Andrew Lison
Minimal Computing

Shintaro Miyazaki: So how would you describe or define minimal computing?

Andrew Lison: I come at this question from my experiences in IT, where it has long been commonplace to separate out user-facing applications from their back-end processes. Whether this server end is “thin,” providing a relatively simple set of data for a full-featured user agent to interpret, or whether it is “thick,” so that the client simply displays whatever the server delivers to it, what remains the same from an IT perspective is to focus on what is absolutely necessary for the server to do its job, and as much as possible restrict the server to those functions.

Minimal computing for me hearkens back to these concerns, and in this sense I have been a “minimal computationalist” for as long as computers have been able to multitask. As soon as you have programs competing for CPU time, memory, etc. the question of how to optimize your system for what you would actually like it to do quite inevitably arises. So as an IT professional I focused on having the bare minimum of software installed and/or active on production systems – for a variety of reasons, but boiling them down to three I would say performance, security, and manageability. Quite simply, systems run more smoothly, have fewer vulnerabilities, and are easier to maintain if they are not larded up with “helper” tools, add-on applications, and even GUIs, if we’re talking servers. But this philosophy extends to the way I tend to engage with desktop operating systems: less is more. In the Windows world, manufacturers tend to preinstall all sorts of vaguely-useful-at-best applications onto the systems they sell, so installing from clean media and being selective about what you choose to install on top of that can often dramatically speed a computer up.

So my definition of minimal computing is in line with what Alex Gil laid out in some of the earliest digital humanities writing on the topic:[1] an emphasis on the sparsest of configurations necessary for a given task or set of tasks at hand. This might seem like a rather technical definition, so let me now embellish it with some more conceptual considerations. What I have said above comports, I think, with about half of what is laid out in influential analyses of minimal computing building on Gil’s from people like Jentery Sayers,[2] while it also shies away from the other half, especially the focus on minimal obstacles to understanding or technical jargon. The reason for this is simple: lowering these barriers depends on a certain amount of abstraction that is purchased with computing power itself! In other words, there is a contradiction, or – what is probably a better way to describe it – a *dialectic* between machinic and human effort such that the less of one you employ the more of the other is engaged.

The consequences of this antinomy are profound enough that it is currently the subject of my first monograph, anticipated by a piece I published in the journal *Configurations* about eighteen months ago.[3] It is my contention that we are approaching a reckoning with the proliferation of computing power we have come to take for granted over the past forty years or so, an explosion that can be dated back even further to the 1965 formulation by Intel Corporation co-founder Gordon Moore's prediction-turned-"law," namely that, for the same price, the number of transistors on an integrated circuit will double every twelve to twenty-four months.[4] The historical anomaly of exponential technological growth represented by Moore's law is a special case of the more established relationship between fixed and variable capital identified by Marx: the tendency of capitalism to replace people with machines, despite the fact that only people can add value. We have been living so imperceptibly in this environment that it has largely, with a few notable exceptions,[5] shaped our discourse in media theory and the digital humanities about what computing is, what it is for, and what kinds of justice we ought to seek from it. So, for me, minimal computing is in some sense a practical return to the materiality and specificity of a medium that '90s new media theory defined – at the inauguration of the field and as it became clear that the internet was more than a passing fad – primarily in terms of semiotics. From this starting point, the impulse to translate its operations into easier and more accessible forms is understandable, but it can only proceed by ignoring the computational cost of such translations, a luxury we can no longer afford as Moore's law itself is in the process of running out.[6] So, for me minimal computing is or at least can be a variety of practical ideology critique, just as the book, tentatively titled *100% Utilization: Computation and Labor after Moore's Law*, is a theoretical version of the same.

Shintaro Miyazaki: How is minimal computing different from let's say supercomputing or feminist computing?

Andrew Lison: According to definitions offered above supercomputing might qualify as a kind of "minimal computing"[7] in the sense that whatever computing power is directed toward the tasks a supercomputer is designed to address is probably being done so with extreme parsimony since the cost is already so great! If you're concerned with the limits to and material foundations of computation then naturally supercomputing will be of interest to you as well. It's also an area of extreme secrecy; the commonly-accepted lists of "most powerful supercomputers"[8] of course cannot include those we don't know about – that is, those that are under the control of the world's most secretive state security agencies and almost certainly directed, among other things, toward breaking the encryption that their adversaries (and the public) use to communicate confidentially. Nobody knows how powerful these are (or, more accurately, those who do aren't telling), but it seems safe to say that there probably exist computers more powerful than the ones we *do* know about, and, further, that whatever hardware advantage this brings is coupled with similarly secret software/ algorithmic advances.[9] So this is an area where, by definition, you are unable to do anything more than you have to in order to solve these incredibly complex problems because the task already strains the limits of the hardware. In these applications and other performance-intensive algorithms like Bitcoin mining, there is an inbuilt drive to reduce the "socially necessary computing time" as much as you can, since the computations that need to be performed are already so intense as it is, and this, perhaps paradoxically, aligns these processes with a kind of minimal computing.

