Minimalism (computing)

In computing, **minimalism** refers to the application of minimalist philosophies and principles in the design and use of hardware and software. Minimalism, in this sense, means designing systems that use the least hardware and software resources possible.

Contents

History

In the late 1970s and early 1980s, programmers worked within the confines of relatively expensive and limited resources of common platforms. Eight or sixteen kilobytes of RAM was common; 64 kilobytes was considered a vast amount and was the entire address space accessible to the 8-bit CPUs predominant during the earliest generations of personal computers. The most common storage medium was the 5.25 inch floppy disk holding from 88 to 170 kilobytes. Hard drives with capacities from five to ten megabytes cost thousands of dollars.

Over time, personal-computer memory capacities expanded by orders of magnitude and mainstream programmers took advantage of the added storage to increase their software's capabilities and to make development easier by using higher-level languages. By contrast, system requirements for legacy software remained the same. As a result, even the most elaborate, feature-rich programs of yesteryear seem minimalist in comparison with current software. Many of these programs are now considered abandonware.

One example of a program whose system requirements once gave it a heavyweight reputation is the GNU Emacs text editor, which gained the backronym "Eight Megabytes And Constantly Swapping" in an era when 8 megabytes was a lot of RAM.[1] Today, Emacs' mainly textual buffer-based paradigm uses far fewer resources than desktop metaphor GUI IDEs with comparable features such as Eclipse or Netbeans.[2] In a speech at the 2002 International Lisp Conference, Richard Stallman indicated that minimalism was a concern in his development of GNU and Emacs, based on his experiences with Lisp and system specifications of low-end minicomputers at the time.[3]

As the capabilities and system requirements of common desktop software and operating systems grew throughout the 1980s and 1990s, and as software development became dominated by teams espousing conflicting, faddish software development methodologies, some developers adopted minimalism as a philosophy and chose to limit their programs to a predetermined size or scope.[4] A focus on software optimization can result in minimalist software, as programmers reduce the number of operations their program carries out in order to speed execution.[5]

In the early 21st century, new developments in computing have brought minimalism to the forefront. In what has been termed the post-PC era it is no longer necessary to buy a high-end personal computer merely to perform common computing tasks.[6] Mobile computing devices, such as smartphones, tablet computers,

netbooks and plug computers, often have smaller memory capacities, less-capable graphics subsystems, and slower processors when compared to the personal computer they are expected to replace. In addition, heavy use of graphics effects like alpha blending drains the battery faster than a "flat ui".[7] The growing popularity of these devices has made minimalism an important design concern.

Google's Chrome browser and Chrome OS are often cited as examples of minimalist design.[8][9] In Windows 8, Microsoft decided to drop the graphics-intensive Aero user interface in favor of the "simple, squared-off" Metro appearance, which required fewer system resources. This change was made in part because of the rise of smaller, battery-powered devices and the need to conserve power.[10][11][12] Version 7 of Apple's iOS made similar changes for user experience reasons.[13]

Usage

Developers may create user interfaces to be as simple as possible by eliminating buttons and dialog boxes that may potentially confuse the user. Minimalism is sometimes used in its visual arts meaning, particularly in the industrial design of the hardware device or software theme.

Some developers have attempted to create programs to perform a particular function in the fewest lines of code, or smallest compiled executable size possible on a given platform.[14][15] Some Linux distributions mention minimalism as a goal. Alpine Linux, Arch Linux, Puppy Linux, Bodhi Linux, CrunchBang Linux, dynebolic[16] and Tiny Core Linux are examples. The early development of the Unix system occurred on low-powered hardware, and Dennis Ritchie and Ken Thompson have stated their opinion that this constraint contributed to the system's "elegance of design".[17]

Programming language designers can create minimal programming languages by eschewing syntactic sugar and extensive library functions. Such languages may be Turing tarpits due to not offering standard support for common programming tasks. Creating a minimal Lisp interpreter is a common learning task set before computer science students.[18] The Lambda calculus, developed by Alonzo Church is a minimal programming language that uses only function definitions and function applications.[19][20] Scheme,[21][22] Forth,[23] and Go[24][25] are cited as examples of practical, minimal programming languages.

