A History of UNIX (From the Computer World Article)

In August 1969, Ken Thompson, a programmer at AT&T subsidiary Bell Laboratories, saw the month-long departure of his wife and young son as an opportunity to put his ideas for a new operating system into practice. He wrote the first version of Unix in assembly language for a wimpy Digital Equipment Corp. (DEC) PDP-7 minicomputer, spending one week each on the operating system, a shell, an editor and an assembler.

Thompson and a colleague, Dennis Ritchie, had been feeling adrift since Bell Labs had withdrawn earlier in the year from a troubled project to develop a time-sharing system called Multics (Multiplexed Information and Computing Service). They had no desire to stick with any of the batch operating systems that predominated at the time, nor did they want to reinvent Multics, which they saw as grotesque and unwieldy.

After batting around some ideas for a new system, Thompson wrote the first version of Unix, which the pair would continue to develop over the next several years with the help of colleagues Doug McIlroy, Joe Ossanna and Rudd Canaday. Some of the principles of Multics were carried over into their new operating system, but the beauty of Unix then (if not now) lay in its less-is-more philosophy.

"A powerful operating system for interactive use need not be expensive either in equipment or in human effort," Ritchie and Thompson would write five years later in the Communications of the ACM (CACM), the journal of the Association for Computing Machinery. "[We hope that] users of Unix will find that the most important characteristics of the system are its simplicity, elegance, and ease of use."

Apparently they did. Unix would go on to become a cornerstone of IT, widely deployed to run servers and workstations in universities, government facilities and corporations. And its influence spread even farther than its actual deployments, as the ACM noted in 1983 when it gave Thompson and Ritchie its top prize, the A.M. Turing Award for contributions to IT: "The model of the Unix system has led a generation of software designers to new ways of thinking about programming."

Early steps: Ken Thompson and Dennis Ritchie in the early days of Unix.

Of course, Unix' success didn't happen all at once. In 1971 it was ported to the PDP-11 minicomputer, a more powerful platform than the PDP-7 for which it was originally written. Text-formatting and text-editing programs were added, and it was rolled out to a few typists in the Bell Labs Patent department, its first users outside the development team.

In 1972, Ritchie wrote the high-level C programming language (based on Thompson's earlier B language); subsequently, Thompson rewrote Unix in C, which greatly increased the OS' portability across computing environments. Along the way it picked up the name Unics (Uniplexed Information and Computing Service), a play on Multics; the spelling soon morphed into Unix.

It was time to spread the word. Ritchie and Thompson's July 1974 CACM article, "The UNIX Time-Sharing System," took the IT world by storm. Until then, Unix had been confined to a handful of users at Bell Labs. But now with the Association for Computing Machinery behind it -- an editor called it "elegant" -- Unix was at a tipping point.

"The CACM article had a dramatic impact," IT historian Peter Salus wrote in his book The Daemon, the Gnu and the Penguin. "Soon, Ken was awash in requests for Unix."
Hackers’ heaven

Thompson and Ritchie were the consummate "hackers," when that word referred to someone who combined uncommon creativity, brute force intelligence and midnight oil to solve software problems that others barely knew existed.

Their approach, and the code they wrote, greatly appealed to programmers at universities, and later at startup companies without the mega-budgets of an IBM, Hewlett-Packard or Microsoft. Unix was all that other hackers, such as Bill Joy at the University of California, Rick Rashid at Carnegie Mellon University and David Korn later at Bell Labs, could wish for.

"Nearly from the start, the system was able to, and did, maintain itself," wrote Thompson and Ritchie in the CACM article. "Since all source programs were always available and easily modified online, we were willing to revise and rewrite the system and its software when new ideas were invented, discovered, or suggested by others."

Korn, an AT&T Fellow today, worked as a programmer at Bell Labs in the 1970s. "One of the hallmarks of Unix was that tools could be written, and better tools could replace them," he recalls. "It wasn't some monolith where you had to buy into everything; you could actually develop better versions." He developed the influential Korn shell, essentially a programming language to direct Unix operations, now available as open-source software.

Author and technology historian Salus recalls his work with the programming language APL on an IBM System/360 mainframe as a professor at the University of Toronto in the 1970s. It was not going well. But the day after Christmas in 1978, a friend at Columbia University gave him a demonstration of Unix running on a minicomputer. "I said, 'Oh my God,' and I was an absolute convert," says Salus.

He says the key advantage of Unix for him was its "pipe" feature, introduced in 1973, which made it easy to pass the output of one program to another. The pipeline concept, invented by Bell Labs' McIlroy, was subsequently copied by many operating systems, including all the Unix variants, Linux, DOS and Windows.

Unix was developed at Bell Labs' headquarters in Murray Hill, NJ. Credit: Alcatel-Lucent/Bell Labs

Another advantage of Unix -- the second "wow," as Salus puts it -- was that it didn't require a million-dollar mainframe to run on. It was written for the tiny and primitive DEC PDP-7 minicomputer because that's all Thompson and Ritchie could get their hands on in 1969. "The PDP-7 was almost incapable of anything," Salus recalls. "I was hooked."

A lot of others got hooked as well. University researchers adopted Unix in droves because it was relatively simple and easily modified, it was undemanding in its resource requirements, and the source code was essentially free. Startups like Sun Microsystems and a host of now-defunct companies that specialized in scientific computing, such as Multiflow Computer, made it their operating system of choice for the same reasons.
Unix offspring

Unix grew up as a non-proprietary system because in 1956 AT&T had been enjoined by a federal consent decree from straying from its mission to provide telephone service. It was okay to develop software, and even to license it for a "reasonable" fee, but the company was barred from getting into the computer business.

