3 A KERNEL OF HOPE
The Story of Linux

THE FALTERING REVOLUTION: GNU AND BSD IN THE EARLY 1990S

FOR POLITICAL REVOLUTIONARIES in eastern Europe and central Asia, 1991 was a good year. By the close of that year, the Soviet Union had collapsed, leaving in its wake fifteen independent states and fledgling democracies.

In contrast, the GNU free software revolutionaries had much less to celebrate by the end of 1991. It appeared increasingly uncertain that the work they had pursued over the previous seven years would bring the FOSS revolution to fruition and make the world safe for hackers once again.

True, GNU had accomplished much by this time. As the previous chapter notes, the project had released feature-rich versions of almost all of the programs required to build a free, Unix-like operating system. People around the world were using GNU software. GNU developers were amassing cash donations in the six-figure range and enjoying strong support from a variety of companies and universities.

Yet GNU was stumbling in certain key respects. One of these was in the realm of kernel development. As the previous

chapter explains, GNU programmers did not finally start writing the Hurd kernel until the middle of 1991, and things did not go well from there. Without a kernel, the GNU system that Stallman had envisioned in 1983 remained akin to a mansion without a roof or a jet fighter without an engine: it was sophisticated and complete in all respects except for the one that mattered most.

Kernel problems were only one of the serious challenges GNU faced in the early 1990s. The others (also detailed in the previous chapter) involved GNU's failure to take PC hardware and software seriously until well after they had assumed an outsize role in the computer market. The centralized development approach that Stallman and his collaborators clung to also was a problem. The latter issue proved particularly detrimental for GNU because it meant that, for programmers, writing GNU code entailed basically the same type of work as coding for a commercial software company. From a developer's standpoint, there was nothing different about the way GNU created software that set the project apart from other major development initiatives. The GNU code was free, and that mattered to a lot of hackers. But coding for GNU was no more fun than coding for any other project. That fact helped to create an opening for Linux, which offered developers a fundamentally new model of collaboration, free of centralized hierarchies and lethargic release schedules.

As a result of these setbacks, by 1993 people began asking (to quote the title of a *Wired* magazine article) "Is Stallman Stalled?"[1] GNU had grown "bogged down," observers said, and GNU developers were sensing that their "window of opportunity to introduce a new operating system" was quickly passing

by, if it had not already disappeared—as Robert Chassell, one of the founders of the Free Software Foundation, put it at the time.[2]

Meanwhile, hackers had reason to doubt the prospects of the free software community's other great hope for building a free implementation of Unix. The BSD NET 2 operating system from Berkeley was released in 1991, as the previous chapter explains, and several derivative ports for PCs appeared during the two years that followed.[3] For a time, BSD seemed promising to hackers as an alternative to AT&T Unix. Even the GNU programmers, despite viewing the BSD licensing terms as insufficient for building a free operating system, saw enough value in NET 2 to start distributing copies of it by the summer of 1992 as part of the Free Software Foundation's software-distribution service.[4]

Yet the early momentum that the BSD derivatives enjoyed did not endure. For two main reasons, none of the BSD-based operating systems proved to be a good substitute for the type of system that hackers like Stallman envisioned. The first problem stemmed from the licensing terms under which the BSD software was released. As noted above, the Berkeley license that governed the BSD derivatives did not require people who distributed the software to provide source code with it. This meant that the BSD licenses failed to protect the hacker imperatives of transparency and sharing.

The second, more significant issue affecting the BSD derivatives involved legal troubles. In January 1992, a company called Berkeley Software Design, Inc. (BSDI) began selling a commercially supported version of the BSD NET 2 operating system with a price tag of $995—which, although steep for an

individual consumer, was 99 percent less expensive than AT&T's commercial Unix, as BSDI's advertisements noted prominently.[5] Soon after the platform hit the market, however, Unix Systems Labs, which owned the Unix trademark at the time, sued BSDI, alleging that the company infringed its trademark by describing BSD-based software as a form of Unix. It also charged that NET 2 contained copyrighted Unix code.

The case was settled out of court the next year. But the Regents of the University of California then countersued Unix Systems Labs. The university contended that the company had not properly acknowledged the BSD code that formed part of the System V version of Unix, as was required by the license agreement under which the University of California had released BSD.

The parties moved toward settlement in June 1993, when Novell acquired Unix Systems Labs. Novell's CEO at the time, Ray Noorda, began talks with Berkeley at the end of the summer, and the parties reached a settlement the following February.[6] As a result of the lawsuit, only three files were removed from the eighteen thousand that comprised the NET 2 code, and minor changes were made to some others. The legal challenges that BSD faced had no meaningful effect on the ability of hackers or anyone else to use and redistribute BSD-based systems freely.

Yet the challenges did serious damage. During the nearly two years between the start of the Unix Systems Labs lawsuit and the final settlement, uncertainty over whether users of BSD-based operating systems might be compelled to purchase an expensive license to run versions of BSD stymied adoption of the systems. Also problematic was the settlement's requirement

that BSD developers rewrite some minor parts of the BSD code that the agreement deemed to be in violation of the Unix Systems Labs copyright. The changes were not extensive and entailed relatively little effort. But they created a distraction for BSD programmers at a crucial moment in the system's development, when they might otherwise have been able to focus on implementing novel features instead of rewriting functions that already existed.[7]

The free versions of BSD did not go extinct in the early 1990s, and many survive into the present, as the next chapter notes. However, Berkeley's decision to disband the Computer Science Research Group, which had been the center for BSD development, following the release of the final version of BSD in June 1995 meant that the BSD community lost its central reference point. The various BSD derivatives that remained diverged in certain respects from one another. It is undeniable that the BSD legacy remains central to the FOSS world today: as many as half of the utilities in most GNU/Linux distributions descend from BSD code.[8] Nonetheless, because of both the Unix Systems Labs lawsuit and the BSD licensing terms, BSD-based operating systems never evolved into a satisfying solution for most hackers seeking a Unix-like operating system that promised to be free as in freedom.

THE MAN BEHIND THE KERNEL

While GNU listed and BSD NET 2 faced a stillbirth, a new generation of hackers dreamed of an operating system that would combine the allure of a free version of Unix with the accessibility

of PC hardware. It took one of the generation's own members, Linus Torvalds, to deliver on that vision.

In many ways, it was no surprise that Torvalds ended up producing the kernel that eluded teams of seasoned Unix hackers at both Berkeley and GNU. Geographically as well as technologically, Torvalds grew up in a very different world than that of the generation of programmers who preceded him. His background helped him to think in new ways about old programming problems.

One important element of Torvalds's experience was that, in contrast to programmers like Stallman or the BSD developers at Berkeley, he had only minimal exposure to advanced academic computer science research when he began working on the Linux kernel. He eventually completed a master's degree in computer science but not until 1996. When he introduced Linux to the world in the summer of 1991, he was not even close to finishing his undergraduate degree.

The difference in this regard between Torvalds and Stallman, who had nearly a decade of experience under his belt working as a programmer at one of the world's leading technical universities when he founded GNU, is striking. It also suggests part of the reason that Torvalds readily embraced a decentralized, Internet-based approach for developing Linux—even though such a strategy would have seemed anathema to most professional programmers at the time, weaned as they were on Brooks's law.

Another key distinction for Torvalds was that most of his early experiences with computers involved PCs. As a teenager, Torvalds used a Commodore VIC-20 machine, predecessor to the better known Commodore 64, that he inherited from his

grandfather.[9] Around his eighteenth birthday, using money he earned cleaning city parks in Helsinki, he replaced the Commodore with a QL computer from a British company named Sinclair.[10] Torvalds's third computer, which he purchased as an undergraduate student on the eve of beginning the programming venture that resulted in Linux, was a PC with an Intel 386 processor.[11] These machines did not look or work anything like the powerful, expensive PDP and VAX computers that the Unix hackers of old had learned to program on—and that developers like Stallman continued to regard as the only hardware platforms worthy of hosting an operating system as sophisticated as GNU's.

Torvalds did not encounter a large computer running Unix until the fall of 1990, when the University of Helsinki, where he was a student, acquired a MicroVAX computer that sported Ultrix, Digital Equipment Corporation's implementation of Unix.[12] By then, the young hacker had acquainted himself with operating systems that mimicked Unix but ran on the PC hardware with which he had grown up. To Torvalds, building a Unix-like operating system for the PC seemed like the obvious thing to do from the start, a characteristic that distinguished him in a crucial way from the hackers of Stallman's generation.

