Issue 022: The Cloud

Graham Lee

July 6th, 2020

Welcome to the twenty-second issue of *De Programmatica Ipsum*, dedicated to the subject of *The Cloud*. In this edition:

- Adrian argues that the PC was an accident in history[1], stuck between two chapters of the same story.
- Graham explains how the self-fulfilling (and mythical) prophecy of a "world market of at most 5 computers"[2] has become true.
- In the Library section[3], Adrian will talk about three major books written by Brian Kernighan[4].

Enjoy this issue! Please subscribe to our free newsletter[5] to stay updated about new releases, or contribute[6] if you would like to support our work.

Cover photo by C Dustin[7] on Unsplash[8].

[1] https://deprogrammaticaipsum.com/somebody-elses-computer-as-a-service/
[2] https://deprogrammaticaipsum.com/five-computers/
[3] https://deprogrammaticaipsum.com/category/library/
[4] https://deprogrammaticaipsum.com/brian-kernighan/
[5] https://deprogrammaticaipsum.com/newsletter/
[6] https://deprogrammaticaipsum.com/contribute/
[7] https://unsplash.com/@dianamia?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[8] https://unsplash.com/s/photos/cloud?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Somebody Else's Computer As A Service

Adrian Kosmaczewski

July 6th, 2020

On November 12, 1990, a 35-year-old Bill Gates introduced his "Information at your Fingertips"[1] concept during his keynote at COMDEX. The PC, he said, would become "more personal," integrating "fax, voice and electronic mail," and providing "easy access to a broad range of information." More or less at the same time, Tim Berners-Lee was giving the final touches to the first web browser and web server software ever put in production. Bill Gates was right; as any Internet user can easily confirm 30 years later, PCs become much more interesting when you connect them to other computers. Or, as they call it, "The Cloud."

More crunchy anecdotes about Bill Gates; his 1995 book "The Road Ahead"[2] contained almost no reference to the World Wide Web. Make no mistake, he quickly caught up and merely three years later his Internet Explorer ate Netscape for breakfast, becoming a nightmarish standard for the following 12 years.

The Browser is the window that allows you to peek into The Cloud. How appropriate to name your operating system "Windows" then. Marketing genius at work.

But browsers talk HTTP, and HTTP sits on top of TCP/IP; let's remind ourselves that back in the 80s not everybody agreed in a protocol to make computers talk to each other. Standardisation took some time. During the 70s and the 80s, then, each vendor had its own "Cloud" computing platform, absolutely and completely incompatible with one another. IBM created OS/360[3], which later begat The Mythical Man Month[4] and z/OS[5]. The Mainframe and the terminals. The Cloud, as conceived in 1965.

Where does this phrase "The Cloud" come from? The origins of the word are shrouded in mystery. Legend has it that the symbol used in network diagrams[6] begat the name; others tried to claim the trademark[7] "cloud computing" somewhere in the 90s; others say Compaq[8] came up with the idea. The origin matters not, for the writing was on the wall: the PC was going to evolve into the window to a larger world, a network.

"The Network is the Computer®™" said John Gage[9] from Sun in the 80's. Yes, ® and ™. Because this phrase has been recently trademarked again by Cloudflare[10], because it is still valid, it is still relevant, and actually not much has changed in 35 years. (And Oracle's lawyers must be really busy suing Google[11], to have missed the opportunity to renew the trademark of such an iconic phrase!)

The Cloud is that which continuously changes shape, moves from place to place, sometimes disappear and sometimes blocks the light of the sun. Not as radically as Eclipse[12] would, but

[1] https://youtu.be/uGA1Chm_8RE?t=3026
[2] https://en.wikipedia.org/wiki/The_Road_Ahead_%28Bill_Gates_book%29
[3] https://en.wikipedia.org/wiki/OS/360_and_successors
[4] https://en.wikipedia.org/wiki/The_Mythical_Man-Month
[5] https://en.wikipedia.org/wiki/Z/OS
[6] https://www.cnbc.com/id/43483060
[7] http://tsdr.uspto.gov/#caseNumber=75291765&caseType=SERIAL_NO&searchType=statusSearch
[8] https://www.technologyreview.com/2011/10/31/257406/who-coined-cloud-computing/
[9] https://en.wikipedia.org/wiki/John_Gage
[10] https://blog.cloudflare.com/the-network-is-the-computer/
[11] https://en.wikipedia.org/wiki/Google_v._Oracle_America
[12] https://twitter.com/sburlot/status/5199586622

still.

