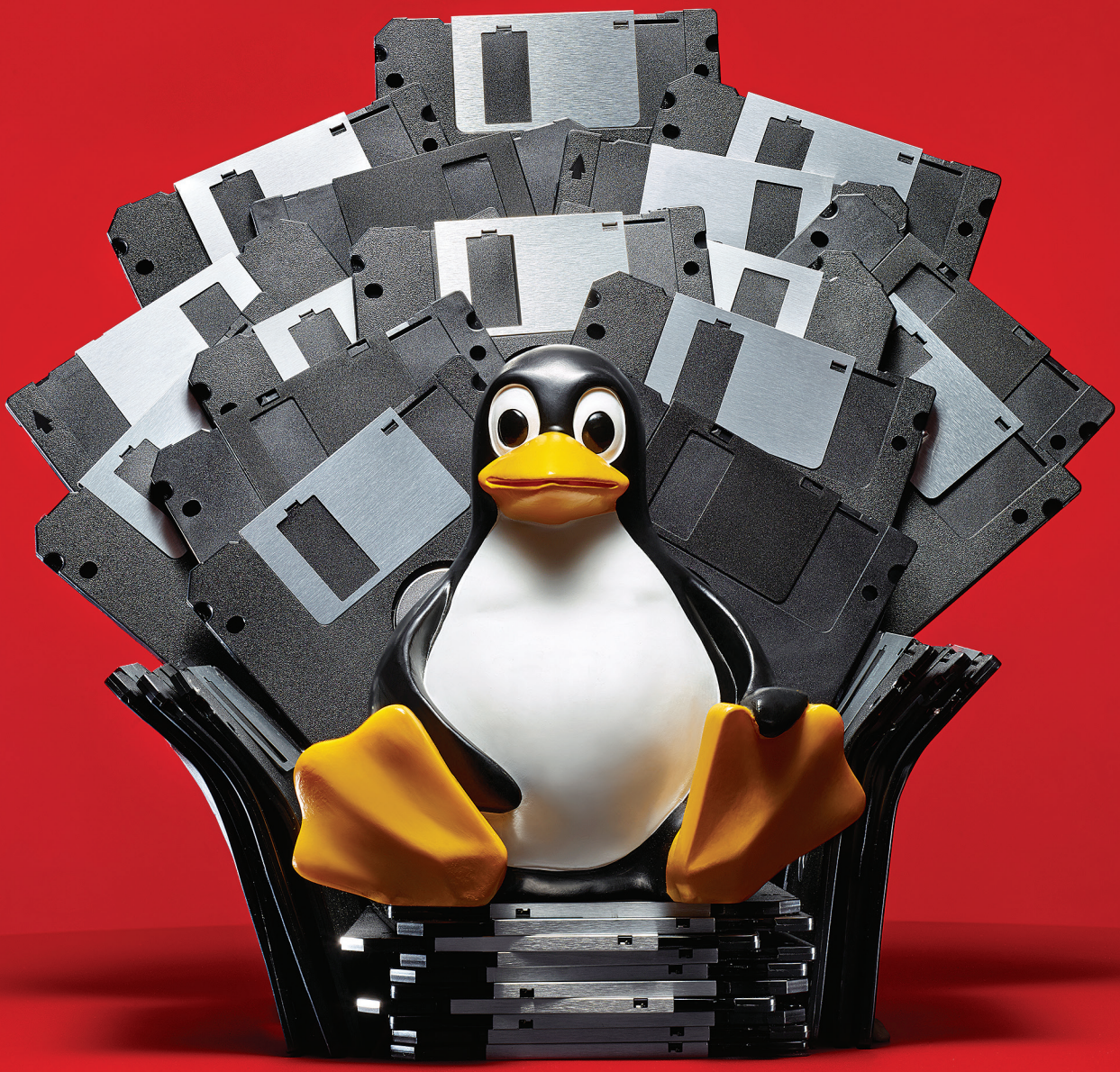
Linux at 25

Why this open-source marvel triumphed over its rivals

By Christopher Tozzi

Revolution was brewing on the European periphery in the summer of 1991. Years of struggle by outspoken rebels against the status quo were coming to a boil, and a way of life that once seemed unassailable neared its collapse. ¶ Ask most historians to tell you about that revolution and they'll describe the events that preceded the dissolution of the Soviet Union. As an attempted coup d'état by reactionary hardliners failed and Boris Yeltsin outlawed the Communist Party, it became clear to the world that the radical fervor that began sweeping across Eastern Europe in the late 1980s would soon undo the once-mighty Soviet empire. ¶ Yet, in another corner of Europe, a revolution of a different sort was stirring. No one—not even its chief instigator—recognized its significance. Nonetheless, the code that an irreverent Finnish college student named Linus Torvalds quietly unveiled in August 1991 has ended up touching at least as many lives as did the political upheavals of the late 20th century. I'm talking, of course, about Linux. »»

PHOTOGRAPH BY The Voorhes