It also mattered that Torvalds lived in Finland, far from the major centers of computer science research at the time. Torvalds's hometown, Helsinki, had a university with a strong computer science department and as much Internet connectivity as most other major European cities of the time. Indeed, Finland was an early hub of computer research in northern Europe.[13] Yet because the country's computer industry remained minuscule compared to that of larger nations, Torvalds often had to order computer parts and programs from abroad, especially

during the years when his main computer was the British Sinclair. International snail-mail delivery of such items could take months. That was perhaps one factor that attracted Torvalds to distributing software via the Internet, which became crucial to the way he developed and shared Linux. The fact that he could meet relatively few computer science experts in Helsinki meant that the Internet also served as his lifeline for communicating with developers and exchanging ideas, as he did when announcing Linux over Usenet. In these respects, Torvalds's experience was distinct from that of the GNU and BSD programmers operating out of Cambridge and Berkeley, where they were at the centers of their respective hacker universes. In contrast, Torvalds was in a remote galaxy.

Just as Torvalds grew up on the geographic margins of the hacker community that he later helped to lead, he also belonged, in certain ways, to a marginal community within his own society. He was born into Finland's tiny Swedish-speaking minority, which constituted around 6 percent of the country's population at the time of his birth. Swedish-speaking Finns have long lived happily among their neighbors, and there is no evidence that Torvalds's linguistic background proved a major obstacle for him. (He also became fluent in English, the *lingua franca* of programmers on the Internet, at an early age.)[14] On the contrary, it placed him in a community that was at the forefront of the computing industry in Finland, where a majority of managers at companies such as IBM Finland belonged to the Swedish-speaking minority.[15]

Still, Torvalds's experience living outside the mainstream Finnish population may have helped him to think differently about projects like Linux. Torvalds lived on the periphery of

the dominant linguistic community in his country, just as the kernel he developed existed on the margins of the mainstream software culture of the early 1990s, defying the norms of both proprietary and free software programmers alike.

Finally, Torvalds has stated that his family had an important effect on some of the decisions he later made regarding Linux. "I undoubtedly would have approached the whole no-money thing a lot differently if I had not been brought up under the influence of a diehard academic grandfather and a diehard communist father," he wrote in his 2001 autobiography to explain why he was committed to releasing Linux free of charge.[16] His grandfather, one of Finland's first professional statisticians, died in 1983, when Torvalds was in his early teens. And Torvalds's father eventually moderated his hardline communist beliefs. Nonetheless, these two patriarchs imbued Torvalds at a young age with an appreciation for the value of research and exploration and radical notions of sharing and anticapitalist mores, and they shaped how Torvalds thought about the best ways to design and distribute software.

All of these experiences help to explain why Torvalds produced a novel operating system kernel and pioneered a radically new mode of software development that varied in crucial ways from what older hackers in the United States were building. Yet the specific event that set him on the path toward developing Linux was his purchase in the summer of 1990 of the book *Operating Systems: Design and Implementation*, which Andrew S. Tanenbaum, an American computer science professor teaching in the Netherlands, had published in 1987.[17] According to Torvalds, the book, which exposed him to "the philosophy behind Unix and what the powerful, clean, beautiful operating system

would be capable of doing," changed his life and "launched me to new heights."[18]

"As I read and started to understand Unix," he added, "I got a big enthusiastic jolt. Frankly, it's never subsided."[19]

WHY LINUX?

Tanenbaum's book was not about AT&T's version of Unix, though. It focused on a Unix-like operating system called Minix, which Tanenbaum created after receiving a Ph.D. from Berkeley and working for some time with the Unix group at Bell Labs.

According to Tanenbaum, his chief goal in writing Minix, which he coded from scratch entirely on his own, was to provide his computer science students with a Unix-like operating system whose source code was much less expensive than Unix's code. But it gained a broader following. Within a couple of months of Tanenbaum's release of the first version of Minix in 1987, the system "became something of a cult item, with its own USENET newsgroup, comp.os.minix, with 40,000 subscribers," he wrote. "Many people added new utility programs and improved the kernel in numerous ways."[20]

Tanenbaum, however, was reluctant to extend Minix—which was so-named because it was a miniature version of Unix—beyond the basic teaching tool he had designed it to be. In his recollection, "I didn't want it to get so complicated that it would become useless for my purpose—namely, teaching it to students." He added that there was every reason to believe that when Minix appeared, GNU and BSD were on the verge of providing a free, production-quality Unix implementation

that would far surpass anything Tanenbaum could develop on his own.[21]

From the perspective of someone like Torvalds, however, who began using Minix shortly after acquiring an Intel-based PC in early 1991, the chief importance of Tanenbaum's operating system was not its use for teaching.[22] It was that Minix was one of the first Unix-like operating systems that had been designed to run on microcomputer hardware rather than the PDP and VAX machines on which GNU and BSD development centered. In addition, although Minix cost some money (which irked Torvalds, as noted below), it was a teaching tool rather than an explicitly commercial product. That set it apart from other Unix-like operating systems for microcomputers, such as Coherent and Xenix.

Yet for Torvalds, Minix was not at all an adequate PC-based Unix implementation. In his view, Tanenbaum's lack of interest in extending the platform to include more features represented a major shortcoming. Torvalds was particularly disappointed that Minix did not work well on his PC, which had an Intel 386 processor. Minix ran on this type of computer, but because Tanenbaum had designed the operating system for a different family of microcomputers, it supported 386 chips only with the assistance of a special patch written by Bruce Evans. The patch was difficult to install, and after it was applied, it left much to be desired for someone like Torvalds who wanted to get the most out of a Unix-like environment on a 386 PC.

Minix also lacked terminal emulation, a feature that made it possible to log in to remote computers. That deficiency prevented Torvalds from using Minix on his home PC to connect

to the Unix computer at the university where he was a student.[23] Minix "had been crippled on purpose, in bad ways," because Tanenbaum "wanted to keep the operating system as a teaching aid," according to Torvalds.[24]

Another shortcoming of Minix, in Torvalds's view, was that it used a microkernel. As the previous chapter noted, Torvalds has strongly criticized microkernel architectures on technical grounds. Although his opinions on that topic may not have been fully developed before he started writing his own kernel, in early 1992 Torvalds felt strongly enough that Minix was poorly designed to complain to Tanenbaum that it was filled with "brain-damages," which derived from its microkernel architecture.[25] Torvalds also criticized Minix for its lack of portability and compliance with the POSIX standards of Unix-like operating system design.[26]

But the technical features that Minix lacked were only part of its problem for Torvalds. The licensing parameters that Tanenbaum imposed, particularly those that required payment of a fee to run Minix, seemed even worse. "Look at who makes money off minix, and who gives linux out for free," Torvalds seethed in a January 1992 Usenet post, in which he excoriated Tanenbaum for charging for Minix: "Then talk about hobbies. Make minix freely available, and one of my biggest gripes with it will disappear."[27] Torvalds found the cost of Minix—which amounted to $169 "plus conversion factor, plus whatever" someone in Finland might have to pay to acquire a Minix license—"outrageous at the time," he wrote in 2001, adding, "frankly, I still do."[28] He told me in 2016 that "free as in 'gratis' was actually an earlier concern than the whole 'free as in freedom'" consideration in motivating his decision to write Linux.[29]

Tanenbaum took offense at the criticism of Minix's cost, particularly because he thought he was doing cash-strapped students like Torvalds a favor by releasing Minix as a low-cost way to study Unix system design. In the same Usenet discussion mentioned above, Tanenbaum told Torvalds that, in writing Minix, "an explicit design goal was to make it run on cheap hardware so students could afford it." Referring to the GNU project, which was producing free software that for the most part ran only on very expensive computers designed for purchase by institutions, Tanenbaum sarcastically added, "Making software free, but only for folks with enough money to buy first class hardware, is an interesting concept."[30]

Tanenbaum has stuck by this argument. In an essay on Minix's history that he posted on the Internet in the early 2000s, he noted that Minix was never "free software in the sense of 'free beer'" (meaning that it was never free of cost) but that its source code was always freely available. In addition, the cost of a Minix license was a tiny fraction of what users paid for a commercial version of Unix. "By 1987," he explained, "a university educational license for UNIX cost $300, a commercial license for a university cost $28,000, and a commercial license for a company cost a lot more. For the first time, MINIX brought the cost of 'UNIX-like' source code down to something a student could afford."[31]

Yet Torvalds's abiding irritation with the fact that Minix and similar operating systems cost anything—even if Minix was much more affordable than the alternatives—reveals much about how thinking about money affected the nascent Linux community. Because Linux and the GNU software that accompanied it in the late 1990s became the centerpiece of a

flourishing commercial ecosystem with Torvalds's blessing, it has been easy for FOSS users to forget how opposed Torvalds was to the notion of making money off of Linux early on and how important that factor was in pushing him to start writing the kernel. In fact, Torvalds's animosity toward the prospect of profiting from software represented a crucial dimension of why he chose to write Linux, and it made his work different from that of Stallman and GNU.

