Mac for Linux Geeks

Copyright © 2009 by Tony Steidler-Dennison

ISBN-13 (pbk): 978-1-4302-1650-6

ISBN-13 (electronic): 978-1-4302-1651-3

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `http://www.apress.com/info/bulksales`.

CHAPTER 1

The Backstory

The focus of this book is migrating from Linux-based systems to Mac OS X. To lay the foundation for the information to come, a little history is in order. This chapter provides a brief history of UNIX, BSD, and Mac OS X.

Of Macros and Manuals: UNIX

The creation of Mac OS X really starts with the creation of UNIX. That story is well known, especially among Linux geeks—a group that owes a great debt of gratitude to the work of Dennis Ritchie, Ken Thompson, and a team of Bell Labs engineers. In the summer and early fall of 1969, these engineers cobbled together a rough operating system based on the Multiplexed Information and Computing Service (MULTICS) operating system. MULTICS was a project taken on jointly in 1966 by Bell, General Electric, and Massachusetts Institute of Technology (MIT), but dropped in 1969. Although MULTICS would continue as a commercial venture until 2000, its life in the labs was limited. However, its contribution to the world of computing cannot be overestimated. It produced the team of engineers that, during those heady days of 1969, would create the UNIX operating system.

Like so many technical projects, work on the UNIX system began with an informal discussion. Ritchie, Thompson, and fellow Bell Labs engineer Rudd Canaday met to talk about the project in the summer of 1969. The notes from that brainstorming session were phoned to the Bell Labs dictation system, transcribed, and sent to the engineers. These informal notes would become the working concept of operations for the initial version of UNIX.

Over four months following that meeting, work on the UNIX system rolled forward. A rough filesystem was created on the PDP-7, a system that, at its creation, was state of the art. The engineers, primarily Thompson, created the operating system, shell, assembler, and editor in just four weeks. They also developed a set of tools that would be accessible to users on the system, including tools to copy, print, and delete files. This core tool set was created with the General Electric Comprehensive Operating System (GECOS)—a system still in limited use on servers and mainframes today. The tools were then transferred to the PDP-7 using paper tape. With the assembler—the final piece of the original system—successfully transferred to the PDP-7, the fledgling UNIX system was no longer reliant on GECOS. UNIX was completely self-contained, with the full capabilities to develop and build new tools for the system included as part of the system itself.

From Assembly to C

The year 1970 would prove to be one of high activity for the UNIX team at Bell Labs. Concurrent to the development of the early UNIX framework on the PDP-7, Bell Labs had acquired an upgrade to the PDP-7 system: the PDP-11. As components of the PDP-11 rolled into Bell Labs, the UNIX system was ported to this new platform, taking full advantage of its extended capabilities.

Originally written in assembly language, the new system was simultaneously ported to the word-based B language. It was quickly adopted by the Bell Labs patent department to process the reams of applications and research the office churned through each month. But the B language was limited in its use of data structures—code objects containing fields, items, members, and/or attributes that describe the operation of the code itself. These structures exist as abstractions of actual code operations, and they help to allow developers to work at an increasing distance from assembly language. In short, the creation and existence of data structures can dramatically reduce the time and the number of errors in developing subsequent code.

To further streamline the development process, Ritchie and Bell engineer Brian Kernighan developed a set of data structures to overlay on the B language. The resulting powerful language, C, bore little resemblance to its predecessor. To this day, C serves as the core of many operating systems and applications. It also serves as the basis for such widely used languages as C++, Perl, and Java. The scope and philosophy of C have been carried on in countless other languages and environments.

Macros and Pipes

Two more milestones in the development of UNIX were accomplished in the years between 1970 and 1975. The first was, in essence, another modernization of an older computing idea—the concept of macros. Like data structures at the code level, macros contained a set of actions and operations that could be executed by users and developers with minimal keystrokes. The overriding idea was to group these sets of tasks together in a series of operations initiated by a single keystroke. Also like data structures, these operations could be created and tested discretely, assuring error-free operations when combined.

Macros did not exist in the early iterations of the C language. Due to the growing complexity and power of that language, macros for it were more difficult to create, because that power and complexity demanded a similar level of power in macro-like operations. Ritchie and Kernighan approached this problem head on, creating a concept that would truly distinguish the UNIX operating system from others of its day and from most that followed.

Rather than creating new code for macros, Ritchie and Kernighan envisioned a concept that would allow the output of one existing command or tool to be passed as input to another. This concept efficiently leveraged the previous work of creating the individual system tools, eliminating duplicate effort. More important, it also created a seemingly infinite number of tool combinations. Any tool could perform its discrete operations, and then seamlessly pass the result of those operations as input to any other tool for further processing and output—perhaps to yet another tool, if necessary. In effect, the concept created system “glue” capable of tying many tools to many others as required. Ritchie and Kernighan called this glue *pipes*.