Torvalds did not plan any of this. He was merely an "accidental revolutionary," as he described himself in his autobiography, *Just for Fun* (2001, HarperBusiness). Almost unwittingly, he kick-started the free-software revolution—a movement that much more prominent programmers had been trying to get off the ground for years.

It's all the more remarkable, then, that Linux, which celebrates its 25th birthday later this year, has so profoundly challenged the norms of software development. It showed programmers everywhere that a different world was possible—a world where they could share code openly, collaborate informally, and make a decent living, even if they gave away the chief product of their labor for free. The advantages of working this way have since become obvious to even the most hard-headed of business leaders, with most large software-development companies now sharing at least some of the fruits of their programmers' efforts openly.

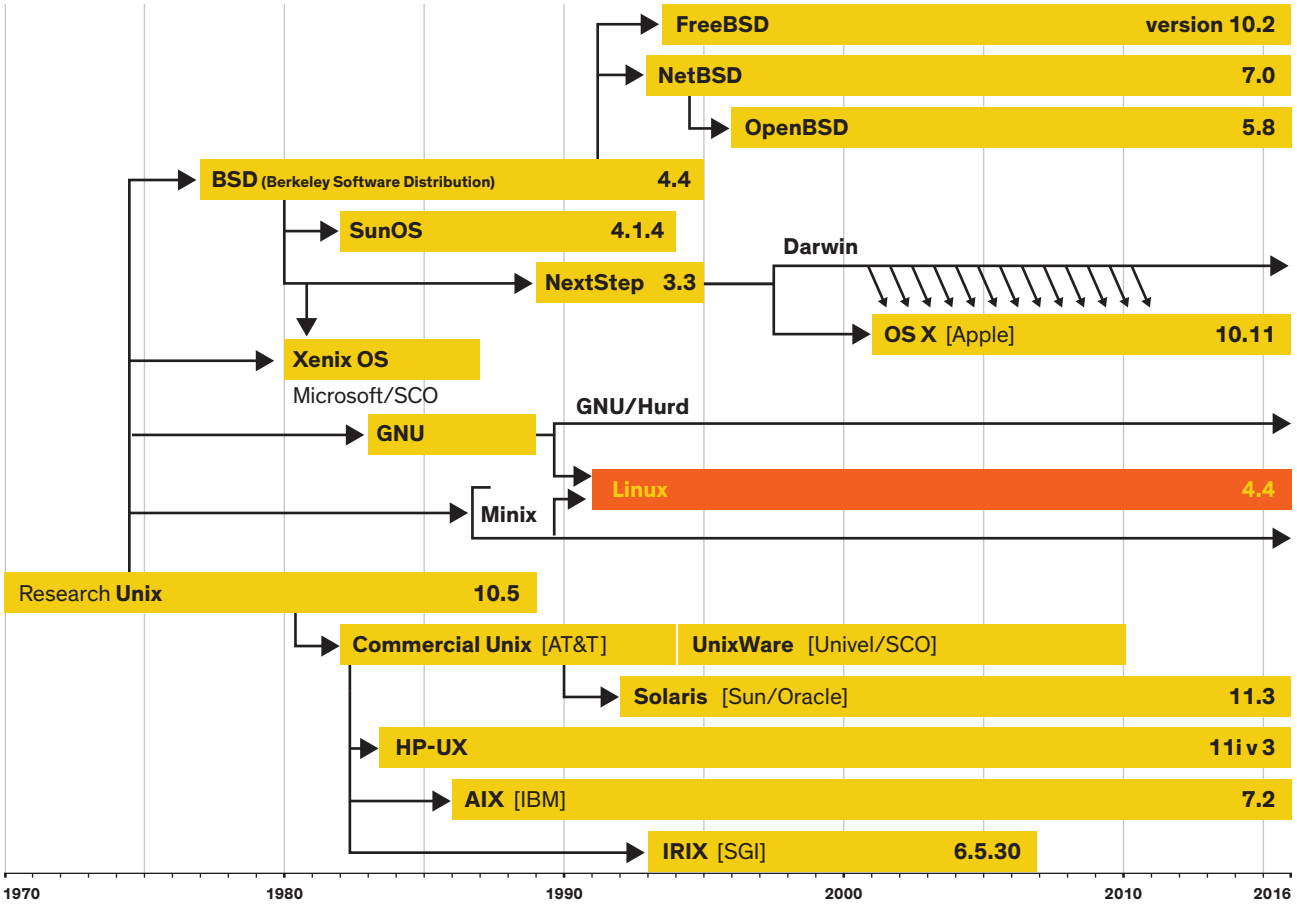
How did Linux end up producing such radical change? And why did other free-software activists' attempts to build bigger and seemingly better systems than Linux fail to achieve as much momentum? With the insight that comes from retrospection, it's now possible to answer those questions.