Torvalds's choices are not fully explained by any of the other factors that influenced him. For one, there is little evidence that the philosophy of the Free Software Foundation had much effect on Torvalds when he crafted Linux. Torvalds attended a speech that Stallman gave "probably in 1991 or so," in Torvalds's recollection, at the Helsinki University of Technology, which introduced him to the Free Software Foundation's ideology.[32] That may have been part of the reason that he complained in October 1991 about Unix-like operating systems that "come with no source" and as a result "are ideal for actually using your computer, but if you want to learn how they work, you are f--ked."[33] That remark made it clear that Torvalds cared about keeping source code available for the utilitarian purpose of ensuring that other programmers could understand how it operated.

That sentiment, however, was different from sharing source code as a matter of moral principle, as Stallman did. Torvalds wrote that in 1991 he "wasn't much aware of the sociopolitical issues that were—and are—so dear to [Stallman]. I was not really all that aware of the Free Software Foundation. … judging from the fact that I don't remember much about the talk back in 1991, it probably didn't make a huge impact on my life at that point. I was interested in the technology, not the politics."[34]

The deficiencies that Torvalds perceived in Minix also do not account, on their own, for his decision to produce his own kernel. If those had been the only issue at play, Torvalds could simply have written a terminal emulator for Minix, tweak the Minix code so it would run better on Intel 386 hardware, and left it at that. Torvalds may have disagreed with the microkernel design of Minix, but that did not significantly affect his ability to use the operating system.

Neither was the joy of coding the main factor in Torvalds's decision to write Linux. It is true that he described programming in his autobiography—which was titled *Just for Fun*, implying that having fun was a large part of the reason for writing Linux—as "the most interesting thing in the world. It's a game much more involved than chess, a game where you can make up your own rules and where the end result is whatever you can make of it."[35] Depicting oneself as a developer who was passionate about coding for coding's sake might seem to be a mere public relations move. But there is good reason to believe that Torvalds was expressing genuine sentiment through such statements. The time and effort he invested into writing the first versions of Linux were considerable, and it is difficult to imagine a rationale human being undertaking such an endeavor, especially with no strong financial incentive, unless that person truly and deeply enjoyed the art of programming.

Yet personal amusement does not explain why Torvalds chose to write Linux or give it away for free to other people. If he just wanted to have fun programming, he could have written plenty of other programs rather than one that duplicated much of the functionality already available to him from Minix. And if Linux was purely about having fun for Torvalds, there was no

reason for him to invite others to contribute to his kernel, which reduced the amount of fun-inducing programming he had to do for himself.

So if free software ideology, shortcomings in Minix, and a desire to have fun do not fully explain why Torvalds wrote Linux, what does? The answer centers on money. Torvalds wanted a Unix-like kernel that was totally free of cost. As noted above, his chief complaint about Tanenbaum's Minix system in the early 1990s was its price tag. That issue remained a major point of contention for Torvalds at least as late as the 2001 publication of his autobiography—even though, by that time, he was a millionaire with plenty of cash at his disposal. As recently as 2013, Torvalds declared, with his characteristic humor, that "software is like sex: it's better when it's free."[36]

To note that the most significant motivation for Torvalds in writing Linux was to obtain an operating system that cost no money is not to dismiss the authenticity of his intentions or the validity of his work. On the contrary, as Tanenbaum himself suggested when contending that GNU's designers failed to give enough weight to hardware cost when choosing which platforms to support, there was plenty of good reason in the early 1990s to design a system that cost absolutely no money and ran on affordable PCs.

Yet the opposition to profit does mean that Torvalds thought very differently from the GNU developers in one key respect and that his work appealed primarily to a different set of users. Although the GNU project never required users to pay for its software if they copied it among themselves or downloaded it from the Internet, the project charged significant fees—up to $5,000—for official distributions of the code on disk as a way

to support itself, as the previous chapter notes. GNU was not concerned with making software free of cost for anyone, even those without the ability to download or copy code. If it had been, the project would likely have found a way to distribute its software on disk for free (as the organizers of the Ubuntu GNU/Linux distribution did in the 2000s via the ShipIt program, for instance).

For all of these reasons, the fact that no fully functional Unix-like operating system was available from GNU or elsewhere free of cost in 1991 gives the key to Torvalds's decision to write Linux. It also explains why his August 1991 Usenet post announcing Linux to the Minix community, which is quoted below, notes prominently that the operating system was "free" (with the implication that it cost no money) yet makes no mention of whether Torvalds intended to allow other people to view and modify his source code freely—which underscores how unimportant the Free Software Foundation's philosophical message was to Torvalds.[37] And it accounts for his decision, detailed below, to release the first versions of Linux under a license that prevented anyone from using the code to make money.[38] Only later did Torvalds's thinking about the relationship between software and money evolve in a way that led him to adopt the GPL license for the kernel, making it possible for Linux and GNU software to join forces to advance the FOSS revolution—and for people to make money using Linux code.

FROM MINIX TO LINUX

Although Torvalds's greatest complaint about Minix was its cost, his desire to shore up some of what he viewed as the

operating system's technical deficiencies sparked the beginnings of Linux—even though designing a complete kernel was not on his agenda when he first went to work.

The first release of the Linux kernel started as an effort by Torvalds to write a terminal emulator for Minix. As noted above, the absence of this type of program in Tanenbaum's operating system prevented Torvalds from using Minix on his home computer to connect remotely to his university's Unix server. To produce his own terminal emulator, Torvalds had to delve into the lower-level functionality of the 386 computer chip. That endeavor entailed tedious work. But for Torvalds it was a bonus because it meant that creating the terminal emulator would be a way to explore the intricacies of the computer hardware he owned. Toward that end, he wrote the emulator in assembly language, which was much more complex than a higher-level programming language, "just to learn about the CPU."[39]

Torvalds's terminal emulator for Minix was complete by the spring of 1991, and he could use it to log into his university's Unix system. Yet he was not satisfied. The emulator in its first iteration lacked disk and file system drivers, and without them, it was impossible to upload or download files from the remote server. So Torvalds went to work adding those features to his program.

The ground that Torvalds needed to cover to extend his terminal emulator into a complete operating-system kernel was narrowing because reading and writing to disks is one of the core responsibilities of a kernel. By the time he started work on the disk and file system drivers, "it was clear the project was on its way to becoming an operating system" rather than just a terminal emulator, Torvalds recalled.[40]

Perhaps because he trod a path similar to Stallman's, Torvalds referred to his project early on as the "gnu-emacs of terminal emulation programs."[41] As the previous chapter notes, Stallman's endeavor to build the GNU operating system started with the GNU Emacs text editor, a far cry from a complete operating system. For Stallman and GNU, Emacs was hardly the most essential part of the system, nor was it the most difficult to write. But it was the seed from which the rest of the GNU system sprang, at least conceptually. Torvalds's Minix terminal emulator, which was a similarly nonessential part of the kernel he eventually produced, did the same thing for Linux.

In descriptions of Linux's early history—or what could be called its prehistory because this was the period before Torvalds had fully conceived the kernel or even given it a name—Torvalds is unsure exactly when he made the choice to extend the terminal emulator into a kernel. He has written only that this happened sometime around April 1991.[42] Perhaps the decision evolved too gradually to associate with a particular date, or maybe Torvalds never consciously made it until after he was already well on his way to writing a kernel.[43]

What is certain is that Torvalds had a kernel on the brain by July 1991. In that month, he posted to the comp.os.minix Usenet newsgroup requesting links to a current version of the POSIX standards definition from the IEEE (Institute of Electrical and Electronic Engineers) Computer Society.[44] POSIX, an acronym for Portable Operating System Interface (with an X added in traditional Unix fashion), specifies how a Unix-like operating system should be designed to be feasible for porting to different hardware platforms. The standard also ensures compatibility with any software applications written for POSIX

compliance. By asking about the POSIX standards, Torvalds made clear that he was at least considering writing a kernel that would serve not only his personal purposes but also those of others interested in a new Unix-like operating system.