But for practical purposes, minimal computing principles can be explored on much lower-end hardware. This is perhaps more in line with DH definitions suggesting there is much in mainstream computing that you don't "need" to get the job done. Despite the ongoing slowdown in Moore's law it is still remarkable how much you can do with a very cheap single-board computer, and the DH minimal computing community from Gil's think piece on have correctly looked to systems like the Raspberry Pi[10] as a part of their vision. Working with this type of hardware can help really bring into focus which aspects of the computational experience are easily addressable with comparatively low-end equipment and which require more computing power to tackle. In this regard I think minimal computing aligns well with other movements like feminist or queer computing that focus on re-envisioning the experience of and access to computation and its culture along more equitable grounds, since the purpose of these systems and the open-source ecosystem that surrounds them is to encourage a wider range of participation in computer science specifically and computing more broadly. And I can say, as someone who grew up at a time when getting one's hands on a C compiler would have been a difficult task, that the range of options available to someone just starting out today is nothing short of breathtaking! For feminist computing in particular I think the key role of women in the historical development of programming from its very inception connects quite strongly with a back-to-basics approach.

That said, there is something unique to minimal computing, at least for me, that might not be easily subsumed into either of the above or similar terms, and that has to do with an attention to not simply the intensity or form of computing but an emphasis on those elements vis-àvis the material components of computation. It is not that long ago, really, that programmers, especially programmers of desktop software, could write in assembly if they so chose, or their needs so demanded. I have always been struck, for example, by Jimmy Maher's description of bare-metal Amiga programming in his fabulous *The Future Was Here*,[11] and while the world of contemporary computing is so far from that kind of practice that not only our own user-facing operating systems but the "invisible" operating systems that often subtend them[12] make it a demanding if not impossible task, we do have access to this lower layer through emulation as well as via even more basic microcontrollers like the Arduino and Raspberry Pi Pico.[13] So, to me, minimal computing is an ethos that is interested in and appreciative of this kind of baseline functionality even if it doesn't always engage with computing at this level. Indeed, rather than being a lost or irrelevant art in today's computing landscape it is my contention that such skills are only going to become more salient as Moore's law peters out and we can't rely on exponential computing power growth to mask the cost of more "user-friendly" implementations any longer! This, of course, poses profound questions of what access to computing skillsets looks like in an era where complicated, low-level programming may be the most realistic way to eke the last performance gains out of stagnating hardware.

Shintaro Miyazaki: What can minimal computing look like in practice? What are some things that people can do with it and/or with its principles in mind?

Andrew Lison: This will be a long answer as there are a lot of things one could do to experiment in this area! Others have covered some of the more straightforward applications, say, generating documents or creative coding,[14] so I will focus on my own inclinations as a former network administrator and historian and theorist of computing.

I will start by echoing the platform recommendation I made above: although a little difficult to get a hold of at the moment with the ongoing semiconductor shortage, the Raspberry Pi[15] is a phenomenal device that provides a great sandbox for those looking to get started. It runs a variety of operating systems and configurations,[16] but its core system software, the Raspberry Pi OS (based on the Debian Linux distribution[17]), has recently made dramatic improvements, converting to a 64-bit base and maturing out of some of the initial design decisions, like a default username, that to my mind made it a little less suitable as a peer to, or replacement for, a "regular" computer. It uses very cheap SD card media[18] as its storage base, which means you can swap out operating systems easily. Just putting in a different card can give you an entirely new computing experience, and this modularity also comes in handy if you are afraid of messing something up – just re-flash or replace the card with another one and you're back up and running! There are multiple different options, each with its own use case: for example, the “lite” version of RPi OS, which is the command-line-only version, is perfect for servers; the desktop version of the OS, which includes a GUI window manager capable of running modern-day office suites and web browsers; and derivatives like RetroPie, which is an emulation-focused fork ideal for retro gaming.[19] And those are just the ones built on RPi OS – there is also the more user-friendly Ubuntu Linux distribution, which now has a Raspberry Pi version,[20] but since RPi OS made the improvements I noted above it hasn’t seemed as necessary.

