A cover letter to regular writing

Ka Wai Cheung, December 2023

I guess you become a professional writer the same way you become a professional poker player. One day you just decide this is what you're going to spend your waking hours doing. Both have equally poor chances of positive financial outcomes to boot.

But this is what I am committing to do now. I have been a writer of another kind for a quarter century—a software writer. Some will call this programming or coding (even programmers and coders call it that) but I prefer "software writing". The part I love about creating software is the *writing* of it. And just like the more common form of writing, software writing has two main components.

First, there is the thing you are writing about—your subject matter. You are making a game or an email client or a project management tool or a whatever. You are writing about *something*. Second, there is the way you write it. An infinite set of ways to describe the thing you are trying to build—your style.

It is hard to write code that makes a thing do what you want it to do at *all* times. This is stressful. You are doing your best to build a system that can accept a variety of inputs which create certain states of your system, the permutations of which quickly grow into the millions, then in a heartbeat, into the billions. Your job is to ensure the system handles each set of permutations correctly.

The way you write (your style) plays a large role in how well you can satisfy these permutations. And from this comes another form of stress: The opinions of others. Heated battles about *how* to write code usually devolve, on the fringes, into binary opinions. If you aren't doing this (or you are doing that), you are doing it wrong. Everyone is wrong to someone.

Software also has no finality. It is in a never-ending edit state. Something you've written must be maintained, fixed, patched, updated, perhaps rewritten. You can write something today that just broke something you wrote ten years ago. You carry with you the baggage of all of your previous works. This is why the word legacy only seems to have bad connotations when you're referring to software.

These stresses have a way of slowly catching up to you. As much as you might think you are doing great work (and you probably *are*), it never feels like you really are—at least that's how it's felt to me lately. And I am very much like most programmers, simultaneously egotistical and fragile. Set in my ways but fearful of the disapproval of others.

Professional software writing has—in my opinion—gotten harder all the while it really should've gotten easier. The new ways of writing code—the fashionable tools, frameworks, libraries, and architectures—require so much more of everything than the old ways. More configuration. More complexity. More memory. More electricity. But I don't believe they've actually made things better. The important part of technical progress seems like the speed of change rather than the direction. Or, I'm just getting older.

But even the audience needs a technical understanding today levels above what we needed a few decades ago. A very long password isn't enough. The 6-digit code you just got texted isn't enough either. It's a whole lot of work just to get past the front door to a place you're not even that interested in going to. Meanwhile, I can still open a book to page 3 the same way a reader of the Diamond Sutra did a thousand years ago.

I love this profession. I have given it everything but I'm getting too surly with it. It is my 25-year old child that just needs to leave the house already. And maybe it's getting a bit tired of me too.

Writing (the conventional form) surely does the same thing to great writers. But I am ready for the slow march toward this kind of exhaustion because, like software writing, there are immense bounties to be had in the march. Not necessarily the financial kind (though there can be plenty there too) but the intrinsic ones. Those are the ones you start caring more about as you get to a certain age.

Besides, the tools around regular writing *have* gotten better. As recently as seventh grade, I was still typing school essays on a typewriter (albeit an electric one), and retyping one pagers several times an evening just to correct a few couple punctuation errors. It just seemed like the lot of being a writer back then.

Today (and for quite some time before today) I can open up a document and just start typing on my laptop. I can close it up in an instant and come back to it sometime later to continue my unfinished thought. I can sketch an idea that's popped into my head with the phone that sits in my pocket (and this happens at least 3 times a day). Maybe a sojourn into this place will remind me of what was great about the other place. And with all of this new technology, a decent pen and weighty paper still can't be trumped.

I am looking forward to not having so many rules. Grammar is a framework, not the law and we can appreciate the writer who treats it merely as a suggestion. We can write about all sorts of subjects and have all sorts of opinions and it is often the way in which we describe those subjects and frame those opinions that the audience appreciates. That's a challenge that feels enticing to me. As for the detractors? Software writing has prepared me well for that I think.

I am looking forward to *finishing* things—not the finishing part so much as the not having to maintain part. I've written hundreds of essays about software writing before and the wonderful thing about those essays is they remain unaltered and yet they are still doing what I want them to do.

Now to make this a living? I have no idea where to start. Then again, I didn't know where to start 25 years ago either. I will need some faith, youthful optimism, and a lengthy stay on my wife's health insurance plan. And after this, I guess I just need a chip and a chair.