It is a very religious thing. Businesses swear by it, developers can invoke it, pray for things to happen, and receive messages from it. I am surprised the church has not yet assigned a saint to our profession.

Before the PC, legend has it that computers were huge and expensive and occupied whole rooms. Users would log in to those behemoths from small VT100 terminals[13], and run programs written in FORTRAN and COBOL. The output would come as a long paper trail out of a dot matrix printer.

After the PC, legend has it that data centers were huge and expensive and occupied whole rooms. Users would log in to those behemoths from small "Post PC" devices or "PC Plus" devices (the names matter not)[14], and run programs written in PHP, Go, JavaScript, Java, C#, or whatever you can fit inside a Docker container. The output would be stored in a database, in a PDF file, or sent as a push notification to one of the aforementioned devices.

At the time of this writing, you can get free access to a mainframe running COBOL on z/OS[15] following this link[16]. You can then open up your copy of Visual Studio Code, add a few plugins, and voilà! You can start coding COBOL programs. Then you can submit those batch jobs to the mainframe, and they will be executed in the mainframe. Thanks to Visual Studio Code, you can even (shock!) debug your programs and inspect the state of the variables.

At the time of this writing, you can get free access to a cloud provider such as AWS. You can then open up your copy of Visual Studio Code, add a few plugins, and voilà! You can start coding in pretty much any language that can be bundled in a Docker container. `kubectl apply` those containers into your nearest Kubernetes cluster, and they will be executed in the cloud. Thanks to Visual Studio Code, you can even (shock!) debug your programs and inspect the state of the variables.

What has changed is the availability of small and cheap terminals occupying whole pockets. Programs running in The Cloud inside some virtualization solution. All based upon SaaS, itself based on PaaS, itself based on IaaS, and so forth until the end of the abstraction layer stack. Maybe there are also a few more unit tests somewhere than there used to be in 1974, but I would not hold my breath. There are definitely more standup and retrospective meetings, that is for sure.

Oh, and God forbid, you can go "serverless" now, whatever that means.

What has not changed in 50 years is the fact we are still using centralized architectures, prone to government intrusion and privacy leaks. Maybe it is time to think about a "Post Cloud" era (or a "Cloud Plus" era?) where information is distributed instead of centralized. Of course this raises questions of trust, cryptography, security and collaboration, but the technology to build such systems already exists. It is more of a question of policy and education than of technology. It is actually a question of sustainability[17] and protection of the environment[18], too.

[13] https://en.wikipedia.org/wiki/VT100
[14] https://www.theverge.com/apple/2012/7/12/3151491/fighting-words-apple-post-pc-microsoft-pc-plus
[15] https://en.wikipedia.org/wiki/Z/OS
[16] http://ibm.biz/cobollabs
[17] https://deprogrammaticaipsum.com/choosing-a-programming-language/
[18] https://www.theguardian.com/environment/2010/apr/30/cloud-computing-carbon-emissions

We need decentralized systems; based on peer-to-peer algorithms, using distributed keys for security and free from corporate and government snooping. Because the big idea of the ARPAnet[19], as conceived by the US Department of Defense during the Cold War, was precisely to create a distributed system thanks to packet switching, resilient to nuclear weapon attacks. We have forgotten the core ideas behind the Internet. It is time to bring them back.

A word before ending this article. Since its inception, articles in this magazine use pictures from Unsplash[20] as illustrations. This one is no exception, but it is interesting for a different fact: it is labeled with the words "white Macintosh computer"[21]. Take a good look at it; this is no Mac, it is an IBM PC 5150, the eponymous PC, the one that represents nothing more than a hiccup between two incarnations of the same idea.

Cover photo by bert sz[22] on Unsplash[23].