Of course, in 1991 it would have been ludicrous to suggest that Linux would end up as anything notable. Torvalds had less than two years of college-level work to his name when he started writing the Linux kernel—the code that manages an operating system's core functions—early that year. He worked out of a sparse apartment in Helsinki, where his only computer was an off-the-shelf PC with an Intel 80386 processor, known colloquially as a "386."

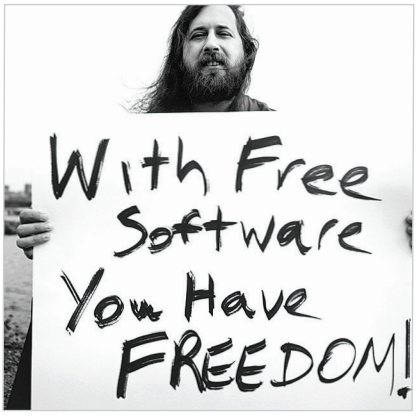
Meanwhile, two major teams of professional programmers at some of the world's most elite computer-science research labs were developing other free-software kernels. (The term *open-source software*, more commonly used to describe Linux and similar efforts today, did not come into vogue until 1998.) These teams had each spent years laboring to create a kernel that could do what Linux alone ultimately did.

One team was working on an operating system known as Berkeley Software Distribution, or BSD, which had been in development since the late 1970s at the University of California, Berkeley. BSD was initially conceived as an enhanced version of the Unix operating system, whose code was owned by AT&T. But the BSD project evolved into an endeavor to create a Unix clone that was completely free of AT&T code.

ORIGIN OF SPECIES: Linux, along with many competitors, had its conceptual roots in the Unix operating system, which the late Dennis Ritchie, Keith Thompson, and others developed at AT&T's Bell Labs.



SOURCE: WIKIMEDIA COMMONS



LONG LIVE HACKERDOM: Richard Stallman [above, in 1998] spearheaded another effort to develop a Unix-like operating system: GNU/Hurd.

In June 1989, the BSD team rolled out Networking Release 1, or Net/1, its first package of Unix-like software that users could legally deploy without purchasing a Unix license from AT&T. But because Net/1 consisted primarily of networking code, it was hardly a replacement for Unix. Two years later, the developers released a complete operating system. The platform, called Net/2, provided what most people considered the first fully functional free-software operating system.

"Free software" is in the eye of the beholder, however, and not everyone was satisfied with the code from Berkeley. On the other side of the United States, in Cambridge, Mass., another team of programmers led by Richard Stallman was building its own freely redistributable Unix clone, called GNU, a recursive acronym for "GNU's Not Unix!"

Stallman, who viewed the BSD license as problematic because it did not require the source code of derivative works to remain available, started the GNU project in January 1984. Like the BSD developers, Stallman and his team were professional computer scientists. They had access to computer labs at MIT, where Stallman—who in 1990 received a MacArthur "genius" grant for his work on GNU—had been employed prior to launching the GNU project.