Unix, which was developed with no encouragement from management, was first viewed at AT&T as something between a curiosity and a legal headache.

Then, in the late 1970s, AT&T realized it had something of commercial importance on its hands. Its lawyers began adopting a more favorable interpretation of the 1956 consent decree as they looked for ways to protect Unix as a trade secret. Beginning in 1979, with the release of Version 7, Unix licenses prohibited universities from using the Unix source code for study in their courses.

No problem, said computer science professor Andrew Tanenbaum, who had been using Unix V6 at Vrije Universiteit in Amsterdam. In 1987 he wrote a Unix clone for use in his classrooms, creating the open-source Minix operating system to run on the Intel 80286 microprocessor.

"Minix incorporated all the ideas of Unix, and it was a brilliant job," Salus says. "Only a major programmer, someone who deeply understood the internals of an operating system, could do that." Minix would become the starting point for Linus Torvalds' 1991 creation of Linux -- if not exactly a Unix clone, certainly a Unix look-alike.

Stepping back a decade or so, Bill Joy, who was a graduate student and programmer at the University of California at Berkeley in the '70s, got his hands on a copy of Unix from Bell Labs, and he saw it as a good platform for his own work on a Pascal compiler and text editor.

Modifications and extensions that he and others at Berkeley made resulted in the second major branch of Unix, called Berkeley Software Distribution (BSD) Unix. In March 1978, Joy sent out copies of 1BSD, priced at $50.

So by 1980 there were two major lines of Unix, one from Berkeley and one from AT&T, and the stage was set for what would become known as the Unix Wars. The good news was that software developers anywhere could get the Unix source code and tailor it to their needs and whims. The bad news was they did just that. Unix proliferated, and the variants diverged.

In 1982 Joy co-founded Sun Microsystems and offered a workstation, the Sun-1, running a version of BSD called SunOS. (Solaris would come about a decade later.) The following year, AT&T released the first version of Unix System V, an enormously influential operating system that would become the basis for IBM's AIX and Hewlett-Packard's HP-UX.
The Unix Wars

In the mid-'80s, users, including the federal government, complained that while Unix was in theory a single, portable operating system, in fact it was anything but. Vendors paid lip service to the complaint but worked night and day to lock in customers with custom Unix features and APIs.

In 1987, Unix System Laboratories, a part of Bell Labs at the time, began working with Sun on a system that would unify the two major Unix branches. The product of their collaboration, called Unix System V Release 4.0, was released two years later and combined features from System V Release 3, BSD, SunOS and Microsoft's Xenix.

Other Unix vendors feared the AT&T/Sun alliance. The various parties formed competing "standards" bodies with names like X/Open, Open Software Foundation, Unix International and Corporation for Open Systems. The arguments, counter-arguments and accomplishments of these groups would fill a book, but they all claimed the high road to a unified Unix while taking potshots at each other.

In an unpublished paper written in 1988 for the Defense Advanced Research Projects Agency (DARPA), the noted minicomputer pioneer Gordon Bell said this of the just-formed Open Software Foundation, which included IBM, HP, DEC and others allied against the AT&T/Sun partnership: "OSF is a way for the Unix have-nots to get into the evolving market, while maintaining their high-margin code museums."

The Unix Wars failed to settle differences or set a true standard for the operating system. But in 1993, the Unix community received a wakeup call from Microsoft in the form of Windows NT, an enterprise-class, 32-bit multiprocessing operating system. The proprietary NT was aimed squarely at Unix and was intended to extend Microsoft's desktop hegemony to the data center and other places owned by the likes of Sun servers.

Microsoft users applauded. Unix vendors panicked. All the major Unix rivals united in an initiative called the Common Open Software Environment, and the following year more or less laid down their arms by merging the AT&T/Sun-backed Unix International group with the Open Software Foundation. That coalition evolved into today's The Open Group, certifier of Unix systems and owner of the Single Unix Specification, now the official definition of "Unix."

As a practical matter, these developments may have "standardized" Unix about as much as possible, given the competitive habits of vendors. But they may have come too late to stem a flood tide called Linux, the open-source operating system that grew out of Prof. Tanenbaum's Minix.
Just what is "Unix," anyway?

Unix, most people would say, is an operating system written decades ago at AT&T's Bell Labs, and its descendents. The major versions of Unix today branched off a tree with two trunks: one emanating directly from AT&T and one from AT&T via the University of California at Berkeley. The stoutest branches today are AIX from IBM, HP-UX from HP and Solaris from Sun.

However, The Open Group, which owns the Unix trademark, defines Unix as any operating system that has been certified by it to conform to the Single Unix Specification (SUS). This includes OSes that are usually not thought of as Unix, such as Mac OS X Leopard (which descended from BSD Unix) and IBM's z/OS (which descended from the mainframe operating system MVS), because they conform to the SUS and support SUS APIs. The basic idea is that it is Unix if it acts like Unix, regardless of the underlying code.

A still broader definition of Unix would include Unix-like operating systems -- sometimes called Unix "clones" or "look-alikes" -- that copied many ideas from Unix but didn't directly incorporate code from Unix. The leading one of these is Linux.

Finally, although it's reasonable to call Unix an "operating system," as a practical matter it is more. In addition to an OS kernel, Unix implementations typically include utilities such as command line editors, application program interfaces, development environments, libraries and documentation.