In practice, pipes were revolutionary. Pipes gave users power and flexibility that simply could not be achieved with mere macros. They also had an interesting side effect on subsequent UNIX development: they narrowed the scope of new tools to single tasks. What would become the hallmark philosophy of UNIX systems was born in that reduced scope: "Do one thing, and do it well." The implementation of pipes allowed developers to write programs that performed a single task well, and then to tie those applications, as necessary, to others created under the same philosophy. At the highest level, the use of pipes encouraged developers to create system and user tools that worked well together. Many would later make the case that this single concept was the genesis of the legendary flexibility, stability, and reliability of overall UNIX system performance.

User Manuals

One more significant internal milestone lay ahead in the steady progression of UNIX from esoterica to mainstream use. Engineer Doug McIlroy took on the task of creating a manual that would fully expose the operation of the system to those on the other side of the development and engineering fence—the users.

McIlroy appreciated the shortcomings of existing user manuals. For the most part, they didn't go far enough in explaining the operations and options of the systems. And they didn't present that minimal information in a way that was readily accessible to the end users. McIlroy believed Bell Labs could do a much better job with its user manual and would, by that effort, fully distinguish its operating system from other burgeoning operating systems. In some sense, McIlroy took on the creation and management of the UNIX manual with a unique mix of pride of ownership and commercial savvy.

The result opened the eyes of the computing world to an approach that seemed almost paradoxical. It was at once revolutionary yet born in pure common sense. For starters, the UNIX manual was incorporated into the operating system itself. It was fully accessible within the very environment in which questions about the system's use would arise. The `man` command, while strictly adhering to the philosophy of doing "one thing right," also put the full knowledge of the system's creators at the fingertips of its users. And that full knowledge was breathtaking in its scope. Every option for the use of every tool on the system was described. Examples were included where they might clarify a complex implementation. Tool descriptions included a brief summary of the relationship of a single tool to other system tools. And, in a revelation for system users, the limitations of individual tools were described, as well, including existing bugs.

The scope, accessibility, and practicality of the UNIX manual made it possible for near-novices to learn the system quickly. True to the business goal, UNIX adoption in business and academia sectors began to escalate at a rate unseen in previous UNIX versions or in rival operating systems.

By 1976, just seven short years removed from the ad hoc meeting that created its concept of operations, UNIX had become the operating system of choice for many companies and schools around the world. Licensing fees for the system reached an all-time high and continued to grow. Through constant revision, the system had reached a previously unsurpassed level of reliability—a key factor in its adoption by increasingly computing-intensive businesses and schools. And it also proved to be highly portable, allowing its adoption on a widening range of hardware platforms.

The Fork: BSD

In a sign of the broad acceptance of UNIX, engineer Ken Thompson was invited by University of California, Berkeley (UCB), to take a yearlong sabbatical from Bell Labs to teach computing at the school. It would serve as a much-needed break from the daily rigors of operating system development. Unknown to the principal players, that sabbatical would mark the beginning of another critical milestone in the history of UNIX. Indirectly, it would also serve as the first real seed for Apple's Mac OS X operating system.

1BSD to 4BSD

Thompson focused his teaching at UCB on UNIX. He found the students to be more than capable of making system improvements and upgrades. Interest in the system ran so high during Thompson's time at the school that students continued to develop the system after Thompson's departure. One, graduate student Bill Joy, took particular interest in the system. Shortly after the end of Thompson's sabbatical, Joy released the first Berkeley Software Distribution, or 1BSD. Rather than a complete system, 1BSD was an add-on to UNIX System 6, including a Pascal compiler and Joy's text editor, ex. Further upgrades and releases followed, including 2BSD, with the C shell and the vi text editor, in 1978.

In the meantime, UCB purchased and installed a Virtual Address eXtension (VAX) computer on the campus. Built by Digital Electronics Corporation (DEC), the VAX was a significant upgrade to the PDP-11 system, including a full step up from 16-bit to 32-bit memory addressing. In order to utilize the larger address space of the VAX, students began what would amount to a full rewrite of the UNIX code.

Incorporating rewritten kernel code, the 2BSD utilities, and other operating system tools, 3BSD was released in 1979. The release was a critical milestone in the development of Berkeley's UNIX-based operating system. It was the first to contain a complete kernel, rather than merely providing a set of add-on tools for UNIX. In reality, 3BSD was the first stand-alone BSD operating system and the first fork of UNIX.

3BSD also represented an important win for UCB. The Defense Advanced Research Projects Agency (DARPA) was so impressed with the 3BSD release that it agreed to fund efforts at UCB's Computer Systems Research Group (CSRG) to create a standard UNIX platform for future DARPA research projects.

BSD releases into the 1980s continued to refine and add new tools to the system. Major releases occurred in 1980 (4BSD) and 1981 (4.1BSD). The 4.2BSD version took two years to release in total, including several incremental releases during that period. Of significant interest, the 4.1a release included a complete rewrite of the Transmission Control Protocol/Internet Protocol (TCP/IP) stack originally developed by Cambridge, Massachusetts, acoustic analysis firm Bolt, Beranek and Newman (BBN). Again, one of the foundational pieces of BSD's success was the result of the efforts of Bill Joy. Reportedly disappointed by BBN's TCP/IP implementation, Joy completely rewrote the stack. Later, when questioned in a meeting with BBN about how he accomplished the mammoth rewrite, Joy reportedly responded that the task was simple. "You read the protocol and write the code."