No one took serious notice of Torvalds's request at the time, and no Usenet readers responded with tips about obtaining the POSIX standards definition. As a result, Torvalds mined the information he sought from manuals at his university that described Sun's implementation of Unix. He also referred to Tanenbaum's book about operating-system design. Neither source provided the complete reference on POSIX standards that Torvalds sought, but they sufficed to allow him to continue to work on what was becoming the Linux kernel.[45]

Progress was slow at first, and this was not only because Torvalds lacked access to full POSIX documentation or prior experience programming something as complex as a kernel. His enthusiasm made up for those obstacles, but it did not make it any easier to contend with the complexities of the Intel 386 computer-chip architecture. Nor did it resolve the difficulty of debugging a kernel system during the early stages of development, when the code remains so incomplete that it is not possible to take advantage of sophisticated debugging tools, which help programmers identify the parts of code that cause a program to crash.

Asked in 1992 how he debugged early versions of the kernel, Torvalds wrote that what made the work so tedious was that he could not "even think about debuggers" at the time because none of those available to him worked well with Intel 386 hardware.[46] His code remained so basic that even printing error messages on the screen when a crash occurred was not possible.

Lacking sophisticated debugging resources, Torvalds explained, he adopted an approach that was pragmatic yet crude from the perspective of programmers like those at GNU, who wrote elegant, sophisticated code and had advanced debugging software at their disposal:

> What I used was a simple killing-loop: I put in statements like `die: jmp die` at strategic places. If it locked up, you were ok; if it rebooted, you knew at least it happened before the die-loop.[47]

In less technical terms, Torvalds meant that he inserted snippets of code at various places in his kernel program that he called *die-loops*, which caused the kernel to cease executing and freeze. If the system rebooted—as it did whenever the kernel crashed due to a code flaw—he would know that the issue had occurred at a point in the program prior to where he inserted the die-loop. If the system merely froze without rebooting, Torvalds knew that his program had run successfully up to the point in the code where the die-loop appeared. This approach to debugging was tedious and primitive, but it worked.

Progress became smoother after Torvalds had implemented enough of the kernel to "have a minimal system up [that] can use the screen for output." Yet even then he had to resort at times to the ugly die-loop approach described above. "All in all, it took about 2 months for me to get all the 386 things pretty well sorted out," to the point necessary to use more sophisticated debugging tools, Torvalds wrote in an early account of Linux's history.[48]

With the kernel code stable enough to accommodate more complicated debugging strategies, Torvalds implemented task switching, a key component of a Unix-like kernel, which allows

users to switch between different applications. He next wrote keyboard and serial drivers so that his kernel could communicate with the input devices connected to the computer. With those components in place, Torvalds finally began to enjoy "plain sailing: hairy coding still, but I had some devices, and debugging was easier. I started using C at this stage" rather than assembly language, a change that "certainly spe[d] up development. This is also when I start[ed] to get serious about my megalomaniac ideas to make 'a better minix tha[n] minix.'"[49]

Lack of reliable documentation on the finer points of Intel's 386 hardware remained a persistent problem for Torvalds. But he eventually overcame the challenge and completed a file system, the last part of the code that was required to have a basic equivalent of the Unix kernel. Torvalds's kernel at that point "wasn't pretty, it had no floppy driver, and it couldn't do much of anything. I don't think anybody ever compiled that version," he recalled in 1992.[50] Nonetheless, "by then I was hooked, and didn't want to stop until I could chuck out minix" completely, replacing it with the kernel he had built on his own.[51]

It was at this point, in August 1991, that Torvalds announced his kernel project to the world in a Usenet post on comp.os.minix that has since become something of a legend among Linux acolytes. Confirming the suspicions of users who had read his query about POSIX documentation a month earlier, Torvalds opened a Usenet thread on August 25, 1991, titled "What would you like to see most in minix?" It began:

> Hello everybody out there using minix—
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been

> brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).
>
> I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them:-)
>
> Linus (torvalds@kruuna.helsinki.fi)
>
> PS. Yes—it's free of any minix code, and it has a multi-threaded fs. It is NOT [portable] (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have:-(. [52]

At the time he made his efforts public, Torvalds had not yet given the kernel a name, asked other people to help him with development, or even shared his code publicly. (That did not happen until September 17, 1991, as the next section explains.) Yet his plan to build "a better Minix than Minix"—an important development because his attitude had originally been that his kernel "didn't have to do more than Minix"—was clear in his request for suggestions from the Minix community regarding features that were absent in Tanenbaum's operating system, but which they would like to see implemented.[53] That made Torvalds's kernel compelling. It encouraged contributions to Linux from other programmers, whose ranks steadily grew, along with the Linux code base.

Before examining how Linux evolved after other programmers began collaborating with Torvalds, it is worth reflecting on the significance of his achievement in writing an early version

of the kernel without outside assistance. Some members of the FOSS community, such as those who have called Torvalds a "god," have perhaps overstated how heroic his single-handed kernel-development effort was.[54] As Tanenbaum noted in his essay on Minix history, Torvalds was hardly the first programmer to write a Unix-like kernel with minimal assistance from other people. On the contrary, by Tanenbaum's count, "five people or small teams had independently implemented the UNIX kernel or something approximating it" before Torvalds began his work.[55] Moreover, as complex as a monolithic kernel is, the amount of coding involved in writing one is relatively small compared to the work required to produce all of the other programs that comprise a full operating system. Even by the late 1990s, when Linux had grown into a mature, feature-rich kernel that contained many times as much code as the primitive version Torvalds completed on his own in 1991, Linux accounted for only about 3 percent of the total code in a GNU/Linux operating system. GNU utilities and programs constituted about ten times as much.[56]

Yet Torvalds's ability to transform his Minix terminal emulator into a complete kernel in a matter of months was no mean feat. Torvalds was not the first programmer to write a kernel single-handedly, but his predecessors in that role had all been professional developers or computer science professors. In contrast, Torvalds had barely a year of college education behind him when Linux debuted. In addition, Torvalds's predecessors had access to many resources that he did not, including powerful hardware, sophisticated debugging tools, and on-site colleagues with whom they could share ideas. In contrast, Torvalds worked from a low-cost 386 computer in his Helsinki

apartment, using only the coding tools available to him on Minix—which included many vital GNU utilities but not the extensive software resources of an institution such as Bell Labs, MIT, or Berkeley. In this respect, Torvalds's ability to produce a working kernel was truly remarkable—so much so that, as the next chapter explains, Microsoft-funded researchers in the early 2000s accused Torvalds of plagiarizing the Linux code by claiming that it was simply impossible for him to have produced all of it himself.

LINUX GROWS UP

On September 17, 1991, about three weeks after Torvalds announced his kernel project on Usenet, the first version of the software, 0.01, became publicly available. The release arrived thanks to Ari Lemke, an administrator at the Helsinki University of Technology, who posted the Linux source code to the university's FTP server, where anyone could download it freely.[57]

In a decision that exerted a much more enduring effect than anything in the actual Linux 0.01 code, Lemke uploaded the source into a directory named *put*/OS/*Linux* on the FTP server he administered. That proved to be important because users came to know the kernel by the name *Linux*, which was not the one Torvalds initially intended to give his software publicly. Although Torvalds had stored the kernel code on his own computer in a directory called *linux*, he planned to release it to the world under the more jocular and modest name *Freax*—a combination of the words *free* and *freaks* that ended in the letter *X*, like the names of most other Unix-like operating systems. "I didn't want to ever release it under the name Linux because it

was too egotistical," Torvalds wrote after the fact. Yet *Linux* was the name that stuck.[58]

Torvalds did not publicly announce the availability of Linux 0.01 when it appeared on Lemke's server in September. "Instead, I just informed a handful of people by private email, probably between five and ten people in all," he wrote in his autobiography.[59] Meanwhile, he worked on transforming Linux into a standalone kernel that was capable of running a complete development environment. He originally used Minix as the platform on which to write the Linux code, but after destroying his Minix installation by accidentally writing data to the device /dev/hda1 (his computer's hard disk) instead of /dev/tty1 (a terminal), he decided to discard Minix and never look back.[60]

The work toward a standalone kernel was not quite complete when, on October 5, 1991, Torvalds announced the release of Linux 0.02 in his next major Usenet post, this one written with strong conviction and purpose—not to mention Torvalds's characteristically impressive command of the English language. It began as follows:

> Do you pine for the nice days of minix-1.1, when men were men and wrote their own device drivers? Are you without a nice project and just dying to cut your teeth on a OS you can try to modify for your needs? Are you finding it frustrating when everything works on minix? No more all-nighters to get a nifty program working? Then this post might be just for you:-)[61]

The post went on to describe the technical features and limitations of Torvalds's kernel, including the requirement that users have a Minix system because Linux was still not yet a "standalone" kernel capable of running on its own. Linux also worked

only with certain types of hard disk, which not all potential users possessed. Lastly, Torvalds warned, "You also need to be something of a hacker to set it up," emphasizing that Linux was not at all ready for the masses. If users sought a stable Unix-like operating system for 386 computers, he wrote, they should stick with Evans's Minix port.

