Unix as a tool forge

7 Nov 2022

programming, technology, emacs

Wikipedia[1] cites a few different sources on what "Unix Philosophy" is. Peter Salus summarizes it as:

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

That second bullet point is my favorite: making *composable* programs rather than monolithic systems. In this way, Unix is designed to be a forge for easily building new tools. The first rule—writing programs that do one thing well—is largely a means to the second. When you have building blocks that take simple shapes, you can compose them easily like Lego pieces.

I think that second goal is what makes Unix win: instead of providing you with every tool under the sun, you get a set of composable tools that allow you to construct better tools perfectly tailored to your problem. No one hacking on a PDP-11 thought to make an easy way to publish a blog like this one, but they put the framework in place to let *me* put together the tools I need to deploy this very post with a single command.

How Emacs fits inside of Unix philosophy

One might argue that Emacs goes against Unix philosophy, for it can quite literally do pretty much everything.[2] But that only violates the first rule—if you consider Emacs to be a tool forge, then Emacs is quite in line with the Unix philosophy. Emacs provides functions that all work on the buffer or bits of text, and these can all be composed to craft a work environment to fit your needs. I use over 100 different packages, and they all play nice together!

I have come to view Emacs as my primary forge. It's my layer on top of Unix, if you will. If I have Emacs customized how I like it, it doesn't matter too much what operating system lives underneath: I can get a lot of work done. I used to view Emacs just as a tool, and I used it exclusively as a text editor. As time went on, though, I began to value the extreme keyboard-centric control Emacs gave me over my system. That's why I moved from the terminal to the GUI version of Emacs: I wanted to have more modifiers available to bind functions to.

Many people use Emacs exclusively as a text editor, and that's fine. Usually these people have gotten comfortable with the command line, which is just another kind of tool forge. The great thing is both places make building new tools easy. Whatever your toolkit (though I do recommend you add Emacs to it if it's not already there!) make sure you can build new tools with ease.

Further Reading

- Discussion on Hacker News
- A kind chap sent me a link to this blog post as well as their own thoughts, which seemed like good things to link to.

Footnotes

[1]

https://en.wikipedia.org/wiki/Unix_philosophy

[2]

Emacs once ran Germany's flight control software. Please don't try this at home. https://www.reddit.com/r/emacs/comments/lly7po/comment/gnvzisy