seconds in the Emacs window.

It’s hard to discern any general trend here. It appears that the ratio started at 0.2 in October 2020, oscillated arour 0.3 for about 7 months, then moved closer to 0.4 until January 2023, after which jumped to 0.45-0.5.

The three peaks in September 2021 (0.526), January 2022 (0.532), and August 2022 (0.568) may correspond to my vacations during which I didn’t have to spend time in Chrome DevTools (I do web development as my “primary” job), but I’m not entirely sure.

The jump in January 2023 definitely matches my adoption of telega.el instead of the official desktop client. The time redistributes rather cleanly in the detailed ActivityWatch data.

It’s also interesting that switching from i3 to EXWM didn’t seem to have any distinguishable effects.

The mean Emacs screen time ratios are 0.39 since October 2020 and 0.47 since January 2023. So, as you might infer, Emac is quite prominent in my PC usage.

Time spent in Emacs

Now let’s examine the structure of time spent in Emacs. Fig. 3 shows how many Emacs-hours per month I spent on differer activities, and Fig. 4 shows the same in stacked form.

Unlike Fig. 2, the time here is calculated with a 15-minute timeout preference, as it’s done in WakaTime. For instance if I work on a project in Emacs for 10 minutes, then switch to something else for 10 minutes (i.e. no heartbeats recorded during that time), then return to the project another 10 minutes, this will be counted as 30 minutes in that project.

This is mostly so because it’s the default format for the WakaTime export, but I also believe it’s reasonable since I may open package documentation during configuration, experiment in scratch buffers while working on a package, and so on. This time really has to be included in the final tally.

Of course, this will also include all the times I was distracted by the System Crafters Discord server, emacs.ch Mastodon instance, or whatever else. Therefore, consider the numbers that follow as an upper bound.

The categories are as follows:

- **Config** ([REDACTED] total hours, [REDACTED]% of all time)
  Time spent on actual Emacs configuration.
- **Emacs Packages** ([REDACTED] total hours, [REDACTED]% of all time)
  Time spent in other Emacs Lisp files, such as writing my packages or debugging other packages. See the packages section.
- **Org Mode** ([REDACTED] total hours, [REDACTED]% of all time)
  Time spent in my `org-mode` project, which is mostly org-journal, org-roam, and project management. By the way, guess the month in which I read Sönke Ahrens’ book about Zettelkasten.
- **sqrtminusone.xyz** ([REDACTED] total hours, [REDACTED]% of all time)
  Working on this strange little website.
- **Other Code** ([REDACTED] total hours, [REDACTED]% of all time)
  Doing something marginally useful in Emacs, which is mostly work, education, and a few personal projects unrelated to Emacs.
- **Misc** ([REDACTED] total hours, [REDACTED]% of all time) Time spent in Emacs but not in an actual project (i.e. accounted by the window watcher of ActivityWatch but not WakaTime, which watches for files). That includes reading RSS, writing emails, using messengers, doing some idle experimentation in scratch buffers, etc.

The [REDACTED]% of time spent on configuration is actually less than I expected. Unsurprisingly, the first month had th highest value of around 20% (I used Emacs without WakaTime for a few days).

By the way, I spent just 39.0 hours configuring Neovim, although the number is probably not representative anymore because Neovim has changed a lot over these 2 years.

Switching from Neovim

The period of my transition from Neovim to Emacs seems particularly interesting. Fig 5 zooms in on that, with the switc represented by “Other Code (Emacs)” replacing “Other Code (Vim)”.

It appears that getting from zero to somewhat productive took me about 11.1 hours over 4 days of just experimenting wi
Emacs, and one day with 3.8 hours on configuring and 4.4 hours on coding, apparently alternating between the two.

Configuration

Now, let’s examine where these [REDACTED] config hours went.

Configuration sizes are a common topic of discussion among Emacs users. I’d guess that mine falls into the category of the longest, although maybe I’ll do some research on that someday. Fig. 6 shows how my configuration size changed over time.

So, my Emacs.org is [REDACTED] lines long, and the resulting init.el is [REDACTED] lines long.

As you can see, I switched to literate configuration pretty early on, and so far, I have not regretted it. It’s also interesting to note how the two sizes diverged as I was writing more elaborate commentary.

Also, I never had any substantial issues with maintaining that configuration. Perhaps Emacs Bankruptcies are just not that common nowadays.

For the sake of completeness, let’s compare that to my Neovim usage. Fig. 7 shows the dynamics of config size for the first 400 days of using both programs.

As I previously mentioned, Neovim (or rather its ecosystem) seems to have undergone significant changes since I last used it, so my number of [REDACTED] init.vim lines may no longer be relevant. Nonetheless, it’s quite interesting.

Emacs packages

Working with Emacs packages was an interesting experience, not least because it was my first experience with Lisp. Fig
8 shows the breakdown of the [REDACTED] hours I spent on that.

As I expected, my org-journal-tags tops the chart with [REDACTED] hours. The most interesting part was implementing se
logic on the org-journal entities to create a query engine. I’m fairly certain that I’m the only user of this package, but I use it all the time.

The second place, “Unknown project”, stands for Emacs Lisp files that didn’t belong to any project, which should be mostly built-in Emacs files.

My elfeed-summary ([REDACTED] hours), lyrics-fetcher ([REDACTED] hours), and reverso ([REDACTED] hours) are also among the packages that I use almost daily. Thus, I do not regret investing time in developing any of those.

`org-roam`

It’s not directly related to Emacs, but I include it here because it’s highly unlikely that I would have heard the ter
“Zettelkasten” outside the Emacs space.

I already mentioned Sönke Ahrens’ book, but I believe the website zettelkasten.de would be a better resource if you ar
curious about that. And I was initially made curious by this stream of David Wilson.

Anyway, Fig. 9 shows the dynamics of my org-roam node count over time. A significant fraction of my [REDACTED] hours spent on Org Mode went there. Although I don’t have any particular goals in this regard.

Some observations

Let’s see where all of that leads us.

As I said, I started from the point of zero experience with Lisp. I had a degree in software engineering, but I don’t feel like it has helped me in any direct sense. At most, it exposed me to different kinds and concepts of programming, but I am confident that it’s anything but a prerequisite, as also shown by the story of Protesilaos.

The number of [REDACTED] total hours of configuration may seem huge, but I don’t think it’s that much over 2.5 years ar
in comparison to the alternatives. For instance, it would take 6th place from the top if placed among my job projects.
Also, my AntennaPod shows 196.9 hours of podcasts played since December 2021, and some of my friends report having sper
thousands of hours on video games.

And keep in mind that I use Emacs almost as extensively as it gets. You might as well spend much less time figuring it out for a more minimal use case. So, at least in my view, this weighs against describing Emacs usage in terms of sunk cost fallacy.

However, my story is consistent with the perception of a steep learning curve in the Emacs community. 19.3 hours over
days to get started is definitely a lot.