

Grieving the loss of details

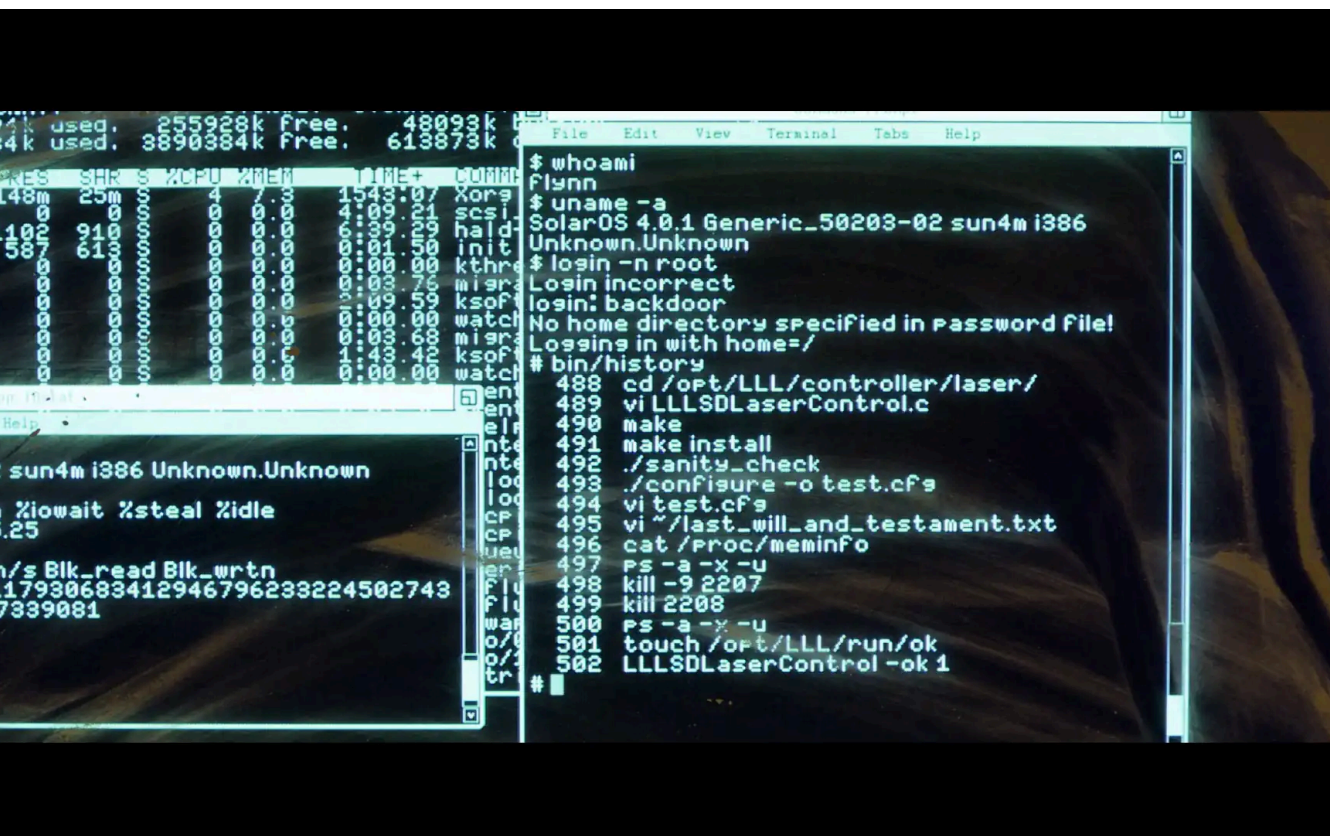
September 24, 2026

I've been thinking about where I stand in respect to the current state of the industry. This is more of a journal note than a post, apologies for that. I usually leave this stuff private, but I thought I'd post this as a trial in case someone relates to this experience and feels less alone. All opinions are my own.

Before the term “vibecoding” was coined and the industry shifted to valuing programmers primarily for designing architecture, I proudly called myself a coder as opposed to an engineer. In my eyes, this highlighted the way I focused on the details, performance optimization, knowing the intricacies of the language, and being able to explain how things work, as opposed to juggling Java-esque abstractions.