Licensing Issues

By the time of the 4.2BSD release in 1983, significant portions of the BSD code had been completely rewritten or created from scratch. What remained was still licensed by Bell Labs, under the control of AT&T. In 1984, AT&T divested itself of Bell Labs, creating AT&T Computer Systems, with the primary goals of UNIX development and licensing.

Licensing fees for UNIX System V, which included code created at UCB, continued to increase. As a result, many began to push for and investigate the possibility of a BSD-like release that would be freely distributable, without the encumbrance of AT&T's licensing fees. This lead to the release of Networking Release 1 (Net/1) in 1989, which provided only the Berkeley-developed networking tools to those without licenses for the AT&T code.

Work then began in earnest on Net/2, a release that would completely rewrite the UNIX tools using non-AT&T-licensed code. That release, in 1991, spawned two new efforts focused on porting BSD to the Intel architecture: 386BSD, a free version of the operating system based on Net/2, and another version created by Berkeley Software Design (BSDi).

BSDi quickly became a legal target for AT&T's newly formed UNIX System Laboratory (USL) subsidiary. USL successfully pursued injunctive relief against BSDi, preventing the distribution of the system while further legal action determined whether BSDi had violated the USL-held copyright on UNIX System V and its trademark on UNIX. That action prevented the distribution of BSDi's version for nearly two years. It also prevented the further development of BSDi-derived systems, as rights to all BSDi code remained in question.

The USL action was settled in 1994. After two years of litigation, USL signed off on an agreement that would require only 3 of the 18,000 BSD files in the Net/2 system to be removed. An additional 70 files were targeted for modification to show USL copyright notices. Critically, the settlement also stipulated that no further legal action would be taken by USL against BSD developers, users, or distributors. With those minimal modifications, 4.4BSD was released in June 1994.

BSD had won a hard-fought freedom from the strictures of AT&T licensing, but that freedom did not come without a cost. The 1995 release of 4.4BSD-Lite Release 2 would mark the end of formal BSD development at UCB. The university closed the CSRG shortly after the release. But the freely distributable code created at CSRG remained in the public domain under the permissive BSD license. 4.4BSD-Lite Release 2 served as the basis for the BSD projects we recognize today: FreeBSD, NetBSD, and OpenBSD.

The Enthusiast and the Marketer: Apple Computer

In March 1975, as Dennis Ritchie and Ken Thompson worked toward a stable version of the C programming language, a group of electronics enthusiasts met for the first time in Palo Alto, California. While providing an opportunity to exchange information, to discuss the latest advances in electronics, and to swap electronic parts, the Homebrew Computer Club also had an informal mission of making computing affordable to individuals.

Homebrew Days

Meeting in what would later be referred to as the Silicon Valley, just 40 miles south of Berkeley, many of Homebrew's initial members were well aware of the advances in computing taking place on the UCB campus. At the time, the benefits of those advances were targeted at business and academia. The idea of personal computing was limited, for the most part, to technologists and amateur hobbyists like those who gathered to form the Homebrew Computer Club.

At the time of Homebrew's formation, Stephen "Woz" Wozniak had left the Electrical Engineering program at UCB without a degree. A 25-year-old introvert with the dual talents of hardware design and software programming, Wozniak found in the Homebrew club a group of peers with an enthusiasm for computing he might be able to share.

In 1970, Wozniak became friendly with a summer intern at Wozniak's employer, Silicon Valley stalwart Hewlett-Packard. The friend made an increasingly compelling case that a computer could be built and sold on a single circuit board; that such a computer could, in fact, be the basis of a company created specifically to sell computers to individuals, rather than to businesses. Though initially skeptical, Wozniak was eventually convinced that his friend, Stephen Jobs, might be onto something. After ending a brief college career of his own at Reed College in Portland, Oregon, Jobs returned to Palo Alto in 1974, taking a job as a technician at Atari. Jobs and Wozniak became regular attendees and contributors at the Homebrew meetings.

During their time in the Homebrew Computer Club, Wozniak and Jobs achieved several milestones that reinforced their shared view that computers could be built in a size that would fit on desktops. In the first, Jobs enlisted Wozniak's assistance in collecting a bonus initiated by his employer. Atari instituted a program to reduce the number of chips used in its game consoles, offering employees $100 per eliminated chip. Jobs turned to Wozniak for the hardware design, offering a 50/50 split of the bonus. Wozniak quickly reduced the number of chips in the system by 50. Though Atari later paid out only $600 for the reduction of 50 chips, the effort further honed Wozniak's uncanny ability to design for maximum power with minimal hardware resources.