Torvalds also thought it necessary, when announcing Linux 0.02, to justify his project. Previously, when Linux remained only a personal endeavor, he had not bothered explaining to other people why he thought the kernel was worth developing. In the October 1991 post, however, he declared:

> I can (well, almost) hear you asking yourselves "why?" Hurd will be out in a year (or two, or next month, who knows), and I've already got minix. [Linux] is a program for hackers by a hacker. I've enjouyed [sic] doing it, and somebody might enjoy looking at it and even modifying it for their own needs. It is still small enough to understand, use and modify, and I'm looking forward to any comments you might have.

In Torvalds's own words, the fact that the Hurd was not yet ready to use—and as he joked in another Usenet post a couple of months later, might not appear for "the next century or so:)"—was one justification for writing Linux that would make it appealing to other programmers.[62]

However, the most important part of Torvalds's pitch was that his kernel was written "for hackers by a hacker." He implied in the post that this characteristic was the chief distinction between Linux and Minix. He did not elaborate on exactly what made a hacker's kernel different from the alternatives. But his thinking on this point came through clearly enough in other

parts of the Usenet post where he emphasized that Linux was free to use and its source code was fully redistributable. The post did not mention what Torvalds believed to be the technical deficiencies in Minix as a reason for other programmers to consider using Linux. The message was instead about offering the hacker community something it lacked.

In addition to explaining why Linux might appeal to other hackers, Torvalds used the 0.02 release announcement to solicit their feedback and help for developing the kernel further. "I'm also interested in hearing from anybody who has written any of the utilities/library functions for minix," he wrote. "If your efforts are freely distributable (under copyright or even public domain), I'd like to hear from you, so I can add them to the system."[63]

On this occasion (unlike when Torvalds had requested information about POSIX standards the previous summer), helpful responses proved forthcoming. At first, hackers wrote to Torvalds to offer only "maybe one-line bug fixes."[64] By November 1991, however, many people were emailing him to help develop Linux. These included programmers who sent code to implement new features in the kernel and others who merely troubleshot what Torvalds had written himself. "Each day, the community of Linux users expanded, and I was receiving email from places that I'd dreamed about visiting, like Australia and the United States," Torvalds wrote of the period in late 1991.[65]

Collaborators and supporters whom Torvalds never met in person were also offering cash gifts. Such donations were in step with the tradition of the PC shareware culture of the time, in which it was common practice to send the author of a software

program who gave away his work for free something on the order of $10 or its equivalent as a thank-you.[66]

Torvalds, however, responded to the offers of cash by asking for postcards instead.[67] "I didn't want the money for a variety of reasons," he wrote in his autobiography. One was that receiving praise and feeling that he was collaborating with a community of researchers—even though he had never held any sort of title or formal status as a researcher himself—was more important to him than material reward. "When I originally posted Linux, I felt I was following in the footsteps of centuries of scientists and other academics who built their work on the foundations of others—on the shoulders of giants, in the words of Sir Isaac Newton," Torvalds wrote in 2001. "Not only was I sharing my work so that others could find it useful, I also wanted feedback (okay, and praise)."[68]

Torvalds also explained his aversion to cash donations by noting, "I suppose I would have approached it all differently if I hadn't been raised in Finland, where anyone exhibiting the slightest sign of greediness is viewed with suspicion, if not envy."[69]

Whatever the root of his desire to avoid financial compensation for his work on Linux, the decision underlined how committed Torvalds was to keeping Linux a completely free operating system, distinguishing it from Minix and most other Unix-like platforms of the time. Even voluntary cash donations, in his mind, would have sullied his project.

The enthusiastic responses that Torvalds received for his work on Linux were not universal. The platform had its critics, chief among them Tanenbaum, the computer science professor whose work on Minix had been an important stepping stone

for Torvalds in writing his kernel. On January 29, 1992, Tanenbaum publicly berated Linux as an "obsolete" kernel, primarily because of its monolithic design. Chiding Torvalds for being a lowly student rather than a credentialed computer science researcher like himself, Tanenbaum wrote, "among the people who actually design operating systems, the debate is essentially over. Microkernels have won."[70]

Tanenbaum also criticized Torvalds for having written Linux specifically to run on Intel's 386 computer chips. Tanenbaum wrongly believed these processors would not remain an important part of the hardware market. To Torvalds, who had written Linux on a 386 computer because that was all he happened to have, the attack probably seemed cruel, coming as it did from a computer science professor who faced no such constraints obtaining access to different, more expensive hardware platforms.

"Don't get me wrong, I am not unhappy with LINUX," Tanenbaum concluded. "It will get all the people who want to turn MINIX in[to] BSD UNIX off my back. But in all honesty, I would suggest that people who want a **MODERN** 'free' OS look around for a microkernel-based, portable OS, like maybe GNU or something like that."[71]

For Torvalds, Tanenbaum's public attack touched a sensitive nerve, particularly because it threatened "my social standing" within the hacker community.[72] He responded to Tanenbaum in a Usenet post that made the affair personal, sarcastically writing that, as "a professor and researcher," Tanenbaum had "one hell of a good excuse for some of the brain-damages of minix." He added, "I can only hope (and assume) that Amoeba," another operating system that Tanenbaum had helped to develop,

"doesn't suck like minix does." Torvalds enumerated the reasons that he found monolithic kernels to be superior to a microkernel like Minix. He also argued that Linux would prove easier to port to other hardware platforms than Minix because Linux conformed to POSIX standards.

The sparring between Torvalds and Tanenbaum continued for several days. But Torvalds apologized to Tanenbaum for having initially written with "no thought for good taste and netiquette." Other users of Minix and Linux weighed in as well, and a majority supported Torvalds. Although some Linux sympathizers agreed with Tanenbaum that microkernels were superior, several stated that they would still opt to use Linux over Minix because the former was free of cost and available with full source code. "There are really no other alternatives other [sic] than Linux for people like me who want a 'free' OS," one user informed Tanenbaum. In the words of another, "for many people Linux is the OS to use because it's here now, is free and works." The writer added that another benefit of Linux was that it was possible to run the kernel "without paying $$ and/or asking permission from someone."[73]

Neither Tanenbaum nor Torvalds won the debate. Their final correspondence occurred near the end of the year, when Tanenbaum emailed Torvalds to notify him that someone had posted an ad in *Byte* magazine advertising a commercial version of Linux. Tanenbaum asked if that was the sort of thing Torvalds wanted to happen to his kernel. "I just sent him an email back saying Yes, and I haven't heard from him since," Torvalds wrote in 2001.[74]

The criticism that Linux faced from some quarters did not stunt its rapid growth. On the contrary, thanks in large

part to the help Torvalds received from fellow programmers who contributed code over the Internet, it gained features and functionality rapidly. By the second week of October 1991, even before Torvalds had completed the transition to making Linux a standalone system, it was capable of running a variety of GNU programs, including the GNU compilers, assembler, make, tar, bash, sed, and awk and a version of Emacs.[75] This was an important step because it meant that Linux had become a kernel capable of supporting a basic distribution of GNU's software suite.

By January 1992, when Torvalds released version 0.12 of the kernel, it had gained page-to-disk support, which allowed the operating system to use hard-disk space to supplement a computer's physical RAM memory.[76] This feature, which Torvalds implemented in response to a request from a Linux user in Germany who wanted to be able to execute code that required more physical memory than his computer contained, was the first that truly set Linux apart from Minix because it added low-level functionality that was entirely absent from Tanenbaum's kernel.[77] Linux 0.12 became "the first release that started to have 'non-essential' features, and [that was] being partly written by others," Torvalds recalled in 1992. "It was also the first release that actually did many things better than minix, and by now people started to really get interested."[78]

More interest fueled an even faster pace of development. By the spring of 1992, a mere eight months after Torvalds had announced on Usenet that he was writing a Unix-like kernel, Linux 0.95 debuted, bringing with it support for graphical applications via the X Windows System.[79] The version number suggested that release 1.0 of the kernel, which would signal that

Torvalds deemed it sufficiently mature for production-level use, was tantalizingly close.