The programming hobby of code golf results in minimalist software,[26] but these are typically exercises or code poetry, not usable applications software.

John Millar Carroll, in his book *Minimalism Beyond the Nürnberg Funnel* pointed out that the use of minimalism results in "instant-use" devices such as video games, ATMs, voting machines, and mall kiosks with little-or-no learning curve that do not require the user to read manuals.[27] User Interface researchers have performed experiments suggesting that minimalism, as illustrated by the design principles of parsimony and transparency, bolsters efficiency and learnability.[28] Minimalism is implicit in the Unix philosophies of "everything is a text stream" and "do one thing and do it well", although modern Unix/Linux distributions do not hold so rigorously to this philosophy.[29]

See also

- Code bloat
- Code refactoring
- Concision: to be brief, terse, succinct. To say in few words.
- Don't repeat yourself
- Feature creep
- KISS principle

- Light-weight Linux distribution
- Muntzing
- Pareto principle 80:20 rule
- Rule of least power
- Software bloat
- Unix philosophy
- Wirth's law
- Worse is better
- Zawinski's law of software envelopment

References

1. Hagen, William von (2010-05-13). *Ubuntu Linux Bible: Featuring Ubuntu 10.04 LTS* (https://books.google.com/books?id=rsSlrQLB8-gC&pg=PT328&lpg=PT328). ISBN 9780470881804.
2. "Five reasons why Emacs will always be better" (http://www.sigasi.com/content/five-reasons-why-emacs-will-always-be-better).
3. "My Lisp Experiences and the Development of GNU Emacs" (https://www.gnu.org/gnu/rms-lisp.html). "...I aimed to make the absolute minimal possible Lisp implementation. The size of the programs was a tremendous concern. There were people in those days, in 1985, who had one-megabyte machines without virtual memory. They wanted to be able to use GNU Emacs. This meant I had to keep the program as small as possible. For instance, at the time the only looping construct was *while*, which was extremely simple. There was no way to break out of the 'while' statement, you just had to do a catch and a throw, or test a variable that ran the loop. That shows how far I was pushing to keep things small. We didn't have 'caar' and 'cadr' and so on; "squeeze out everything possible" was the spirit of GNU Emacs, the spirit of Emacs Lisp, from the beginning."
4. "dwm - dynamic window manager" (http://dwm.suckless.org/).
5. ne has been written with sparing resource use as a basic goal. Every possible effort has been made to reduce the use of CPU time and memory, the number of system calls, and the number of characters output to the terminal. -- ne info page
6. Strickland, Jonathan (2009-02-12). "HowStuffWorks "What's the difference between notebooks, netbooks and ultra-mobi" " (http://computer.howstuffworks.com/notebook-vs-netbook-vs-ultra-mobile-pc.htm/printable). Computer.howstuffworks.com. Retrieved 2013-07-08.
7. "5 ways to improve battery life in your app" (http://radar.oreilly.com/2014/04/5-ways-to-improve-battery-life-in-your-app.html).
8. "Google Chrome Cr-48, Paragon of Minimalist Design" (https://www.pcmag.com/article2/0,2817,2374276,00.asp). *PC Magazine*. 2010-12-13.
9. Pilcher, Pat (2009-07-13). "Battle of the browsers - which is master of the web?" (https://www.independent.co.uk/life-style/gadgets-and-tech/features/battle-of-the-browsers--which-is-master-of-the-web-1743947.html). *The Independent*. London.
10. Chang, Alexandra (2012-05-21). "Microsoft Drops 'Aero Glass' User Interface in Windows 8 | Gadget Lab" (https://www.wired.com/gadgetlab/2012/05/microsoft-drops-aero-glass-ui-in-windows-8/). *Wired*. Wired.com. Retrieved 2013-07-08.
11. McCracken, Harry (2012-05-22). "Windows Aero: Why I'm Glad It's Dead | TIME.com" (http://techland.time.com/2012/05/22/windows-aero-why-im-glad-its-dead/). *Time*. Techland.time.com. Retrieved 2013-07-08.