The second was a purely personal milestone for Wozniak. In early 1975, Wozniak read an article in *Popular Mechanics* describing how to build a video computer terminal. At the time, video terminals were rare; most computers were paper-based teletypes. Following the concepts of that article, Wozniak designed and built a 24-line, 40-character-width video terminal from off-the-shelf parts. The terminal was capable of producing 60 characters per second, more than six times the number of printed characters from the existing teletype terminals. Later, to foster his growing reputation among his Homebrew peers, Wozniak built a microprocessor into the terminal, creating, in essence, a complete computer.

In 1976, MOS Technology introduced a new central processing unit (CPU). The $20 6502 chip was a close relative to Motorola's $170 6800, which was the chip Wozniak preferred in his computer designs. Wozniak modified an earlier 6800-based computer design to incorporate the 6502 chip, and took the machine to demonstrate to his peers in the Homebrew club.

Based on this design, Jobs convinced a local computer shop to purchase 50 of the machines. Jobs also managed to secure the necessary parts on credit, paying on net-30 terms when the machines were delivered on time. Eventually, 200 of these machines, dubbed the Apple I, would be built. With the considerable profits on that initial sale, Jobs and Wozniak formed Apple Computer on April 1, 1976. Wozniak left Hewlett-Packard to focus full-time on the Apple II, as Vice President of Research and Development in the new company. Jobs focused his attention on marketing, sales, and fund-raising.

Unlike the histories of UNIX and BSD, which were primarily technical achievements, the story of Apple Computer is equal parts technical wizardry and marketing savvy. It's almost impossible to tell the story of one without telling the story of the other.

Apple I to Lisa

During the early years, both the Apple I and Apple II computers utilized a tape-based version of Wozniak's Apple BASIC. A 256-byte, firmware-resident system monitor served as the operating system, providing users with a command line for running programs. Shortly after the introduction of the Apple II in 1977, Wozniak produced yet another striking hardware design in the floppy disk drive. This made it possible to move the operating system from read-only memory (ROM) to disk, and further spurred the creation and introduction of Apple Disk Operating System (DOS) to the Apple II in 1978.

In 1980, Apple Computer launched an initial public offering (IPO) of stock. That IPO is legendary in tech business circles, as it created more than 300 millionaires on its first day.

1981 brought a shocking turn of events for the small Apple family. In February, Wozniak crashed his airplane in Santa Cruz, California, causing retrograde amnesia. He didn't remember the crash and often had trouble with his short-term memory. Though his memory was restored by late in the year, Wozniak didn't return to Apple until 1983. By that time, his primary interest in the Apple line of products was in product development. Although he continued with the company until 1987, his full-time employment and influence within the company were effectively ended by the crash. The enthusiast who had almost single-handedly created a computing revolution was hardly more than another employee in a quickly growing corporation.

Even without Wozniak, Apple computers would undergo a near constant series of revisions through 1984. The revised machines included the Apple II+, IIe, IIe Enhanced, and III. These systems used a variety of operating systems, including Apple Pascal, Apple SOS, and Apple ProDOS.

In 1983, Apple introduced the Lisa, and another new operating system. Lisa Office System (OS) implemented a set of process-management system calls that bore some resemblance to UNIX. Among the calls were `make_process`, `kill_process`, `activate_process`, and `setpriority_process`. The initial process, `init`, was created by the operating system as the shell process when the machine was booted, serving as the parent for all other processes. Additional processes could be created only by existing processes.

Additionally, the Lisa OS filesystem bore a striking resemblance to UNIX, albeit with a few additional Apple pieces. Like UNIX, the filesystem contained files, folders, disk volumes, and printer and serial devices. The system also supported permissions on a per-file basis.

But the most striking feature of the Lisa was a full graphical user interface (GUI), as inspired by efforts at Xerox's PARC laboratory. In 1979, Jobs and a team of Apple engineers were granted full access to the PARC facilities in exchange for $1,000 of pre-IPO Apple stock. The Xerox PARC system used a method of human-machine interaction referred to as WIMP (for window, icon, menu, pointing device).

The WIMP paradigm provided nontechnical users with an easily understood visual metaphor—that of the desktop, the elements of which were most popularly accessed with a mouse—another first for personal computing. With menus, WIMP also gave users a means to access the full feature set of an application without a requirement to remember each and every element. Though the concepts of WIMP had been known in technical circles since the early 1970s, Lisa OS was the first to implement those concepts in a computing system outside a laboratory. Relying primarily on Lisa Pascal, the WIMP concept implemented by Apple would prove revolutionary.

And Finally, the Mac

The Apple Lisa was ahead of the curve in personal computing on many levels. Unfortunately for Apple, it also required hardware upgrades that pushed the cost of the Lisa to nearly $10,000—hardly a price point that would appeal to the masses.

That same year, Jobs began the search for a more seasoned chief executive officer (CEO) for Apple. The company had ambitious expansion plans in place for the years ahead and needed an executive who could oversee and implement those plans. Jobs landed on John Sculley, then CEO of PepsiCo. While Sculley had never worked in the tech sector, his reputation and resume with PepsiCo was, in Jobs's eyes, a perfect fit.