Of course, this was partially a misconception, but I can't help but notice how the opportunities to use my strongest sides are getting away, and the industry is switching to a model my mind is incapable of working with.

I first became familiar with computing when I saw the history scene in [Tron: Legacy](#):



10 year old me *needed* to know what these lines meant, and so I began the grueling job of learning programming. I didn't care about practicality, building useful programs,

or writing code per se: all I wanted is to understand *how the machine works*. It took several years of learning and maining Linux until I figured it out, but I got there.

Later, I tried specializing in different areas, like networking, cryptography, or Rust, but I always found myself gravitating towards the machine itself. I learned to appreciate it and respect its wishes. I found joy in writing a toy OS, counting cycles, and writing machine code by hand. I took pride in inventing clever hacks. Like a fisherman might feel one with a fishing rod, I treat the machine as a continuation of myself.

I've been doing this for eight years now, and I'm way in over my head. I hesitate to switch CPUs because I don't understand ARM64 as well as I understand x86, and that genuinely gives me discomfort. I worry about allocations in Python code. I'm deeply concerned about not knowing how to look at JIT disassembly of Java code running on my PC. Today I found [an annotated version of The Story of Mel](#) and caught myself thinking, who could possibly need "hexadecimal" explained?

I don't see myself as a programmer who is also an expert in a niche area; I see myself as *exclusively* a low-level coder, or even just a detail-oriented person. I can make a website if I have to, but I can't work on it for a month straight the way I can on low-level software – but I can very much research physics the same way.

My focus on details doesn't end on technology. I can't learn topics top-down or deal with absent information. When someone explains a concept to me, I need to reinvent it myself from scratch until I "get it". In school, I would spend hours adjusting axioms for my intuition until they fit, and then building theorems on top of them every time I needed to use them, until they, too, became intuitive. (Hell, my and my girlfriend's quality time is often proving theorems.) Closer to reality, I can't experiment with cooking without understanding the underlying chemistry. Generally speaking, I can't use something I don't understand completely – if it doesn't click, I can't work with it.

This is genuinely debilitating in day-to-day activities, but for a while, the software world was the one reprieve I had from this struggle.

Even those who didn't *get* would at least respect my work on low-level projects. There was an understanding that minor optimizations in compilers compound, that someone needs to write assembly for JSON parsers to go brrr, and that programs don't have to suck the way Electron slop does. Maybe it wasn't the central part of software development, but it was important enough that investing some time in it was considered worthwhile. In other cases, I built reputation by *knowing* stuff, the way I could glance at people's issues and know what went wrong.

That lasted until powerful LLMs came out. The overwhelming consensus across the industry is that LLMs will make such a draining and exhausting process as "worrying about details" obsolete, and let developers focus on abstractions and large projects. Great for them, but I'm not exaggerating when I say they took everything from me.

If anyone can point an LLM at slow code and it automatically finds a hot loop and uses a trick it found somewhere on the 'net to vectorize it, there is little point in hiring someone with a focus on that. If an LLM can analyze the code and explain pointer provenance to you and babysit you until you get it, there is little value in expertise.

I cannot use LLMs this way. I get almost physically ill when I work on a project with more source code files or at least general architecture than I can keep in my head, so I don't get anything out of expanded scope.

Last time I tried using an LLM for a pet project, I realized that the LLM would come to know more about it than pretty much any person I could show it to. I can be content with being a misunderstood genius, or whatever – but being not just deemed unworthy of human interaction, but effectively shown the door and redirected to an LLM is beyond insulting. I swore not to use LLMs for anything I care about since then.

Over the course of a year, I went from having a planned future to worrying about applying for disability. LLM-driven development optimized out every part that makes programming bearable for me, the job market decided it's the future, and now that the one option in my life I thought constant disappeared, it turns out there there was never a decent alternative.

That's where I stand for now. Aside from big tech, which is greeting LLMs with wide arms, few companies have resources to care about the type of work I do. Linux had a chance to be an exception, but oh well. Hobbies still exist, but they don't pay the bills, and increasingly retrocomputing and performance optimization are being infested by LLM fans that seek recognition rather than the experience, which sucks the fun out.

Made with my own bare hands (why.)