12. In 2009, desktops were 44% of the worldwide market and laptops were 56%. Just 3 years later, over 61% of the PCs sold are laptops and the trend is accelerating—this is globally, measuring all Windows PCs sold. Among consumers in the United States buying a PC this year, more than 76% will purchase laptops—the absolute number of all US desktops sold will be fewer than the number of tablets in 2012! (http://blogs.msdn.com/b/b8/archive/2012/05/18/creating-the-windows-8-user-experience.aspx)
13. "Why Jony Ive Is Killing Skeuomorphism In iOS 7" (http://www.cultofmac.com/230648/why-jony-ive-is-killing-skeuomorphism-in-ios-7/). 2013-06-10.
14. "Crafting a Tiny Mach-O Executable" (http://osxbook.com/blog/2009/03/15/crafting-a-tiny-mach-o-executable/).
15. "Minimalist Cocoa programming" (http://cocoawithlove.com/2010/09/minimalist-cocoa-programming.html).
16. "Friendly to the environment" (https://web.archive.org/web/20140202101351/http://www.dynebolic.org/2011/09/environment-friendly/). Archived from the original (http://www.dynebolic.org/2011/09/environment-friendly/) on 2014-02-02. Retrieved 2014-01-31. "This operating system is designed to run on Pentium2 processors with 256MB RAM, not even an harddisk is needed. Unleash the full potential of computers even with a second hand PC."
17. "The Art of Unix Programming" (http://www.catb.org/esr/writings/taoup/html/). "A 1974 paper in Communications of the ACM gave Unix its first public exposure. In that paper, its authors described the unprecedentedly simple design of Unix, reported over 600 Unix installations. All were on machines underpowered even by the standards of that day, but (as Ritchie and Thompson wrote) "constraint has encouraged not only economy, but also a certain elegance of design.""
18. "Build Your Own Lisp" (http://www.buildyourownlisp.com/chapter1_introduction).
19. Stuart, Tom (2013-05-15). *Understanding Computation* (https://books.google.com/books?id=TN7UAZTKqccC&pg=PA162&lpg=PA162). ISBN 9781449330118.
20. "7 lines of code, 3 minutes: Implement a programming language from scratch" (http://matt.might.net/articles/implementing-a-programming-language/).
21. "The Evolution of Lisp" (http://www.dreamsongs.com/Files/HOPL2-Uncut.pdf) (PDF). "The initial report on Scheme [Sussman, 1975b] describes a very spare language, with a minimum of primitive constructs, one per concept. (Why take two when one will do?)"
22. "Scheme-faq-general" (http://community.schemewiki.org/?scheme-faq-general). "Scheme is a dialect of Lisp that stresses conceptual elegance and simplicity."
23. Biancuzzi, Federico; Chromatic (2009-03-21). *Masterminds of Programming: Conversations with the Creators of Major Programming Languages* (https://books.google.com/books?id=yB1WwURwBUQC&pg=PA60&lpg=PA60). ISBN 9780596555504. "Forth is a computer language with minimal syntax"
24. "Go" (https://golang.org/). "Go is an open source programming language that makes it easy to build simple, reliable, and efficient software."
25. "Interview with Ken Thompson" (http://www.drdobbs.com/open-source/interview-with-ken-thompson/229502480). "...we started off with the idea that all three of us had to be talked into every feature in the language, so there was no extraneous garbage put into the language for any reason."
26. "Wik Wiki A Wiki in 1287 characters of PHP" (http://c2.com/cgi/wiki?WikWiki).
27. John Millar Carroll (1998). *Minimalism Beyond the Nurnberg Funnel* (https://books.google.com/books?id=LvXiZJEUJjAC&pg=PA400&lpg=PA400&dq=minimalism+atm+machines). Cambridge, Mass.: MIT Press. ISBN 0-262-03249-X. Retrieved 2007-11-21.
28. Wren, C.; Reynolds, C. (2004). "Minimalism in Ubiquitous Interface Design" (http://affect.media.mit.edu/pdfs/04.wren-reynolds.pdf) (PDF). *Personal and Ubiquitous Computing*. Springer. **8** (5): 370–373. doi:10.1007/s00779-004-0299-2 (https://doi.org/10.1007%2Fs00779-004-0299-2). Retrieved 2008-07-29.

29. "Uzbl - web interface tools which adhere to the unix philosophy" (http://www.uzbl.org/readme.php). "The general idea is that Uzbl by default is very bare bones."