Sculley had been the youngest Vice President of Marketing in PepsiCo's history and, subsequently, the company's youngest president. He had a keen marketing eye and was unafraid to spend on advertising. He instituted the company's first consumer-research studies. He took on rival Coca-Cola head-to-head with the Pepsi Challenge, a blind taste test that served as the company's successful national marketing campaign in the mid-1970s. The campaign dramatically increased Pepsi's market share. And, at 44, Sculley was still a relatively young man—a fact of some importance to the executives at the young Apple Computer company.

Among the ambitious plans Sculley was tasked to implement was the introduction of yet another computer line, scheduled for early 1984. That computer, the Macintosh, would leverage and advance the GUI concepts first made publicly available in the Lisa. The single 400KB operating system floppy disk was known as Mac System Software.

Aside from the low-level operating code, the Macintosh ROM load contained the Toolbox and the Finder. The Toolbox was a collection of shared libraries used in the applications and made available to developers for use in future Macintosh application development. The Finder was a system browser, allowing users to view the contents of the filesystem and to launch applications. However, use of the Finder required users to shut down other running applications, as the Macintosh was a single-tasking computer. The Macintosh also contained a 7.86 MHz Motorola MC68000 processor; 64KB ROM; 128KB random-access memory (RAM); and an 8-bit, four-voice sound generator, capable of converting text to speech. An all-in-one unit, the Macintosh weighed just over 16 pounds.

To kick off the launch of the Macintosh, Apple hired director Ridley Scott to produce a television ad intended to air only once: during the broadcast of Super Bowl XVIII on January 22, 1984. At a cost of more than a million dollars, the ad portrayed computer users as conformists in dark-gray suits of sackcloth, herded by helmeted police into a large and dark arena beneath a huge video monitor. From the screen, Big Brother lauded the age of information purity and scorned "contradictory thoughts." From the back of the arena, a heroine raced down the center aisle, flinging a heavy sledgehammer through the video screen, leaving the users stunned. The closing caption and voice-over read like this: "On January 24th, Apple Computer will introduce the Macintosh. And you'll see why 1984 won't be like '1984.'"

Two days later, Jobs introduced the Macintosh at the annual Apple shareholders' meeting. He gave a brief prepared speech and a demonstration of the Macintosh's graphics capabilities, and then said, "Now, we've done a lot of talking about Macintosh, lately. But, today, for the first time ever, I'd like to let Macintosh speak for itself."

To a thunderous round of applause, the Macintosh began.

Hello, I'm Macintosh. It sure is great to get out of that bag.

Unaccustomed as I am to public speaking, I'd like to share with you a maxim I thought of the first time I met an IBM mainframe: never trust a computer you can't lift.

Obviously, I can talk, but right now I'd like to sit back and listen. So, it is with considerable pride that I introduce a man who has been like a father to me . . . Steve Jobs.

The applause in the Cupertino theater continued unabated for more than five minutes.

Although the Macintosh was by all accounts a huge success, Apple had difficulty competing with an increasing number of IBM-clone personal computers. CEO John Sculley, intent on driving the company to profit, instituted tighter engineering procedures and reviews, predevelopment marketability studies, and engineering staff reductions. With control of the company he had started now all but ceded to Sculley, Jobs found himself in almost constant conflict with the CEO. By 1985, both Sculley and the board of directors had had enough. Jobs was stripped of his duties as head of Apple's Macintosh division. Nine years after the company's formation, his time and influence had ended.

The Convergence: Mac OS X

Financially buoyed by his time at Apple, Jobs purchased Pixar, a visual effects studio, for $10 million in 1986, and then founded a new company, NeXT, Inc. NeXT would produce the NeXTStep operating system, a UNIX-like system, and the hardware on which it would run.

NeXTStep would eventually serve as one basis of the rebirth of Apple and the Macintosh. However, the convergence of the Macintosh and UNIX actually began in the early 1990s, with a version of AT&T UNIX known as Apple UNIX. Apple UNIX was the operating system used on the Apple Macintosh Quadra machines. The Quadra 700 and Quadra 900 were introduced in 1991 and were built around the Motorola 68040 CPU. With the known reliability of UNIX and powerful 25 MHz CPUs, the Quadras were marketed as high-end professional machines, geared to replace the Macintosh II.

NeXTStep

NeXTStep was a direct descendent of 4.3BSD. Its distinction from BSD rested in its use of the Mach microkernel, originally designed as a drop-in UNIX kernel replacement. Developed at Carnegie Mellon University in 1985, the Mach microkernel provides operating system and application services by calls to user-mode servers. This is in contrast to the monolithic kernel design of BSD, wherein applications obtain services (such as operating system services) by making system calls directly to the service. These services don't exist in the kernel space of the Mach kernel—they exist in user space. The kernel merely routes the interprocess communication (IPC) call to the appropriate server, which then handles the request to the application. The result is that the Mach kernel is a much lighter structure, with a lot of the "heavy-lifting" done in user space. In almost all other respects, NeXTStep was a standard BSD operating system.