By 1991, the GNU team had produced functional components of all the important parts of the operating system—except the one that mattered most: the kernel. They had hoped they might adapt

NATHANIEL WELCH/CORBIS

Linus Torvalds Reflects on 25 Years of Linux

The creator of the open-source operating system discusses its past, present, and future

By Stephen Cass

In this abridgment of an e-mail interview with *IEEE Spectrum*, Linux's creator reflects on the last quarter century and what's to come. The complete interview is available online.

Stephen Cass: What's one thing you know now that you wish your younger self knew?

Linus Torvalds: Actually, I credit the fact that I didn't know what the hell I was setting myself up for for a lot of the success of Linux. If I had known what I know today when I started, I would never have had the chutzpah to start writing my own operating system: You need a certain amount of naïveté to think that you can do it.

I was [also] perhaps more open to outside suggestions and influence than I would have been if I had a very good idea of what I wanted to accomplish. [Developers] could join with their own vision of where things should go. I think that helped motivate lots of people.

S.C.: Is there one early technical decision made during Linux's development that you now wish had gone a different way?

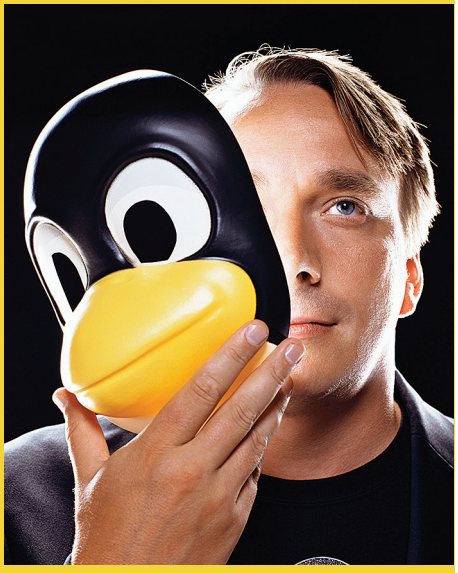
L.T.: The thing about bad technical decisions is that you can always undo them. It's obviously better to always make the right decision every time, [but] I'd rather make a decision that turns out to be wrong later than waffle about possible alternatives for too long.

S.C.: Why do you think Linux never became a significant presence on mainstream desktops?

L.T.: Hey, still working on it. There are multiple reasons, but one of the big ones is simply user inertia. The desktop is…both very per-

sonal—you interact with it rather intimately every day if you work with computers—but also complicated in ways many other computing environments aren't.

Look at your smartphone. That's also a fairly intimate piece of computing technology, and one that people get pretty attached to (and one where Linux, thanks to Android, is doing fairly well). The desktop is in many ways more complex, with much more legacy baggage.



S.C.: What's the biggest challenge currently facing Linux?

L.T.: The kernel is actually doing very well. It can be very challenging to get big and invasive changes accepted, so I wouldn't call it one big happy place, but I think kernel development is working. Many other open-source projects would kill to have the kinds of resources we have.

That said, one continual challenge we've always had in the kernel is the plethora of hardware out there. The good news is that a lot of hardware manufacturers are helping. That didn't used to be true.

S.C.: What current technical trends are you enthusiastic about? Are there any that dismay you?

L.T.: I have always been interested in new core hardware, particularly CPUs. In a bigger picture, it's very interesting to see how AI is finally starting to really happen. I'm not dismayed by the fact that true AI may finally start to happen, like clearly some people are. Not at all.

S.C.: Do you think Linux will still be under active development on its 50th anniversary? What is the dream for what that operating system would look like?

L.T.: I'm not a big visionary. I'm a very plodding pedestrian engineer, and I try to keep my eyes firmly on the ground. I suspect that we've seen many more changes in how computers work in the last 50 years than we're necessarily going to see in the future. [We've] simply learned what works and what does not.

Of course, neural networks, et cetera, will change the world, but part of the point with them is that you don't "program" them. They learn. They are fuzzy. People will want smarter machines, but people will also want machines that do exactly what they're told. So our current style of "old-fashioned" computing won't be going away; it'll just get augmented.

code developed elsewhere to build the GNU kernel, but that strategy didn't pan out. It wasn't until mid-1991—close to the time when Torvalds started writing Linux—that GNU developers finally began the tedious work of developing from scratch their own kernel, which they named "the Hurd."