With any of the above systems installed you have access to pretty much any open-source software that works with ARM processors. As I said, the lite version is great for a server, and also to get yourself familiar with using the Linux command line[21] (this book is slightly outdated with the changes I mentioned earlier but still a solid place to start). You can, for example, install a file server like Samba to store media and/or a web server like Nginx or Apache to serve HTML and/or web applications.[22] There are, of course, more full-featured OSes that will allow you to do many of these same things,[23] but the spirit of minimal computing is such that building your own install from the command line up is a great way to learn more about these lower levels of computing, to understand the way that UNIX derivatives are based largely on text-file configuration, and to familiarize yourself with the key, if somewhat wonky, tools to administer things at this level (as one of my first UNIX teachers drilled into me decades ago, you should be at least passingly familiar with vi[24] because you can pretty much count on it always being installed on any system you need to work on – he was right, but I generally prefer nano,[25] which these days is almost as ubiquitous).

With that kind of Linux base and the specific educational focus of the Raspberry Pi Foundation, you also have access to some of the most powerful and time-tested programming languages, from modern-day favorites such as Python to foundational software like the GNU Compiler Collection.[26] This is particularly useful to those interested in the history of computing; you should be able to work through classic texts like Kernighan and Ritchie’s *The C Programming Language*, more modern introductions, and even a personal favorite of mine, *Structure and Interpretation of Computer Programs (SICP)*,[27] which for a long time formed the basis of the MIT introductory computer science curriculum. *SICP* uses a derivative of Lisp known as Scheme which has often been a little difficult to get working on modern computers.[28] Right now it does not run on Macs, for instance, and while it’s not yet in the RPi OS repositories (I expect it to be when the next version of Debian is released) it is relatively easy to compile, which is in itself another exercise worth learning how to do for those interested in minimal computing! Being able to transform an open-source project into executable code yourself is an excellent first step toward writing or analyzing software on your own. If you choose to experiment with programming on the GUI version of the OS rather than the command-line "lite" version you will have access to built-in text editors like Geany as well as even cross-platform, commercial, proprietary options like Sublime Text,[29] so you don't have to sacrifice usability to experiment with these languages if you don't want to. The Pi's implementation of Java also supports the tools that accompany fascinating texts like *The Elements of Computing Systems*, which models the construction of computers in software from the logic-gate level up to a working application.[30]

While Windows probably still takes the crown in terms of emulation just in terms of the range of software available, the Pi is also a great system for exploring this area, one of great interest to both game scholars and historians of computing. I myself used a Raspberry Pi running the Atari ST emulator Hatari[31] for my recent essay on the 1990s musical composition software Cubase.[32] I've since become interested in the late 1980s and early 1990s precursor to modern-day macOS, NeXTSTEP,[33] emulatable using a program called Previous, which borrows from Hatari and is best compiled from source.[34] RPi OS can also run the classic Mac emulator Mini vMac,[35] which I also used to examine competing software for my Cubase article, and of course RetroPie has you covered not only for a wide variety of gaming consoles but also many early computing systems as well – I've had great fun playing around with these recently-translated versions of Atari and ZX Spectrum games made in state-socialist Slovakia,[36] for instance.

There are also, I think, other applications that are minimal in spirit if not in execution. For example, I've enjoyed learning the basics of 6502 assembly, the low-level language for programming the MOS Technology CPU used in the Atari 800, Apple II, Famicom/NES, and a host of other '80s computing platforms, through this online tutorial,[37] which makes use of JavaScript in order to provide a complete, if limited, sandbox for writing, debugging, and executing the code. While on the one hand the architectural requirements are somewhat high – a modern browser/OS that can execute JavaScript, etc. – on the other the environment being facilitated this way is a simulation of the bare-bones world of low-level personal computer programming. Projects like this mean that people can get started learning without having to configure a toolchain for compiling their code, figuring out what kind of system or emulator to run it on, and so forth, which can be daunting for the uninitiated.

Shintaro Miyazaki: Is minimal computing, then, just about retro or "old" computing?

