

computers should be liberating

In U.S. Constitutional law, there is a distinction between **civil rights** and **civil liberties**. A civil liberty is a positive guarantee, a freedom **to** take action: the liberty of freedom of speech, of freedom of assembly. A civil right is a negative guarantee, a freedom **from** discrimination and mistreatment.

Much of open source is focused on liberties: the liberty to modify your computer, to publish your changes. There is little focus on rights, for example: the right not to have your time wasted; to not lose your work; to be able to control your own data. Most of all, the right to be respected by your computer and the people who program it ^[operators].

I want to imagine a world where these rights are protected and guaranteed. I have been doing so for the last year. Let's imagine, together.

In this world, all your work is constantly autosaved. A crashing program or browser tab doesn't lose your work, because you can restore it to the instant before it crashed ^[persistence]. Deleting any file is reversible, like Recycle Bin. You debug programs by playing them forwards and backwards in time ^[tomorrow].

In this world, the computer cannot take any action that you do not explicitly request ^[audacious]. It can only access resources you give it ^[capabilities]. There is no hidden state ^[ghosts].

In this world, you can talk directly to your friends, without worrying about the jurisdiction in which your data is hosted ^[tailscale]. Discord cannot read your DMs.

In this world, there is no distinction between "programmers" and "users". "Writing a program" is the same as using your computer ^[terminal]. You have the right to repair any program, and there are "software mechanics" the same way there are car mechanics. The concept of "sideloading" disappears; *all* pro-

grams are sideloaded, and there is no difference between a "self-signed" and "verified" build. Companies no longer have a natural monopoly over their software.

In this world, software builds are cached, globally, for every change. Rebuilds are nearly instant, even for enormous programs like a browser; there is no such thing as a "full" build, and builds are easy to run on commodity hardware. Each compiled program has metadata tying it back to its source ^[CTF], and modifying it is trivial ^[Pharo]. "Exporting data" is merely a matter of editing the program to print its data structures.

In this world, computers can be embedded in a place ^[Dynamicland]. Writing programs does not require "learning code", because programs are objects you can pick up and manipulate with your hands. "Sharing code" is handing an object to the person next to you. Computers adapt to humans, not the other way around.

This is a big dream. But in order to create radical change, you have to dream big. You have to move the whole design space at once, so that it coheres ^[coherent]. You have to know what your ideal looks like before you whittle it down to something that's possible to work on incrementally. The design will change as you work and discover more, and that's ok.

I want to build this world. I want to build systems that respect their operators. I want to build computers that aren't just a screen, but as much a part of the real world as a watch or a pencil and paper. I want to build tools for thought, for art, for fun ^[fun], that are *chosen*, not just imposed as an obligation ^[procrustean]. Most of all, I want to build this on top of the tools, apps, and programs people already use today, without needing to adopt radically new workflows.

I hope you will join me.