[19] https://www.history.com/topics/inventions/invention-of-the-internet

[20] https://unsplash.com/

[21] https://unsplash.com/photos/Zd6PL6PSW5E

[22] https://unsplash.com/@bertsz?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[23] https://unsplash.com/s/photos/ibm-mainframe?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Five Computers

Graham Lee

July 6th, 2020

There is no direct evidence that Thomas Watson, Sr. predicted in 1943 that there would be "a world market for five computers". First, he probably didn't know about computers back then: in the 1940s computers were classified research projects at the UK Government Code and Cipher School and the US Army, while Watson's IBM were famously involved with the other side[1] during the war. Second, it's just not something there's evidence of him talking about. When IBM presented plans for their early commercial computers to various businesses, they got about ten immediate sign-ups and as many expressions of interest. That's already a market for twenty computers. Various people in government on both sides of the pond, including the UK National Physical Laboratory's head Sir Charles Darwin (grandson of the one you've heard of), may have said that their respective government's *laboratory* needs could be satisfied by a small number of computers, but these seem to be estimates of immediate resource usage rather than predictions of future market growth.

Like the 640k RAM thing attributed to Bill Gates, the five computers quote seems to be a fabrication from a later age, invented to teach us a lesson about the futility of grand predictions. It's a shame, though, because the five computers prediction looks like it may come true very soon.

Through a combination of adoption of virtualisation technologies (theoretically improving the portability of software deployments) and consolidation onto a small number of providers (practically ensuring that no such portability occurs), three of the five computers can easily be identified. Amazon, Alphabet's Google and Microsoft each run one of the world's three largest commercial computer-time rental businesses. Their three CFOs each centrally command an economy worth in the region of $1T (particular quotes taken from Yahoo! Finance

[1] https://www.huffpost.com/entry/ibm-holocaust_b_1301691

on 14th June valued Amazon at $1.27T, Alphabet at $0.96T, and Microsoft at $1.42T), though as we shall soon see it's probably best to see them as department heads of a single multi-trillion global computer economy encompassing the three businesses and more. To put that into comparison with other planned economies, the Soviet Union reached a GDP of $1T in 1979 and $2T before its dissolution a little over a decade later.

So which computers are fourth and fifth in the global market? Certainly none of the other cloud providers, who are all tiny in comparison. The closest runner is Alibaba ($0.58T), but forget the other western runners, Oracle ($0.16T) and IBM ($0.11T). They can both be seen as attempts to extract rental income from their existing hardware platforms – SPARC and RS6000/Power, respectively.

And talking of RISC CPU designs leads us to disregard another wannabe competitor who'd definitely *like* to be in the big five, but seems destined to come crashing back down to the ground. As excitement mounts over whether certain high-profile computer vendors may switch to using the ARM CPU design in their systems, ARM sales and revenue are down and parent company Softbank is suffering losses, too (both directly, and its $100B Vision Development Fund). ARM's heritage is in desktop and mobile computing (the chip was designed to run the Acorn Archimedes line of personal computers, and when ARM was spun out of Acorn they received funding from Apple to develop chips for the Newton line of handheld computers). This is clearly a market that current CEO, Simon Segars, sees as legacy business, the cash cow that he can squeeze to fund his real interest in the Internet of Things. Thus ARM's new goal, to enable "a trillion connected devices".

You could see Softbank's pattern of investments as an attempt to create that fourth computer in the market of five, the distributed computer of things. Whether you're in the office (Slack, We), at home (Grofers, FirstCry, OpenDoor), or inbetween (Cambridge Mobile Telematics, Mapbox, GetAround), Softbank are hoping that you're connected to Masa's computer. Their particular choices, and scattergun approach, mean that this is unlikely to coalesce within the funds and time available to Softbank. It's unlikely that they'd take ARM down with them before someone stepped in to buy it out. Which brings us on to the *real* contender for the world's fourth computer:

Apple. The Apple cloud is a computer that contains a lot of music, TV shows, movies, games, books, news reports, and storage space that users of Apple's consumer electronics can rent access on. As a cross-selling deal, a small amount of access is offered with the sale of each of the consumer devices. Unlike the other three computers we've seen so far (Amazon's, Alphabet's, and Microsoft's), you can't rent time directly on the Apple computer to run your own code. You can upload your own software, but you rely on people discovering and choosing to run it themselves. The execution is actually done on the end devices, not on the Apple computer, though the documents people create may be synced via the Apple computer. By the way, Apple's planned economy currently amounts to $1.47T.