Andrew Lison: Absolutely not! In addition to the server examples I gave above, which run up-to-date versions of various protocols the same as you would find on any server-class Linux box, over the past year or so I've become fascinated with the Gemini protocol and ecosystem,[38] which is essentially an attempt to update the venerable pre-web Gopher protocol[39] for the 21st century, developing something more complex than that protocol but much less so than the web. Minimalism is absolutely at the core of its concerns, but in a way that is as much forward-looking as it is hearkening back, and I think the trade-offs this entails are pretty interesting.

For example, Gemtext, the markup language that Gemini uses,[40] is extremely compact. In contrast with the huge variety of tags available in HTML, aside from writing plain text there are only about five things that you can do: link, make headers, make unordered lists, quote text, and turn the formatter off (to display code, ASCII art, or anything else where the unformatted placement of characters matters). This makes it a snap to learn, like similar minimal computing favorite Markdown.[41] Yet because UTF-8 encoding is the baseline format in the protocol spec, the entirety of the Unicode character set can be used. So you actually have a fair amount of graphical options, including emoji ✅; what you lose are stylistic options (colors and fonts, particularly, as well as image placement, though images themselves as well as sounds are supported) and, perhaps more importantly, *semantics*. In other words, you can write in bold or italics,[42] but these letters are not "understood" as the same letters as their non-bolded or non-italic counterparts; they are quantitatively different and e.g., won't show up in a search unless you are searching for those specific glyphs. So it's fascinating to compare this mode of engaging with information with that of the World Wide Web. I'll confess that I miss some of the semantic options in HTML – which was, let's not forget, designed originally by academics and in many ways ideally suited to an intellectual, if perhaps STEM-y, style of writing. For example, there is no straightforward method to do footnoting the way there is with HTML anchors, and combined with the other semantic issues I already noted it is difficult to imagine Gemtext becoming a scholarly *lingua franca* in the way that 1990s HTML was. When taken on its own terms, however, you can see how it is trying to do something that is in some ways quite new.

Similar compromises can be found in the protocol's technical details. Gemini traffic is encrypted by default (something which we are only now getting around to with HTTP), but the designers have eschewed the thorny, hierarchical model based on Certificate Authorities that governs the secure web in favor of what is known as the "Trust on First Use," or TOFU, model. Essentially what this means is that instead of looking for a signature to verify your encryption is valid, Gemini clients assume a site is legitimate the first time you ever connect to a server, and only warn you if the certificate changes later on. This is somewhat controversial, since encryption security is a very complex topic and you wouldn’t want to run, say, a Gemini version of Gmail like this, but it does make it easier to get started; you can spin up a server[43] with a self-signed certificate and clients won’t raise a warning like web browsers do. And understanding the pros and cons to either approach will illuminate a whole range of interesting theoretical questions about the nature and usage of public-key cryptography.

More broadly, the Gemini community embraces the idea of a minimal approach to networking and knowledge transfer. I was fortunate enough, especially given my own political leanings, to get what was at the time the last spot on an intentionally small community known as the Mare Crisium Soviet Socialist Regency,[44] which in addition to hosting Gemini sites, or “capsules” such as mine[45] runs a bulletin board only accessible for users with accounts on the server. With the appropriate and obvious caveats about small-scale solutions to large-scale problems, it has been really interesting to see this community and ethos develop. If you are interested in exploring it further, I would recommend you examine some Gemini clients (which you will need to open the above URLs). The most full-featured and aesthetically well-designed browser I have found is a very actively developed cross-platform program called Lagrange that offers binaries for most major operating systems, including the Pi, but also compiles nicely from source on any modern Linux distribution.[46] While Lagrange has recently added an experimental command-line version that looks fantastic, there is also the Python-based AV-98 implementation in that space,[47] named after the robot from *Patlabor* for maximum cyberpunk appeal. If you want to check it out before downloading an app, there is also a proxy server[48] that will translate gemini:// URLs into web pages for you to view from your web browser, like so.[49]

Shintaro Miyazaki: Where do you situate “countering” in minimal computing? How would you situate minimal computing within countering and how would you imagine its role?