But the path to Linux 1.0 proved more difficult than Torvalds envisioned. By the spring of 1992, the only core component of the kernel that developers had yet to implement fully was support for networking, which would make it possible for computers running Linux to connect to the Internet. Although Torvalds did not think that it would take a great deal of time to develop the networking code, it was not complete until late 1993.[80]

Early the next year, with the networking support in place, Linux 1.0 finally debuted. The University of Helsinki, where Torvalds was still studying and working, organized a launch event. "We got access to the auditorium, and the head of the CS department gave a speech, and all this gave us enough credibility that there was a fair bit of interest from mainstream media," recalled Lars Wirzenius, one of Torvalds's friends and a fellow programmer: "There was even a television crew, and the footage is occasionally found in various places on the Internet. During the speeches, we had a ceremonial compilation of the 1.0 kernel running in the background."[81]

As the Linux code base expanded, so did the kernel's user base. Complete quantitative data on the number of people running Linux during this time period is elusive (as it remains for most segments of the FOSS world today) because most FOSS software does not require purchase or registration, which makes tracking users difficult. Nonetheless, Torvalds told reporters in March 1994, "I guesstimate a user base of about 50,000 active users: that may be way off-base, but it doesn't sound too unlikely."[82]

COMMERCIALIZING LINUX

The expansion of commercial activity related to Linux even before the release of version 1.0 was another sign of the kernel's rapid adoption. "During 1992 the operating system graduated from being mostly a game to something that had become integral to people's lives, their livelihoods, commerce," Torvalds noted.[83] "There were all these budding commercial companies that had started to sell Linux."[84]

With the 1.0 release, active efforts to promote Linux emerged. These developed organically within the growing Linux community and did not involve Torvalds himself, who remained chiefly interested only in development of the kernel.[85] At the same time, by January 1994, at least one company, Connecticut-based Field Technology, Inc., had begun "selling 'Linux machines' using only copylefted & public domain software," the GNU project reported to its followers. It added, "The Unix-compatible systems are shipped ready to run, with popular programs such as TeX, Emacs, GNU C/C++, the X Window System, & TCP/IP networking. Field Technology makes a donation to the Free Software Foundation for each system sold."[86]

By 1994, the commercial potential of Linux had become so clear that one opportunistic entrepreneur from Boston, William R. Della Croce Jr., who had played no role in Linux development and was unknown to the people involved in the project, attempted to steal the Linux trademark. Initially, the Linux community had not secured a trademark for the operating system's name because, as Torvalds explained in the spring of 1994, "nobody really found the thing important enough to bother about (especially as it does require both some funds and

work)."[87] That created an opening that Della Croce exploited on August 15, 1994, when he quietly filed for Linux trademark rights in the United States.[88] The action did not come to the attention of the Linux community until early 1995. When Torvalds and other developers realized that someone with no connection to Linux development might use a legal trick to take over the project, "there was some panic," Torvalds recalled.[89]

Because the Linux community at the time lacked the organizational structure and cash that would be necessary to respond to Della Croce, a consortium of companies with a stake in Linux's future pooled their resources to fund Linux International, a nonprofit association that Patrick D'Cruze had founded to promote Linux before the trademark affair. Under the direction of John Hall, Linux International took the trademark case to court and, in August 1997, settled with Della Croce. The latter agreed to transfer the Linux trademark to Torvalds himself, who, as "benevolent dictator" of the Linux community, promised not to use his control of the trademark to thwart Linux development or business related to it.[90]

Settling the trademark case in the United States did not protect Linux in other countries, however. As Hall reported, "all around the world people were getting the same strange idea." He added that Linux supporters "can't afford to go to the 200 countries around the world and buy trademarks, and maintain them, so we have to fight them on a case by case basis."[91] That is what happened. By 2007, Linux International had spent $300,000 protecting the Linux trademark from prospectors around the globe.[92] Despite such challenges, Linux managed to thrive in the business world during the 1990s, as the next chapter explains.

THE ART OF LICENSES

When Torvalds released the first versions of the kernel, Linux's commercial success seemed not only improbable but flatly impossible. That is because he initially protected his code, staring with Linux 0.01, with a copyright notice that stipulated the following:

> This kernel is (C) 1991 Linus Torvalds, but all or part of it may be redistributed provided you do the following:
>
> — Full source must be available (and free), if not with the distribution then at least on asking for it.
> — Copyright notices must be intact. (In fact, if you distribute only parts of it you may have to add copyrights, as there aren't (C)'s in all files.) Small partial excerpts may be copied without bothering with copyrights.
> — You may not distribute this for a fee, not even "handling" costs.
>
> Mail me at "torvalds@kruuna.helsinki.fi" if you have any questions.[93]

The original Linux license "was just me writing things up," Torvalds explained in 2016, adding that "there was pretty obviously no actual lawyerese or anything there."[94] The crude copyright amounted mostly to a less sophisticated version of the GPL. It required users of the kernel to accompany distributions of the software with full source code and to adhere to the same licensing terms if they modified Linux. Yet it differed in a key respect. Torvalds forbade distributors of Linux from profiting in any way from the kernel—even if the charges they imposed were intended simply to recoup the "handling costs" they incurred.

These terms reflected how profoundly opposed Torvalds was in 1991 to the idea of charging money for software.

By early 1992, however, Torvalds was rethinking the Linux copyright. Spurred on by Linux users who distributed the kernel at trade shows on floppy disks and asked Torvalds if they could charge nominal fees to cover the cost of their materials, he came to believe that "as long as people gave access to source back, I could always make [Linux] available on the internet for free, so the money angle really had been misplaced in the copyright" that he originally used.[95] He grew even more comfortable with the idea of allowing the sale of Linux as the community and code base grew, instilling confidence that "momentum had been established" and developers "couldn't possibly veer away from our trajectory," Torvalds recalled in his autobiography.[96]

As Torvalds considered adopting a new license for Linux, the GPL was the obvious option. There were other popular licenses in the free software community at the time, such as the permissive ones that governed BSD. But the GPL most closely resembled the homegrown license Torvalds had originally used, and the GPL governed most of the tools Torvalds had relied on to produce Linux. Beginning with the release of Linux 0.12 in January 1992, Torvalds adopted the GPL for the kernel.[97]

Still, despite the GPL's similarity in intent to the original Linux license, Torvalds has never expressed deep satisfaction with it. That is not because he deems the GPL and GNU too constraining. On the contrary, as he noted in 1992, Linux's original copyright notice "was in fact much more restrictive than the GNU copyleft" because of the clause preventing people from making money off of the code in any way.[98] Nonetheless, he has criticized the "hard-core GPL freaks, who argue that every new

software innovation should be opened up to the universe under the general public license."[99]

Torvalds also continued for years after adopting the GPL to express concern that someone might charge for Linux or be punished for obtaining a commercial distribution without paying. As he wrote in 2001:

> Generally speaking, I view copyrights from two perspectives. Say you have a person who earns $50 a month. Should you expect him or her to pay $250 for software? I don't think it's immoral for that person to illegally copy the software and spend that five months' worth of salary on food. That kind of copyright infringement is morally okay. And it's immoral—not to mention stupid—to go after such a "violator." When it comes to Linux, who cares if an individual doesn't really follow the GPL if they're using the program for their own purposes? It's when somebody goes in for the quick money—that's what I find immoral, whether it happens in the United States or Africa. And even then it's a matter of degree. Greed is never good.[100]

Even though Torvalds came to believe in 1992 that it made pragmatic sense to adopt the GPL in order to make it possible for distributors to charge money for Linux under certain circumstances, his abiding unease with software that costs money has never fully dissipated.

Ultimately, however, although Torvalds admitted in 1994 that "I'm not fanatic[al] about the GPL," he recognized that "in the case of linux it has certainly worked out well enough."[101] He attributed that success in large part to the fact that, in the Linux world, cooperating with the copyright terms and the development community provides benefits that outweigh those of attempting to subvert the GPL's terms. According to Torvalds,

"It is the people who actually honor the copyright, who feed back their changes to the kernel and have it improved, who are going to have a leg up. They'll be part of the process of upgrading the kernel. By contrast, people who don't honor the GPL will not be able to take advantage of the upgrades, and their customers will leave them."[102]

GNU AND LINUX

Torvalds's decision to license his kernel under the GPL aligned the code in an important way with the GNU project. In other respects, however, harmony between the Linux community and GNU developers remained elusive during the 1990s.