You may notice that described this way, the Apple computer sounds a lot like the Ethereum computer, and expect me to suggest that it's the fifth computer, or maybe the bitcoin computer, but no. It's the Facebook computer. Like the four previously mentioned, the Facebook computer is a huge distributed computer with billions of users, running lots of different applications. Like the Apple computer, you can't just run your own code on the Facebook computer (well you *can* run games), but that doesn't matter, because most people use computers for the same reasons. The big feature of the Facebook computer is the contacts app, which has about a third of the world's population on.

Those are the five computers of the world market, together directly controlling $5.8T of the

world's assets, roughly 2/3 the value of the total amount of gold in the world. Indirectly, their influence through app stores, subscription fees, software providers who are dependent on their services, resellers and so on is significantly larger.

And it's important to consider them together, because the veneer of competing in an open market only goes so deep. Look at any relevant trade association and you'll see the trillion-dollar computer vendors working together toward their common interests, for example the Cloud Native Computing Foundation (Alibaba, Amazon, Apple, Softbank ARM, Google, IBM, Microsoft, Oracle, others), or the Linux Foundation (Fujitsu, Google, IBM, Microsoft, Oracle, Alibaba, Facebook, Amazon, Softbank ARM, others). There is really one central computing economy, controlling well over six trillion dollars of technology investment, with five different public faces.

How did the market for computers get consolidated to this extent? It certainly wasn't inevitable. Early examples of outsourced computing were advantageous because the hardware was so expensive, needed specialist and frequent handling, and there just wasn't much of it about. Those early IBM systems that Watson found more than five customers for could be rented for around $12k-18k per *month*, so you had to be sure that your computing job was going to save a few people's salaries before you even ran it in the cloud.

Where companies did invest in computing hardware, offering the platform as a service to others help to offset the purchase and development costs. The first commercial computer in the UK was the Lyons Electronic Office I, developed from the EDSAC machine by food producer Lyons. LEO/I also ran payroll for Ford UK, and weather simulations for the Met Office (until they bought their own Ferranti computer). It was in use between the 1950s and the 1980s, demonstrating great longevity in the days before planned obsolence.

During those decades, the smaller size of components, greater reliability, lower costs, improved usability and application combined with the business vision of key people who set out to provide "a computer on every desktop". Rather than a small number of computers becoming more capable, capabilities became more widespread, more distributed, more affordable, and more accessible. Computers shrank in size and cost until every office building, every cubicle, every home, eventually every pocket could contain one.

None of the technical, physical or practical trends described above have reversed to lead to the cloud computing consolidation we see today, and the world market for five computers. The only change that can account for this behaviour is the business vision, and the direction in which the cadre of executives take their companies.

Since the dot com crash of 1999-2001, investors have looked to seed companies with the smallest amount of money possible to lead to either a rapid collapse or rapid growth supported by multiple follow-on investment rounds until an eventual exit. Hockey-stick charts, "disruption", "MVPs", "growth hacking", "fail fast" are not neutral business advice, but a particular way of working designed to minimise risk and maximise payoff for a small clique of rich investors who expect the money they lend out to work its way back into the system – perhaps through cloud subscription fees paid to companies they hold positions in. Don't go buying expensive capital investments like offices, desks or computers when you can rent them...from us.

This has led to the situation where computing applications are architected with the goal of rapidly meeting the goal of tomorrow's investor presentation, regardless of the effect this has on the later costs or usability of the product. One workload I now support would cost an estimated £6000 per month to run on one of the large cloud providers, which would allow

me to purchase the actual hardware in use four times over every year. But "spinning up an instance" today using free credits means you can worry about that cost later, whereas ordering a box from Dell means waiting two to three business days by which time the hackathon is over. Besides, you'll hit your Series A round before you get that far, and can hand the money straight back to the cloud providers taking a few percent for foosball tables and sodas for your team of devops engineers.

Make no mistake, the VC money in software is a small raindrop let loose from the cloud, with the intention of finding the next valuable idea to absorb. A tiny morsel of this multi-trillion-dollar planned economy is sent out to see where it sticks, and what new ideas the computing Gosplan department should factor into their forecasts.

It's actually pretty easy to stay out of this system, by designing software applications that don't depend on it. Treat your users as your customers, and give them choices: do they want to use the facebook computer, or their own computer? Do they want their files in the Google computer, or the Apple computer, or maybe on a Nextcloud or in Dropbox? Does your processing have to be done on the Amazon computer, or could you buy and house your own machine, or colocate it, or use a virtual private server? Do the interactions of your customers with your software need to become part of the global technology planned economy, or can they be part of an online commons like Wikipedia, OpenStreetMap, or the Internet Archive?