Andrew Lison: The first question is perhaps somewhat difficult in that I wouldn’t want to insist that any or all of the practices I’ve outlined above are by definition “countering”; many people use Linux every day and I think that its once-countercultural (if heavily libertarian) appeal has been largely defanged. That said, as I mentioned earlier, I do view a minimal computing approach as conducive to a kind of ideology critique. We are all familiar, on the one hand, with the paucity of modern, app-ified user experiences, the reduction of human-computer interaction into a handful of actions and icons: like, share, post, buy, etc. On the other, “experimental” approaches to UI often seem to me more like exercises in search of a problem. By contrast, minimal computing holds out the promise of combining Brechtian estrangement, in its jarring divergence from the kinds of onrails experiences that app developers want you to have, with the real awareness that its paradigms, born out of necessity and low bandwidth (recall that the command line was originally designed for teletype machines and that theclassic 80-column width of the terminal is based on that of an IBM punch card[50]), can foster.

This leads me to another way that I think that minimal computing can be a countering force. There is often a kind of faux modernism at work in many justifications of contemporary computing ecosystems that, as someone who arguably identifies more as a modernist than a postmodernist, seems to me to be hopelessly tied up with an ailing neoliberal status quo. For example, a recent article in *The Verge* urges those of us old enough to know what a file system is to “get ready”[51] for a world in which such outdated concepts no longer matter. The argument is that society has progressed past the need for files and directories, and that the range of cloud-based services offered by megacorporations like Google provides a new basis for re-thinking the fundamentals of computational organization. Leaving aside whatever one might think about these systems (although anyone who has tried to dig an old piece of information out of an AJAX web application like Facebook or Twitter’s timeline should come away with a newfound appreciation for manipulable, hierarchical structures like a directory tree or a mailbox file), these supposedly obsolete formats still very much exist, they are just obscured by the higher-layer systems that have been built on top of them! It is not clear to me, for example, that we will be able to build the kinds of file-less applications that the article insists are the antidote to fusty old blocks and sectors without an understanding of the files and folders that a computer will need to make use of in order to access them. Even the most heavily-sandboxed operating system or app still needs to call network drivers, address a display, and so on in ways that kernels, headers, libraries, directories, and other file system components facilitate. Put simply, you still need to know what a file is in order to move beyond it, and in that sense there is nothing “new” or transcendental about proprietary networks built on top of these foundations. The insistence that there is seems to comport all too well with the oligopoly power that large tech firms exercise at the expense of individuals’ control over their own systems.[52]

Minimal computing, then, can help reorient our thinking in this direction, whether we use it to build cloud-based services[53] or reject that paradigm altogether. In this sense I would perhaps resist the connection Sayers identifies between minimal computing and maintenance, and especially the association with the “long now” movement championed by many of the greatest beneficiaries of our present technocratic order. Their affinity for this kind of thinking has been rightly critiqued by, among others, Manuel Castells,[54] and for my part I find the insistence that we are going to conduct our lives in a freefloating cloud internet run by a few extraordinarily powerful corporations without concern for how it works or where our information is located (although they will certainly know all too well) much more resonant with the ideology of an eternal, Fukuyaman present[55] than a renewed attention to the fundamentals of computing. To put it another way, if someone took the plow back to the drawing board, would we understand this as an attempt to maintain a feudal, agrarian order? The real issue is, as we all know, the modernist impulse to “make it new” has seemingly run out of steam, but I would argue that this is now a problem for a *postmodern* order that has, ironically, legitimated itself in no small part on the basis of ever-increasing computational growth. The secret teleology that has been subtending our ostensibly anti-teleological times is reaching an inflection point, and while the effects of this breakdown can be seen all around us, minimal computing can function as a kind of counter to it. As I concluded my *Configurations* article, if we can’t rely on faster computers going forward we’re going to have to get a lot better at using the ones we have, and this requires the kind of closer-to-the-metal approach that minimal computing embodies.

Shintaro Miyazaki: Please suggest further counter-Ns, N-computing(s), or N-futuring(s). Or other scholars/artists to have a conversation with.

Andrew Lison: I think counter-quantum could be quite an interesting topic. The field of quantum computing is often proffered as a solution to the problems posed by the end of Moore’s law, but most computer scientists would agree that the situation is more complicated than that soundbite would suggest. Building on my last response, I think something along the lines of a maintenance-futuring would be a fantastic (and nicely paradoxical) addition; as I noted in my *Configurations* article, infrastructure studies has largely focused on straight conduits – electricity, satellites, fiber optics – and less on computation, on the one hand, while on the other, major initiatives to build support for maintaining our digital infrastructure exclude hardware from their purview,[56] so there is a lot of work to be done in terms of raising awareness of the material aspects of computation specifically.