Initially, the Mach microkernel was, in fact, slower than the monolithic BSD kernel. When the Mach 3 kernel moved the processes of permissioning and security to the applications, it performed its tasks in only one-quarter the time required by a system with a monolithic kernel. Mach was also designed from the ground up with multitasking and multiple-processor support built in. NeXTStep was quickly accepted by many in the computing community as the next-generation operating system.

NeXTStep was also renowned at the time for its use of the Objective-C language. Objective-C is an extension to the C language that adds object-oriented programming. Unlike other C extensions (C++, for example), Objective-C uses a small-footprint runtime. As a result, Objective-C programs, while powerful, are generally relatively small. Additionally, these programs can be compiled with the GNU Compiler Collection (GCC) compiler, included in all implementations of BSD.

The NeXT computer hardware was equally advanced. The initial machines, released in 1989, were composed of the 25 MHz Motorola 68030 processor, up to 64MB RAM, a 10-Base 2 Ethernet port, a 40MB to 660MB hard drive, and an 1120 × 832, 17-inch grayscale display. The "cube" form factor was also unique, carrying a feel of the forward-thinking design principles required by Jobs at Apple. The machines were widely accepted among the computing cognoscenti. Among their users was Tim Berners-Lee, who used the NeXT computer to create the first web browser and web server in 1991.

While the NeXT computer was accepted as an advanced computing workhorse, its high hardware manufacturing costs, like those of Apple's Lisa, would eventually determine its fate. By 1993, it was clear that profitability was nowhere on the horizon. NeXT returned to its original software-only business plan, changing the name to NeXT Software, Inc., and laying off more than half its employees. A partnership with Sun Microsystems, funded by $10 million cash from the Santa Clara, California, company, kept the software business moving forward, though slowly. The partners created OpenStep, a version of NeXTStep minus the Mach kernel, with a commitment to use the software in future Sun SPARC machines. By 1996, still heavily loaded with debt, NeXT Software was ripe for an acquisition.

Back at Apple

Following Jobs's departure from Apple, the company went through a period in which it seemed, at once, overambitious and underachieving. The company took on some daunting development tasks, beginning with a port of the Mac OS to the new PowerPC platform. The development came at a huge financial cost and bore little practical result. To great fanfare, Apple announced the Copland project, a complete update to the operating system. The rewrite was highly anticipated by Mac users, but was never completed. And the product line was fragmented into increasingly smaller chunks, creating a growing impression of Macintosh computers as simply niche machines.

Interest in UNIX and Linux operating systems did continue during this period. At the annual Apple Worldwide Developers Conference in 1996, Apple announced MkLinux, a microkernel-based Linux system intended to bring Linux to the PowerPC platform on the Mac.

By 1996, Apple faced a severe shortage of cash. CEO Gil Amelio cut the workforce, dropped the Copland project, and went looking for a suitable operating system for the Mac. When negotiations with Be Inc., headed by former Apple employee Jean-Louis Gassée and the creator of BeOS failed, Amelio went to NeXT, a company with a strong product and reputation, but little cash of its own. In 1997, Apple acquired NeXT for a staggering $429 million. The deal also included 1.5 million shares of Apple stock, all of which were awarded to Steve Jobs.

Jobs returned to the company as a consultant, and by late 1997, had replaced the newly ousted Amelio as interim CEO.

Apple had returned to its roots, bringing back its founder to head the company. In 2001, Jobs would remove the "interim" from his CEO title. The company had also found the operating system that would take it into the next century in NeXTStep. Over the next four years, NeXTStep would be ported to the PowerPC platform, while maintaining synchronous Intel builds. Critical pieces—such as Cocoa, the Dock, the Services menu, and the Finder's browser view—were all ported directly from NeXTStep and utilized in the new operating system. Jobs introduced Mac OS X, with its internal BSD and Mach kernel, at the January 2000 Macworld conference in San Francisco, California. Apple had created a twenty-first century operating system by returning to technologies born in the 1970s.

Why BSD in Mac OS X?

Following the acquisition of NeXT and the return of Steve Jobs to the company, NeXTStep began a deliberate metamorphosis to Darwin, the system that would become the core of Mac OS X. While retaining its BSD underpinnings, object-oriented libraries, strong graphics orientation, and development tools, the Darwin kernel was hybridized. The XNU kernel took shape with elements of Mach, FreeBSD, and code created in-house by the Apple team.

The decision to develop NeXTStep around the BSD base and the "plug-in" Mach kernel had already proven to be a wise choice for NeXT, Inc. and NeXT Software. For many reasons, the decision to port that system to the PowerPC platform with its BSD base intact made great sense for Apple, as well. These reasons included BSD's history, portability, open source base, economics, and extensibility.

History

UNIX and its various derivatives had been well known and highly regarded since the late 1970s. It was the operating system of choice for business, academia, and, since the early 1980s, government research programs. The large, active code base made it possible to customize a full operating system for almost any need. The huge code archive also made it possible to modernize and implement some tools that were too esoteric for the existing user base, but that might serve a consumer operating system well.