GNU's Hurd project was well publicized, as was the release of BSD's operating system. Even in far-off Helsinki, Torvalds was well aware of both. And yet he wrote the Linux kernel anyway. Why?

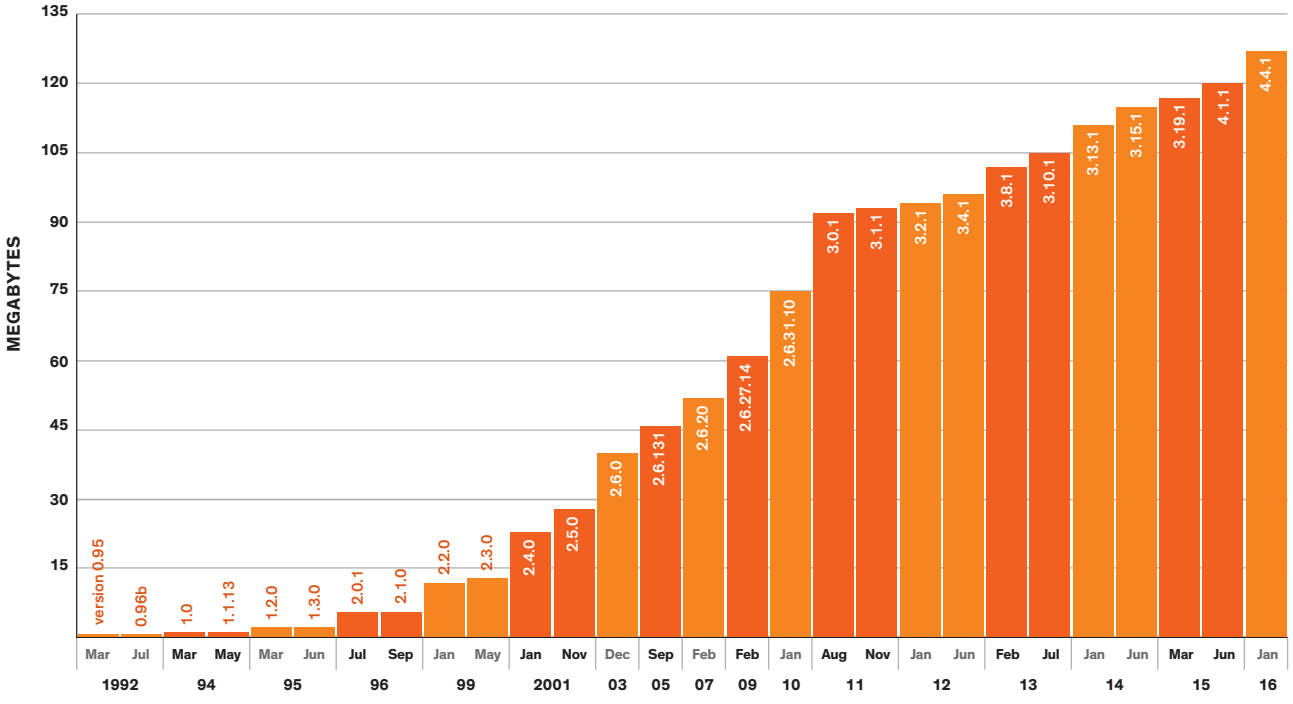
The answer has several parts. The simplest is that Torvalds did it "just for fun," as he explained in his autobiography. To him, Linux was a learning exercise. He was especially keen on using it to familiarize himself with the peculiarities of the 386 PC he had recently acquired.

Torvalds's second motivation stemmed from the deficiencies that he found in the operating system he was running at the time—Minix. He used it because BSD's Net/2 did not run on PC hardware and because GNU was not yet complete. Minix was yet another Unix-like operating system, released in 1987 by Andrew Tanenbaum, a professor of computer science at VU Amsterdam.

Tanenbaum had designed Minix as a teaching tool rather than as a full-blown operating system. It worked well enough for Torvalds, but he felt it lacked many important features. For one thing, it wasn't well suited for PCs with 386 processors. It also offered no terminal emulator, which Torvalds needed to log into his university's Unix system from his home PC.