In terms of other scholars and artists, there is a lot of exciting work being done right now. People like Lisa Parks, Jonathan Beller, Anne Pasek, and Kodwo Eshun are all both academics and makers in varying proportions and would be fascinating interlocutors with whom to discuss counter-strategies to dominant digital paradigms.

1 Alex Gil, “The User, the Learner and the Machines We Make,” *Minimal Computing*, 21 May 2015, available at http://go-dh.github.io/mincomp/thoughts/2015/05/21/user-vs-learner/.
2 Jentery Sayers, “Minimal Definitions,” *Minimal Computing*, 2 October 2016, available at https://go-dh.github.io/mincomp/thoughts/2016/10/02/minimal-definitions/.
3 Andrew Lison, "Toward a Theory of 100% Utilization," *Configurations*, vol. 28, no. 4 (2020): 491–519, available at https://doi.org/10.1353/con.2020.0024.
4 Gordon E. Moore, “Cramming More Components onto Integrated Circuits” (1965), reprinted in *Proceedings of the IEEE* vol. 86, no. 1 (January 1998): 82–5, available at https://ieeexplore.ieee.org/document/658762.
5 Friedrich Kittler, “There is No Software,” *Stanford Literature Review* vol. 9 no. 1 (Spring 1992): 81–90.
6 M. Mitchell Waldrop, “More than Moore,” *Nature* 530 (11 February 2016): 144–7, available at https://www.nature.com/news/the-chips-are-down-for-moore-s-law-1.19338.
7 Gabriel Egan, “Old Machines Running Old Languages,” *Minimal Computing*, 3 August 2015, available at https://go-dh.github.io/mincomp/thoughts/2015/08/03/old-machines/.
8 Don Clark, “U.S. Retakes Top Spot in Supercomputer Race,” *New York Times*, 30 May 2022, available at https://www.nytimes.com/2022/05/30/business/us-supercomputer-frontier.html.
9 Nicole Perlroth, Jeff Larson, and Scott Shane, “N.S.A. Able to Foil Basic Safeguards of Privacy on Web,” *New York Times*, 5 September 2013, available at https://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html.
10 See https://www.raspberrypi.com/.
11 Jimmy Maher, *The Future Was Here: The Commodore Amiga* (Cambridge, MA: MIT Press, 2018).
12 Sylvain Leroux, “The Truth About the Intel’s Hidden Minix OS and Security Concerns,” *It’s Foss*, 18 November 2017, available at https://itsfoss.com/fact-intel-minix-case/.
13 See https://www.arduino.cc/ and https://www.raspberrypi.com/products/raspberry-pi-pico/.
14 Dennis Tenen and Grant Wythoff, “Sustainable Authorship in Plain Text using Pandoc and Markdown,” *Programming Historian*, 19 March 2014, available at https://programminghistorian.org/en/lessons/sustainable-authorship-in-plain-text-using-pandoc-and-markdown; Nick Montfort, Exploratory Programming for the Arts and Humanities (2016) 2nd edition (Cambridge MA: MIT Press, 2021).
15 See https://www.raspberrypi.com/.
16 See https://www.raspberrypi.com/software/.
17 See https://www.debian.org/.
18 “Silicon Power 32GB 3D NAND High Speed MicroSD Karte with Adapter,” Amazon, available at https://www.amazon.com/Silicon-Power-Speed-MicroSD-Adapter/dp/B07Q384TPK/.
19 See https://retropie.org.uk/.
20 See https://ubuntu.com/raspberry-pi.
21 Richard Smedley, *Conquer the Command Line: The Raspberry Pi Terminal Guide* (2016)