Photo by Derek Thomson[2] on Unsplash[3].

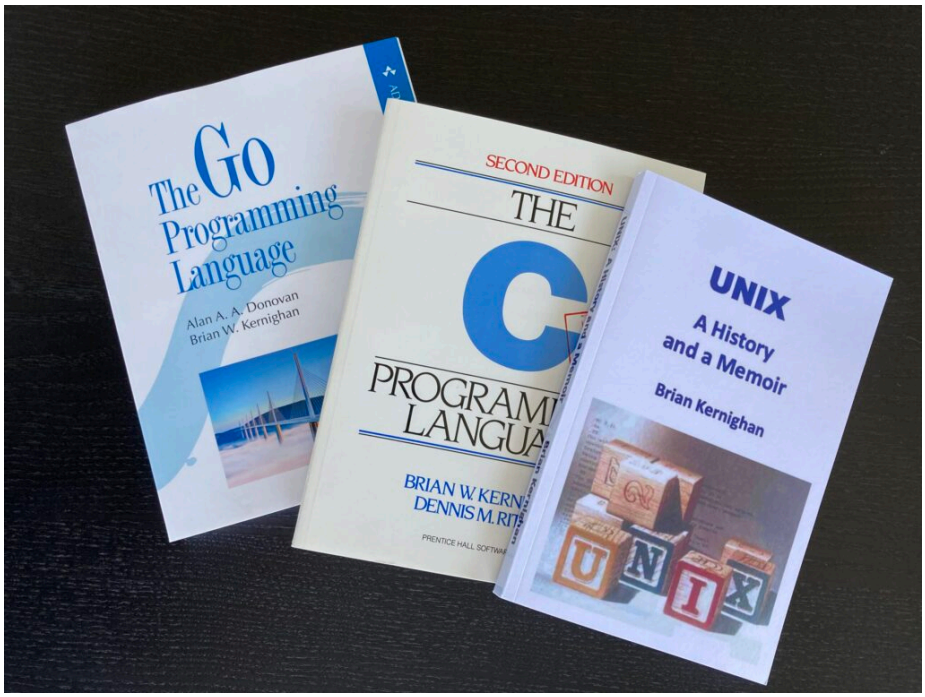
[2]https://unsplash.com/@derekthomson?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[3]https://unsplash.com/s/photos/storm-clouds?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Brian Kernighan

Adrian Kosmaczewski

July 6th, 2020



Of all the articles I have written in this "Library" section, this has been by far the most difficult to write of them all. It is extremely hard to summarize in a thousand words the major achievements of a person that has defined the way our modern world and our industry work, in the most unfathomable ways.

Because, to be honest, it is hard to find a starting place when talking about those who have such long resumés as Mr. Kernighan. Let us try to find some starting points and see where this takes us.

Let us pick one major milestone: this is the person who gave UNIX its name. The operating system architecture that ate them all[1], spreading into computers of all sizes. And the one that even inspired a large chunk of code in the Windows codebase, as Bill Gates himself explained at the Unix Expo in October 9th, 1996[2]:

> And through Windows NT, you can see it throughout the design. In a weak sense, it is a form of Unix. There are so many of the design decisions that have been influenced by that environment. And that's no accident. I mean, we knew that Unix operability would be very important and we knew that the largest body of programmers that we'd want to draw on in building Windows NT applications would certainly come from the Unix base.

[1] http://www.asymco.com/2010/09/29/unixs-revenge/

[2] https://web.archive.org/web/20011102031310/http://www.microsoft.com/BillGates/Speeches/industry&tech/uexpo.asp

We have mentioned Bill Gates enough times in this issue I think.

Brian Kernighan also invented "Hello, World!" programs[3]; his first was for the B programming language, direct ancestor of C. How would programming books look like without such introductions, anyway? That first "Hello, World!" program is available online[4].

Then there's AWK; the programming language that anyone using the "Unix way" in the command line, piping the stdout of a command into the stdin of another, has used at some point. The cornerstone of oh so many duct tape bash scripts that enable the modern world, in one way or another.