And so at some point early in 1991—he doesn't remember exactly when it was—Torvalds began writing a terminal emulator for Minix. After completing it, he added disk and file-system drivers so that he could upload and download files from remote computers through the emulator. Those steps brought him closer to producing a complete operating-system kernel. By July, he was asking fellow Minix users on a Usenet bulletin board about documentation for POSIX, the Portable Operating System Interface. This is a technical standard for Unix-like operating systems, one that the IEEE Computer Society had

IAN WHITE/CORBIS



BIGGER AND BETTER: The size of the Linux kernel (shown here in GZIP-compressed file size) has grown enormously over the past quarter century.

formulated several years before. His interest in POSIX was a sure sign that he intended to write a Unix-like kernel.

At the end of August, in the same Usenet group, Torvalds announced his kernel to the world. He warned that it was "just a hobby" and "won't be big and professional like gnu." The following month, a friend posted the first version of the code on an FTP server, taking it upon himself to name it Linux even though Torvalds had dismissed that name for public use, preferring instead to call it Freax. From that server, anyone could download the code, modify it, and send the changes back to Torvalds. And many began doing so.

Of course, Linux took time to mature. Torvalds didn't release Linux 1.0, the first version he deemed of production quality, until 1994. And only gradually did others—some for fun, some in pursuit of profit—begin combining the Linux kernel with other software programs, notably the utilities that the GNU team had produced, to build complete Linux distributions. (Starting in 1994, Stallman and other GNU developers advocated for the name "GNU/Linux" to describe these distributions, but such terminology has enjoyed little following beyond some free-software purists.)

By the mid-1990s, it was clear that Linux was here to stay—and that the prospects for widespread adoption of competing free-software kernels were growing dim.

It would've made more sense if the Hurd, BSD, or even Minix had caught on. But Linux proved far more enduring. Why?

Most observers of the open-source software movement have attributed Linux's success to fortunate timing. Lars Wirzenius, a Finnish programmer who shared an office with Torvalds at the University of Helsinki when they were students, noted at the 1998 Linux Expo that "if the Hurd had been finished a few years ago, Linux probably wouldn't exist today. Or the BSD systems might have taken over the free operating system marketplace."

Tanenbaum made a similar point when he wrote in the early 2000s that legal troubles surrounding BSD in the 1990s "gave Linux the breathing space it needed to catch on." Those troubles began with a lawsuit in 1992 by Unix System Laboratories (USL), which at the time owned the Unix trademark. The company alleged that distributors of BSD-based software had improperly incorporated Unix code into their products. The case was settled out of court the next year, but then the Regents of the University of California countersued, claiming that USL had not properly credited the university for BSD code that was present in Unix.

In June 1993, Novell acquired USL, and the legal drama subsided. By February 1994, the parties had reached a settlement, which required a handful of changes to the BSD code. So in the end, the lawsuits did not limit the ability of BSD developers to distribute their operating system or of users to run it. And indeed, operating systems derived from BSD, such as FreeBSD and OpenBSD, continue to enjoy a healthy following. But BSD's legal troubles did create uncertainty, which slowed adoption at a crucial moment. Unsure whether they could legally use BSD software without paying steep licensing fees to USL, coders shied away just as Linux was evolving into a full-featured alternative.

Meanwhile, the Hurd remained far from complete: It couldn't even boot a

SOURCE: HTTP://WWW.KERNEL.ORG/PUB/KERNEL/

computer until 1994, and the first alpha release, version 0.0, appeared only in 1996. By the end of the decade, GNU deemed the Hurd basically functional, and it remains under active development. But a production-quality version has yet to be released. In its absence, Linux became the basis for operating systems that otherwise consist mostly of GNU programs.

But timing alone doesn't fully account for the Linux kernel's explosive popularity. Even before the BSD lawsuits began in 1992, Linux had gained a small following. Moreover, when the legal machinations concluded in early 1994, Linux was no more mature, in technical terms, than BSD, which at any rate could boast support for a broader range of hardware platforms.