2nd edition (Cambridge, UK: Raspberry Pi, 2019), available at https://magpi.raspberrypi.com/books/command-line-second-edition.
22 See https://www.samba.org/, https://www.nginx.com/, and https://httpd.apache.org/, respectively.
23 See https://osmc.tv/.
24 “vi,” Wikipedia, last modified 4 October 2022, available at https://en.wikipedia.org/wiki/Vi.
25 See https://www.nano-editor.org/.
26 See https://www.python.org/ and https://gcc.gnu.org/.
27 Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language* (1978) 2nd edition (Englewood Cliffs NJ: Prentice Hall, 1988); Simon Long, *Learn to Code with C* (Cambridge, UK: Raspberry Pi, 2016), available at https://magpi.raspberrypi.com/books/essentials-c-v1; Harold Abelson, Gerald Jay Sussman, and Julie Sussman, *Structure and Interpretation of Computer Programs* (Cambridge MA: MIT Press, 1996), available at https://web.mit.edu/6.001/6.037/sicp.pdf.
28 See https://www.gnu.org/software/mit-scheme/.
29 See https://www.geany.org/ and https://www.sublimetext.com/.
30 Noam Nisan and Shimon Schocken, *The Elements of Computing Systems: Building a Modern Computer from First Principles* (2005) 2nd edition (Cambridge MA: MIT Press, 2021); see also https://www.nand2tetris.org/.
31 See https://hatari.tuxfamily.org/.
32 Andrew Lison, “New Media, 1989: Cubase and the New Temporal Order,” *Computational Culture*, no. 8 (July 2021), available at http://computationalculture.net/new-media-1989-cubase-and-the-new-temporal-order/.
33 Andrew Lison, “Hauntology, 1989,” *Twitter*, 12 July 2021, available at https://twitter.com/andrewlison/status/1414357384265404421.
34 See https://sourceforge.net/projects/previous/.
35 See https://www.gryphel.com/c/minivmac/.
36 “Playable English Localizations of Slovak Digital Games from the Late 80s Period,” *Slovenské Centrum Dizajnu*, available at https://scd.sk/clanky/playable-english-localizations-of-slovak-digital-games-from-the-late-80s-period/.
37 See https://skilldrick.github.io/easy6502/.
38 See https://gemini.circumlunar.space/.
39 “Gopher (protocol),” *Wikipedia*, last modified 7 September 2022, available at https://en.wikipedia.org/wiki/Gopher_(protocol).
40 “A quick introduction to ‘gemtext’ markup,” Project Gemini, available at https://gemini.circumlunar.space/docs/gemtext.gmi.
41 See https://www.markdownguide.org/.
42 See https://yaytext.com/bold-italic/.
43 See https://tildegit.org/solderpunk/molly-brown.
44 See gemini://soviet.circumlunar.space/.
45 See gemini://soviet.circumlunar.space/alison/.
46 See https://gmi.skyjake.fi/lagrange/.
47 See https://tildegit.org/solderpunk/AV-98.
48 See https://portal.mozz.us/.
49 See https://portal.mozz.us/gemini/soviet.circumlunar.space/alison/.
50 Neal Stephenson, *In the Beginning was the Command Line* (1999), archived at

https://web.archive.org/web/20180218045352/http:/www.cryptonomicon.com/beginning.html; Ken Shirriff, “IBM, sonic delay lines, and the history of the 80×24 display,” *Ken Shirriff’s Blog*, November 2019, available at https://www.righto.com/2019/11/ibm-sonic-delay-lines-and-history-of.html.
51 Monica Chin, “File not found: A generation that grew up with Google is forcing professors to rethink their lesson plans,” The Verge, 22 September 2021, available at https://www.theverge.com/22684730/students-file-folder-directory-structure-education-gen-z.
52 Mark Hachman, “New Windows 11 preview makes Microsoft accounts mandatory for (almost) all. Work and school accounts are the only exceptions,” *PCWorld*, 9 May 2022, available at https://www.pcworld.com/article/699100/microsoft-extinguishes-windows-11-pro-account-loophole-in-latest-build.html.
53 See https://nextcloud.com/athome/.
54 Manuel Castells, *The Rise of the Network Society* (1996) 2nd edition (2000) with new preface (Chichester: Wiley-Blackwell, 2010).
55 Francis Fukuyama, “The End of History?” *The National Interest*, no. 16 (1989): 3–18, available at http://www.jstor.org/stable/24027184.
56 “Critical Digital Infrastructure Research,” *Ford Foundation*, available at https://www.fordfoundation.org/campaigns/critical-digital-infrastructure-research/.

Andrew Lison
Minimal Computing

This publication is part of COUNTER-N, an online journal edited
by Özgün Eylül İşcen and Shintaro Miyazaki. It is published electronically
on the edoc-server of the Humboldt-Universität zu Berlin,
https://edoc.hu-berlin.de/

Title: Minimal Computing
Authors: Andrew Lison
Editors: Özgün Eylül İşcen, Shintaro Miyazaki
Graphic Design: Ahmet M. Öğüt
Published in 2022 by counter-n.net

DOI: doi.org/10.18452/25607
Final Submission: September 15th, 2022