Part of the unease between the GNU and Linux camps stemmed from the rapid rise of commercial activity related to Linux by fledgling companies whose operations, in the eyes of some hackers, threatened to tarnish the image of free software as the refined, sophisticated endeavor that GNU was pursuing. Ian Murdock, a strong supporter of both GNU software and the Linux kernel, noted that in the early 1990s, "You'd flip through Unix magazines and find all these business card-sized ads proclaiming 'Linux.' Most of the companies were fly-by-night operations," which produced embarrassing implementations of Linux-based systems.[103] From this perspective, Linux appeared to be a poor tool for helping to fulfill the GNU vision of freeing Unix and saving hacker culture.

Such concerns inspired Murdock to float the idea on comp.os.linux (a Usenet group that developers had recently created so that discussions of Linux would have their own home rather than having to use the Minix newsgroup) of building a

hacker-friendly operating system that combined the Linux kernel with GNU programs. Stallman responded to Murdock's post and indicated, according to Murdock, that "the Free Software Foundation was starting to look closely at Linux and that the FSF was interested in possibly doing a Linux system, too."[104] With the Free Software Foundation's support, Murdock in 1993 began building Debian GNU/Linux, which today remains a popular Linux and GNU-based operating system.

The partnership between the Free Software Foundation and Murdock surrounding Debian was significant because, until that time, Stallman and other GNU developers had shown little interest in Linux. That was due partly to the fact that, until Torvalds adopted the GPL, the Linux kernel was no more useful to the GNU project than BSD's alternative or any other software that could not be distributed under terms that the Free Software Foundation approved. Stallman also placed little stock in Linux early on because a friend who had reviewed Torvalds's code concluded that it would be difficult to port the kernel to hardware other than the Intel 386 platform.[105] Because GNU developers in the early 1990s remained uninterested in supporting microcomputers, as the previous chapter notes, a kernel that ran only on 386 PCs was of little relevance in their eyes. For them, Hurd remained the only obvious solution to implementing a complete free software operating system.

Gradually, however, the Free Software Foundation's interest in Linux grew. The GNU newsletter mentioned Torvalds's kernel for the first time in June 1992 in a brief paragraph within a long section deep in the newsletter containing updates on free software for microcomputers. Describing Linux as a "free Unix system for 386 machines" that was "named after its author, Linus

Torvalds," the newsletter went on to note the kernel's ostensible limitations—that it "runs only on 386/486 AT-bus machines, and porting to non-Intel architectures is likely to be difficult as the kernel makes extensive use of 386 memory management and task primitives." GNU developers did, however, inform readers which servers they could connect to in Europe and the United States to download the Linux code. The notice did not explicitly mention that Linux was licensed under the GPL, but it implied as much by calling it "free."[106]

The GNU newsletter continued to provide basic information on Linux for the next two years, but not until June 1994 did Linux begin featuring more prominently in GNU's announcements. At that time, the project informed followers that GNU developer Arnold Robbins would be authoring a regular column called "What's GNU?" in the new *Linux Journal*, thereby helping to ensure that GNU's name would remain a part of the discussion surrounding Linux.[107]

The launch of the Debian GNU/Linux distribution, which GNU developers pitched in 1994 as "a complete, full-featured system based on GNU and Linux that is easy to install and configure," also strengthened the Free Software Foundation's role in the evolving Linux world.[108] The relationship between Debian and the Free Software Foundation expanded in the spring of 1995, when the Debian distribution on CD-ROM became available through GNU's official software distribution service.

The Free Software Foundation also took clear note by 1995 of the efforts that were underway by that time to port Linux to hardware platforms other than the 386. The organization announced to GNU newsletter readers in January that a port of

Linux to the Motorola m68k architecture for Amiga and Atari computers was already in testing, adding that versions for the AlphaPC and MIPS processors were in progress.[109] GNU developers' awareness of Linux's growing suitability for a broader range of hardware platforms suggests that, by this time, they were giving the kernel more serious consideration as a long-term stand-in for the Hurd.

By early 1996, their position on Linux had grown even firmer. The January GNU newsletter noted, "The Hurd is not yet ready for use, but in the meantime you can use a GNU/Linux system."[110] Hurd remained the centerpiece of the GNU vision, but Linux had emerged as an increasingly appealing substitute. Since, as the previous chapter notes, Hurd development was never completed, Linux remains the kernel that powers most GNU-based operating systems today.

Despite early friendly cooperation, relations between the Free Software Foundation and the Linux community grew more tense starting in the summer of 1996. The shift resulted from decisions by both sides. For one, the Debian project changed leadership, with Murdock stepping down and Bruce Perens signing on in his place.[111] Under Perens, the Debian team decided to forgo the sponsorship from the Free Software Foundation that the project had received previously. The reason, Perens said, was that "I decided we did not want Richard [Stallman]'s style of micro-management."[112] GNU developers insisted at the time that they and the Debian team had parted ways "amicably," yet they also stated that they "wish the situation were otherwise" and were considering ceasing to distribute Debian CD-ROMs.[113]

Also in the summer of 1996, Stallman first went on the offensive against usage of the word *Linux* to describe an entire operating system that combined Torvalds's kernel with the GNU suite of programs. He advocated for *GNU/Linux* instead. Murdock had introduced this term, at Stallman's request, to the free software community following Debian's launch.[114] (Stallman had initially suggested the word *Lignux*, but hackers rejected that idea.)[115] Whether out of ignorance, haste, or active hostility toward the GNU project, however, some developers and users of free software referred to GNU/Linux distributions simply as *Linux*. That was a problem, Stallman wrote in an essay on the subject in July 1996, partly because such usage failed "to give the GNU Project credit for making the free Unix-like system that it set out for a decade ago." But he added that

> there is a more important reason for friends of GNU to use names like "Linux-based GNU system" instead of "Linux system." This is to help spread the GNU Project's philosophical idea: that there is ethical importance in freeing users to share software and cooperate in improving it; that free software belongs to a community, and people who benefit from the community should feel a moral obligation to help build the community when they have a chance.[116]

Stallman's words made clear that, to him, the greatest danger posed by people who did not afford full credit to GNU's role in making Linux-based operating systems possible was that they undercut the Free Software Foundation's effort to protect the community of hackers for whom Stallman had launched his crusade.

The essay quoted above recalled the 1983 announcement of the GNU project, when Stallman said that the "Golden Rule" required developers to share source code. But the language about ethics and moral obligations that Stallman deployed in 1996 elevated his rhetoric to a new level. For many supporters of free software, Stallman was laying the ideology on too thickly. Ultimately, stances such as the one he took in the 1996 essay contributed to the fracturing of the hacker community into "free software" and "open source" camps, which chapter 5 details.

The Free Software Foundation continued over the following years to denounce what it saw as betrayals of the true purpose of free software. In 1997, Stallman lamented in an essay that some supporters of Linux did not endorse the free software philosophy or the GPL at all:

> A conference was held this year on the topic of developing "Linux applications." This conference was about using the GNU system, but the conference announcement did not mention the word GNU. Instead of encouraging users to write more free software, it did just the opposite. It included a panel entitled, "Licenses and licensing—I don't want to give away my application!!!"[117]

By the spring of 1998, Stallman had come to accept that Linux was likely to remain the kernel that most people would use to build free operating systems for the foreseeable future. He acknowledged that year that completing the Hurd had proved "a lot harder than we expected, and we are still working on finishing it," but added, "fortunately, you don't have to wait for it, because Linux is working now. When Linus Torvalds wrote Linux, he filled the last major gap. People could then put Linux

together with the GNU system to make a complete free system: a Linux-based GNU system (or GNU/Linux system, for short)."[118]

In the same essay, Stallman remained harshly critical of people who left the *GNU* out of *GNU/Linux*. Indeed, he went on the counterattack by suggesting that the word *GNU* alone might fairly be used to describe operating systems that included the Linux kernel. That usage, he argued, was valid because statistics showed that GNU software accounted for nearly ten times as much of the total source code of a GNU/Linux distribution of the time as the Linux kernel. "So if you were going to pick a name for the system based on who wrote the programs in the system," he concluded, "the most appropriate single choice would be 'GNU.'"