Timing was therefore only one factor. Another involved licensing. The Linux code was available under the GNU General Public License, or GPL, which is termed "copyleft"—that is, it requires programmers to make the source code for all derivative works publicly available. In contrast, the BSD licenses allowed developers essentially to do whatever they wished with derivative code, including making it closed-source.

In that sense, BSD came with fewer licensing requirements. But many programmers who worked with Unix-like operating systems didn't see that as a virtue. For them, protecting the core tenets of the hacker culture that had given rise to Unix itself—a culture that valued transparency, openness, and the sharing of code—was most important. The BSD licensing terms, more than the lawsuits, were the main reason these programmers focused their attention on Linux.

It mattered, too, that the Linux code never cost a penny. From the outset, Torvalds was profoundly opposed to the idea that anyone should have to pay for his kernel. Indeed, prior to switching to the GPL in early 1992, he had released Linux under a crude license he wrote himself, which required that users "not profit from the distribution. In fact even 'handling costs' are not acceptable."

Linux's radical lack of a price tag set it apart from most of the other freely licensed Unix-like kernels of the early 1990s. Minix cost US $169 dollars, an "outrageous" sum, in Torvalds's estimation. "Look at who makes money off minix, and who gives linux out for free," Torvalds grumbled in a Usenet post at the time of Linux's birth. "Make minix freely available, and one of my biggest gripes with it will disappear." BSD's Net/2 could be legally copied by users without paying, but an official version on disk from Berkeley cost $1,000. Some of the other BSD derivatives were similarly priced. Even GNU's software had a price—as much as $5,000 dollars for the "deluxe distribution" of GNU software and manuals—if ordered directly from those running the project.

Torvalds's adamant opposition to charging for Linux made his kernel a true outlier. It probably also added to

Linux's appeal with hackers, who were wary of any activities by software distributors that smacked even remotely of commercialism.

All of these factors fostered the rich developer community that sprang up around Linux. As early as 10 October 1991, only weeks after the Linux code had become publicly available, Torvalds acknowledged that his nascent kernel "never would have seen the light of day or would have been much worse without the help of some others." He went on to name collaborators who were helping him develop Linux via the Internet.

To be sure, Torvalds was not the first programmer to embrace a decentralized, Internet-based community of developers. Keith Bostic, a lead BSD developer, did something similar in 1990, when he enticed hundreds of volunteer programmers from across the Internet to help rewrite Unix utilities in preparation for the release of Net/2. And according to Tanenbaum, Linux followed "essentially the same development model as Minix."

Still, the size and efficiency of Torvalds's following greatly exceeded anything that had come before. Within a few years, Torvalds had built a burgeoning, loosely organized community in which releasing code early and often, and relying on others to spot and fix bugs, became the means to quick improvement. By early 1994, insiders were commenting on the phenomenon. For example, Robert Young, a founder of the open-source software company Red Hat, wrote in the magazine *Linux Journal*, "The number and frequency of new releases of Linux, and drivers and utilities, are amazing to anyone familiar with traditional Unix development cycles." Such rapid innovation far outpaced what most other software projects could manage with their centralized control. It wasn't until 1997, when Eric S. Raymond wrote an essay (later turned into a book) titled "The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary" describing the way Linux developers operated, that a wider group of people began to appreciate the novelty and power of this new approach.

The momentum that Linux established in the early 1990s on the foundation of its fortunate timing, copyleft licensing, and lack of commercial ambitions among its core developers has sustained the kernel for 25 years. As of June 2015, Linux totaled 19.5 million lines of code—up considerably from just over 10,000 in 1991 and about 250,000 at the start of 1995. It is the work of more than 12,000 individual authors, some of whom have contributed just a few lines of code, others vast amounts. On average, 7.71 updates make their way into the kernel code every hour.

Linux is now used not just in many of the machines you'd recognize as a computer but also for embedded applications, meaning you'll find Linux in things like your wireless router, your e-reader, and your smart thermostat. A 2008 study estimated the kernel's total worth in monetary terms—difficult as that is to quantify for something often available for free—to be $1.4 billion. By now it must be several times this figure.

That's not bad for something a nerdy 22-year-old cooked up for kicks at the end of the Cold War. ■

POST YOUR COMMENTS at http://spectrum.ieee.org/linux0416