BSD also had a large and dedicated user base. Bugs in the system were fixed quickly. The code was under constant review and revision by the community. That community, in fact, made sure that each new tool added to BSD underwent thorough testing under the UNIX philosophy that it should "work well with other tools." That established process and history would potentially reduce the development time for the Apple team.

Release timing was important to Apple, as well. A large element of its decline in the early 1990s was due to the company's inability to stick to a regular operating system update schedule. The historical view of BSD clearly proved that this critical release cycle could be established and followed at Apple.

And history had proven the reliability of the BSD system. BSD-installed machines amassed months, sometimes years, of uptime without a reboot. This was often accomplished despite long periods of intense processing. Apple's primary operating system competition in the consumer computing space, Microsoft Windows, was already scorned in many circles for its tendency to crash at critical moments. While stability was certainly important in commerce, academia, and governmental use, the engineers and marketing groups at Apple saw it as no less important at the consumer level. Stability was a competitive advantage, even with low-intensity users. The BSD base of NeXTStep, and later Mac OS X, was known first and foremost for its long track record of power and stability.

Portability

Portability was another important piece of the decision to retain the BSD underpinnings of NeXTStep in Mac OS X. BSD had proven its portability almost from the beginning, In fact, the FreeBSD group members took no small measure of pride in the fact that their Intel version was the only Intel UNIX that could run reliably as a server. The inclusion of FreeBSD had also helped to make NeXTStep and OpenStep fully operable on a number of platforms, including x86, SPARC, and HPPA.

The NeXTStep engineers, many of whom came to Apple with the acquisition, had begun their foray into the hardware world by porting the system to the Motorola 68030-based architecture. Versions of NeXTStep also existed for the reduced instruction set computing (RISC) and Intel architectures, the latter having been maintained in parallel with the Motorola development. In fact, given the concurrent PowerPC and Intel builds maintained by Apple, and the hiring of a number of FreeBSD core developers from Wind River Systems, it has been speculated that Apple's 2005 PowerPC-to-Intel transition was long planned. In any event, it was a transition made much easier by the BSD foundation of NeXTStep and Darwin.

Open Source Base

The open source basis of Mac OS X was actually misrepresented in Steve Jobs's hyperbolic announcement of the operating system at the 2000 Macworld conference in San Francisco. Two pieces of that announcement in particular made more of those origins than was supported by reality:

- Calling Mac OS X "very Linux-like," Jobs noted that it uses "FreeBSD UNIX, which is the same as Linux." While there are similarities, there are also many differences between the two operating systems.
- Jobs also noted that Mac OS X was "completely open source." Again, that's not exactly true. While the Darwin code is, in fact, open source, many elements of Mac OS X are not.

While the FreeBSD basis of Mac OS X moved the Macintosh into the modern age, it is not the completely free and open source operating system painted by Jobs's Macworld announcement. But the timing was right, even for an open source–based operating system with many proprietary elements. By the time that introduction was made, hard-core computer users and developers had already begun a slow but steady migration to open source.

The concept of free and open source software (FOSS) was increasingly popular, growing from its origins in the Free Software Foundation of Richard Stallman in the 1980s.

In the early 1990s, Linux had become the most well known of an increasing number of FOSS applications. Interestingly, Linux creator Linus Torvalds later noted that it was because of the legal action surrounding BSD in those years that he wrote another operating system. He might have chosen BSD had it been widely available.

The philosophy of open source was further advanced in 1997 by Eric Raymond's publication of "The Cathedral and the Bazaar." In that work, Raymond quotes Torvalds as saying, "Given enough eyeballs, all bugs are shallow." This was one of the fundamental philosophical frameworks around which computer users rallied: the concept that software built, shared, and maintained by a large community of users was, by its nature, better. While not unique to open source, the concept found its strongest example in FOSS.

By 2000, open source software had begun to find its way into both server rooms and technical publications around the world. The continued use of open source tools in Mac OS X allowed Apple to stake a claim to some of that growing buzz. While some greeted the announcement with skepticism, many open source developers were surprised by what appeared to be a significant philosophical shift for the once secretive company.

The easy availability of the Darwin code from Apple's web site also ensured that curious developers would tinker with and improve on the product. The decision to continue using open source tools in Mac OS X created, to paraphrase Torvalds, "enough eyeballs to keep the bugs shallow." It allowed Apple to call upon a large pool of user-created tools. It helped to ensure that the in-house developers would always have the latest improvements in critical development tools. And it did so at little financial risk for Apple. Given the financial state of the company at the time of the NeXT acquisition, it's plausible to believe that open source both saved and renewed Apple.

Economics

As noted, the financial health of Apple at the time of the NeXT acquisition in 1997 was poor, and worsening. The first quarter of 1997 would bring 12-year stock price lows and mark the largest financial losses in Apple history, totaling some $700 million. Of predecessor Gil Amelio's business strategy, Jobs famously quipped, "Apple is like a ship with a hole in the bottom, and my job is to point the ship in the right direction."