Significantly, in this essay—which Stallman wrote just as the term *open source* entered hackers' lexicon and some major free software developers began distancing themselves publicly from the Free Software Foundation—Stallman no longer portrayed correct usage of *GNU/Linux* as a moral or ethical issue. His main argument for giving full credit to GNU developers for their software was instead that they had done most of the work required to make possible a free operating system that used the Linux kernel. Following public suggestions that the Free Software Foundation had become overly ideological, Stallman seemed to decide to temper his rhetoric for strategic purposes.

If that was the case, the toning down of Stallman's rhetoric occurred too late. As chapter 5 shows, the split between the free software community that Stallman led and the open source group that grew up around Torvalds was too divisive by the spring of 1998 to have a clear resolution. Even worse from

the perspective of the GNU camp, observers of the Linux revolution would soon be calling Stallman a "forgotten man."[119]

WHY DID LINUX SUCCEED?

Why did Stallman and the GNU developers, who had breathed the first life into the FOSS movement and written most of the code that made FOSS possible, end up on the sidelines as Linux exploded in popularity? How did a cheeky undergraduate end up—in the eyes of many hackers, at least—inheriting the fruits of the revolution that Stallman had launched?

These were outcomes that few people could have predicted. In August 1991, it would have been almost absurd to expect that the kernel Torvalds had just announced to the world would ever find a large following outside his Helsinki apartment or power millions of computers across the planet. Other people—including Tanenbaum, the BSD programmers, and the GNU developers—were in the process of building freely redistributable Unix-like kernels or had already completed them. By the standards of the time, their kernels were much more technically impressive than the one Torvalds aimed to create. Those other programmers also had many more credentials to their names than the Finnish undergraduate who created Linux, and they enjoyed immensely larger development budgets.

It was telling that Salus's monumental study of the history of Unix-like operating systems, which he published in 1994, devoted only a handful of sentences to the Linux kernel.[120] As late as that year, even someone as tuned into the ecosystem of Unix-like operating systems as Salus did not foresee how important Linux would become. Yet barely five years after Torvalds

announced his project on Usenet, Linux had evolved into the kernel that, by all realistic measures, had beat out all of the better-funded, more elaborate alternatives.

Historically, most observers of the FOSS community have attributed Linux's success to fortunate timing. Wirzenius, the Finnish programmer who shared an office with Torvalds at the University of Helsinki, wrote in 1998, "The success of Linux wasn't automatic, and things might well have gone differently. For example, 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."[121] Tanenbum made a similar point when he noted that the legal troubles surrounding BSD "gave Linux the breathing space it needed to catch on."[122]

To a significant extent, such interpretations account for Linux's improbably rapid and widespread growth during the early 1990s. Uncertainty over the legality of kernels derived from the BSD code base stunted their adoption, creating an opening for Linux that might not otherwise have been so broad. Meanwhile, as Torvalds himself noted, he likely would not have bothered to develop Linux at all if the Hurd had been closer to completion or if Minix had worked better on his PC.

Yet good timing alone does not fully account for Linux's ability to bloom during its first years. Although BSD's legal battles were significant, the Unix Systems Labs lawsuit did not begin until January 1992. By that time, Linux already had a small following, which extended beyond Torvalds's personal circle. Moreover, the BSD case was settled at the start of 1994, just as Linux 1.0 made its debut. Linux at that time was no more sophisticated in technical terms than the free BSD kernels,

which at any rate supported a broader range of hardware platforms. Plus, for most users BSD and its derivatives were also just as free of cost as Linux. The legal troubles between 1992 and 1994 certainly helped to push developers and users away from the BSDs and toward Linux, but the differences between these two options were not great enough to make Linux the only obvious choice by the time the dust had settled in the courtrooms.

On a similar note, Linux faced its own legal uncertainties during the trademark case that began in 1994 and lasted until 1997. Della Croce's attempt to usurp the Linux trademark was not as dangerous for the Linux kernel as the challenges over code ownership were for BSD. But it was still a serious concern. Despite this fact, it did not lead to a massive desertion from the Linux community in the years before Torvalds obtained the Linux trademark. It did not even stop companies from taking a serious commercial interest in Linux. This outcome showed that there was something intrinsically different about Linux, which emerged relatively unscathed from the same genre of legal trouble that had placed the BSD community in crisis.

What was that difference? Above all, licensing. The BSD license was in a sense even freer than Linux's GPL. As noted above, the BSD licensing terms allowed developers essentially to do whatever they wished with code derived from BSD's code base, even if they did not make the source code of derivative software publicly available. For that reason, BSD might have seemed more attractive for developers than Linux because it was more flexible. Yet because the FOSS camp's chief goal was to nurture hacker values, not develop software with as little fuss as possible, the GPL-licensed Linux kernel, which respected hacker mores, proved more attractive than BSD code to many hackers.

It mattered, too, that the Linux kernel code never cost a penny. That set it apart from most of the other freely licensed Unix-like kernels of the early 1990s. NET 2, the first complete BSD-based operating system that did not require users to purchase a Unix license from AT&T, could be legally copied and distributed by individuals without paying, but an official copy from Berkeley itself cost $1,000.[123] Some of the other BSD derivatives were similarly priced.[124] Minix cost a fair amount of money, too, as Torvalds bitterly noted.

The role of cost in Linux's success should not be overstated. By 1992, several BSD-based systems that cost no money, no matter how users obtained them, were in circulation. And again, it was always possible for users to obtain NET 2 without cost if they copied it themselves. Yet Torvalds's adamant opposition during Linux's early days to charging money for the kernel made his project a true outlier. It also probably augmented Linux's appeal for hackers who, having witnessed the troubles that ensued when AT&T commercialized Unix in 1983, were wary of any activities by software distributors that smacked of commercialism—even if the distributors also offered ways to obtain code without paying.

Lastly, the developer community that Torvalds cultivated helped to ensure the kernel's rapid growth. To an extent, as noted above, programmers were attracted to Linux because of uncertainty over BSD's legal future, which sapped the BSD community of its momentum and made some programmers wary of contributing code. Yet programmers who liked Linux's nonexistent price tag and the openness of its code began helping Torvalds write the kernel before the BSD lawsuits commenced. On October 10, 1991, several months before the start of the

first BSD case, Torvalds wrote that his kernel "never would have seen the light of day or would have been much worse without the help of some others."[125] He went on to name collaborators who were helping him develop Linux via the Internet. By the time of the Linux 0.11 release in late 1991, "a small following" of programmers had arisen, according to Torvalds.[126] From there, the community of Linux developers continued to grow.

Torvalds was not the first programmer to endorse 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 without AT&T code in preparation for the release of NET 2.[127] And according to Tanenbaum, Linux followed "essentially the same development model as MINIX," which also received contributions of code from users who wished to add new features.[128]

Yet Torvalds's project was different in several key ways from other collaborative, Internet-based development efforts. First, because Linux was completely free of cost, programmers who donated code to Torvalds could reap what they sowed without having to pay a penny. They could not do the same in the case of Minix. Second, Linux came of age as email access became widespread and the Internet ceased "being an enclave of a few research universities," as Torvalds recalled.[129] That lowered the barrier to participation.

Third and most important, at least in Torvalds's own view, was that the community of programmers who contributed to Linux early in its history was not based at a single institution or derived from a core group of original participants. "There was no historical insider group," Torvalds told me, "so we were a

lot easier to approach if you came from a DOS/Windows background, for example." He added, "there was no cabal, it was easy to send me patches, I wouldn't have stupid paperwork rules like a lot of other projects had, and it really was a much more open project than a lot of software projects that preceded it."[130]

Not until the late 1990s, when Eric S. Raymond published work on the "bazaar" mode of software development, did FOSS developers fully recognize and articulate the novelty of the decentralized, Internet-based Linux development model. And it was only in 2008, with the introduction of GitHub, which reduced barriers to FOSS contribution even more than email-based development had, that open collaboration on FOSS code saw its complete incarnation. Yet the effects of the model Torvalds helped to pioneer were apparent much earlier. They were why observers noted in the spring of 1994, "The number and frequency of new releases of Linux, and drivers and utilities, are amazing to anyone familiar with traditional UNIX development cycles."[131] The approach Torvalds adopted in 1991 remains highly influential today, when most major FOSS projects—from the OpenStack cloud computing platform to the Linux Foundation's various "collaborative projects"—follow the same model.

A variety of factors combined to make Linux into the sophisticated, feature-rich kernel that it became by the mid-1990s. From there, they continued to fuel the remarkable expansion of the Linux ecosystem, which, as the next chapter shows, assumed outsize importance within the computing industry by the end of the decade.