And finally, books. Mr. Kernighan is a prolific author, with a bibliography including major titles whose mere existence have defined our craft. The books chosen for this article show three major stepping stones: Unix, C, and Go.

First, let's talk about "Unix, A History and a Memoir," a short essay self-published last year. The word that describes it best is *humble*.

In spite of the tremendous privilege of having been part of the birth of a whole industry, Mr. Kernighan takes a step back. His writing is full of anecdotes, names of well-known and some less well-known visionaries of the industry, all sharing the same offices at Bell Labs. Anecdotes from daily life in the lab, including that of the various Italian espresso machines, or Bell Labs' work-from-home policy, alternate with technical details about the various tools and steps that make up Unix's history.

There are few locations on Earth where so many ground-breaking technologies have seen the light; if the Xerox Palo Alto Research Center is one such place on the West Coast of the USA, Bell Labs is definitely its counterpart on the East Coast.

The word "accidental" comes also to mind while reading these stories; one has the impression that Mr. Kernighan just *happened* to be there. It is hard to imagine that it all stops at that. Yet, that is the spirit of the text.

Second, there's "The C Programming Language." Having written the most popular tutorial about the B programming language, including the eponymous first "Hello, World!" mentioned earlier, it was only natural for Mr. Kernighan to write a tutorial for C. The book states at the very beginning that C is a "rather low level" programming language. Reading that phrase more than 30 years after the publication of the second edition (right after the ANSI standardization effort) it is hard not to smile. Most of us will agree these days to call C "portable assembly" without blinking an eye.

Yet, in the age of WebAssembly[5] and Emscripten[6], C has found a new niche, as the source language for a new generation of browser applications. Not bad for a 50 year-old programming language who inspired countless others.

Finally, "The Go Programming Language." Go is arguably the language of The Cloud[7]. Docker, Kubernetes[8], OpenShift, Prometheus, and most cloud-native tools sanctioned and approved by the CNCF[9] are written in Go. Some even say that Go is the new Ruby[10].

[3] https://deprogrammaticaipsum.com/memories-of-hello-world/
[4] https://web.archive.org/web/20150611114644/https://www.bell-labs.com/usr/dmr/www/btut.pdf
[5] https://webassembly.org/
[6] https://emscripten.org/
[7] https://deprogrammaticaipsum.com/issue/issue-022-the-cloud/
[8] https://deprogrammaticaipsum.com/antonomasia/
[9] https://www.cncf.io/
[10] https://00f.net/2019/10/28/go-is-the-new-ruby/

Go borrowed the curly brackets of C and achieved the unthinkable: to became popular in spite of lacking classes and inheritance. Or probably, because of that. The book itself is unsurprisingly close to its sibling for C described above; the structure and prose betray the common co-author hiding behind the keyboard, shaping the way programming book should appear and behave. It even starts with a "Hello, World!" program; right there, at the bottom of page 1.

Rankings might not mean much, but let us use them anyway. TIOBE[11] has named C the programming language of the year 3 times: 2008, 2017 and 2019, and at the time of this writing, it states that C is the most popular programming language. It also named Go the programming language of the year in 2009 and 2016, currently occupying the 12th step in the ranking. PYPL[12] puts a contraption named C/C++ (a language that does not exist but fulfills a precious place in the collective unconscious of our craft) in 5th position, with Go occupying the 13th spot. And RedMonk[13] places C at the 9th position, and Go at the 15th place.

Unix remains the basic environment for these two languages (and many others) to run and flourish. Unix, written in C, remains the operating system of choice for the smartphone in your pocket, running apps that connect to The Cloud, itself running apps written in Go, inside yet another Unix environment. It all ties together, somehow, as if Mr. Kernighan had unwillingly predicted and fulfilled the prophecy of a whole industry. All with humble, concise writing, always highlighting the work of others, and making technology as easy to consume as a "Hello, World!" program.

I have left aside "The Elements of Programming Style," "The Unix Programming Environment" and "The Practice of Programming." Not because I would not want to cover those three books, but simply because a thousand words is not enough. But I think the titles are enough to highlight the useless efforts of this author to find ways to transcribe their importance.

Cover photo by the author.

[11] https://www.tiobe.com/tiobe-index/
[12] http://pypl.github.io/PYPL.html
[13] https://redmonk.com/sogrady/2020/02/28/language-rankings-1-20/