Simply put, the financial health of the company allowed no room for immediate bold development initiatives. NeXTStep already existed, and it was in a state in which the only initial cost was that of porting it to the PowerPC platform.

Additionally, the Apple engineering staff had been nearly wiped out under the direction of a changing stream of CEOs. Much of the engineering staff after the NeXT acquisition was brought from NeXT itself. The engineers and developers were, in effect, already up to speed on the state of the operating system. The transition required few resources for retraining, research, and development. By basing Mac OS X on NeXTStep, They could hit the ground running, while realizing a significant cost savings.

Extensibility

Based on BSD, NeXTStep was highly extensible. The basis of NeXTStep was BSD's native C. The power of C had already been proven and could be extended easily as the operating system grew to meet new demands. With additional Objective-C frameworks, the extensibility of BSD and the NeXT tools brought efficiency in both process and economics.

How Is BSD Implemented in Mac OS X?

The full set of UNIX user-space tools available in FreeBSD is available in Mac OS X natively. While there are some core differences between UNIX and Linux, there are enough similarities between BSD and Linux that many common tasks in Mac OS X will seem familiar to Linux users.

If your preference is to use the command line as often as possible, for example, you can do so in Mac OS X. With the bash shell in place, it's possible to port administrative and other scripts from a Linux system to Mac OS X. As the minimal UNIX filesystem structure is intact in Mac OS X, moving and implementing these scripts can be nearly seamless, though some modification may be required to adjust the path of some system binaries called by the script.

As already noted, the core of Mac OS X is based on FreeBSD and is a true UNIX. Mac OS X 10.5, in fact, has been certified by the Open Group as UNIX 03–compliant.

In the Mac OS X implementation, FreeBSD is paired with the XNU kernel—a hybrid Mach microkernel containing additional Apple modifications. The kernel fully supports the following:

- Preemptive multitasking
- Protected memory
- Interrupt management
- Kernel debugging support
- Console input/output (I/O)
- Real-time support

Mac OS X also implements a suite of core services, which include system services such as the following:

- CFNetwork, a user-level networking API
- OSServices, a collection of system APIs
- WebServicesCore, APIs for using web services with SOAP and XML-RPC
- SearchKit, a framework for multilingual searches and indexing

Additionally, Mac OS X implements application services and application environments. The application services include graphics and windowing services, printing services, launch services, event services, and so on. The application environments include execution environments, such as those for BSD, Cocoa (the object-oriented application development API), and Java.

Why Switch from Linux to Mac?

Finally, we get to a question central to the rest of this book: why bother to switch from a Linux-based system to Mac OS X? There are three main reasons: hardware control, common code, and release stability.

Hardware Control

Apple, unlike its primary competitors, is in the business of both software and hardware. While the hardware for Apple's machines is chosen in part for profitability, it's also chosen for reliability and stability.

Hardware components are selected only after rigorous testing using the latest iteration of Mac OS X. While that sounds almost intuitive, the result of controlling the hardware running Mac OS X is that Apple's developers have a known hardware platform for which to develop. By itself, that practice eliminates a large number of the issues seen in other systems, including incompatibility between off-the-shelf components and operating system code. It allows Apple's developers to create device drivers that can be thoroughly tested prior to the release of a new Mac OS X version or a new hardware platform. It allows them to compile in hardware-specific optimization features within the operating system itself.

It's hard to overstate the importance and value of this controlled hardware environment. We've all experienced the frustration of incompatible drivers and other code in competing operating systems. Aside from the frustration, the practice of allowing a multitude of hardware components to utilize the operating system code means that we can never be sure that the code created by hardware vendors will interact optimally with the code created for the operating system. In the open source community, much of this code is, in fact, reverse-engineered. By first controlling the hardware, then developing to that hardware, Apple ensures that its operating system and hardware will work together seamlessly.

Common Code

The APIs, development frameworks, and integrated development environments (IDEs) for Mac OS X are included on the installation disc. Mac owners literally have access to most of the same tools used by the Apple developers to create Mac OS X. That means that applications created specifically for the Mac will have a consistent feel and operation.

A broad range of purely open source development tools is also available to install on the Mac with minimal effort. The Mac development tools can be used in tandem with these open source development tools. From within the same environment, developers can create cross-platform open source applications or applications specifically for the Mac.

Release Stability

Because Apple's operating system and application developers are working with known hardware, using common code for development, Apple's operating system releases are extremely stable. That commonality allows Apple's quality assurance testers to create test cases that will encompass the full range of possible issues in the code.

Summary

UNIX, BSD, and Mac OS X have clearly followed a long path to the convergence represented by the current Macintosh operating system. They're marked by many similarities and many significant differences. UNIX and BSD users will find the underlying Mac OS X environment very familiar, while Linux users will find enough commonality with their own operating system to provide a lot of comfort. Add to that the hardware control, common code, release stability, and beautiful look and feel of Mac OS X, and a move from Linux to Mac becomes much more attractive.

In the next chapter, we'll take a deeper look at the differences and similarities between Mac OS X and Linux to lay a solid foundation for your personal migration.