The Legal History of UNIX® and Free Software

Gregory M. Pomerantz

19 June 2004

The Legal History of UNIX® and Free Software

Gregory M. Pomerantz

> Those who do not understand Unix are condemned to reinvent it, poorly.
> — Henry Spencer

1. Introduction

The policy arguments surrounding the application of intellectual property to computer software remain complex and poorly understood. Many open questions remain: does strong centralized control over development promote innovation or inhibit it? Are patent rewards necessary to reward investment in R&D?

While there is much debate on the influence of patent and copyright law on innovation in software design, very little discussion is taking place on the proper role of trade secrecy.

Discussions about the proper scope of IP rights over software often revert to unprovable arguments about natural rights[1] or imaginary future innovations or products.[2] What is often lacking from these discussions is a close analysis of history.[3] Only through such an analysis can the relative value of various incentive structures be determined.

In a highly networked computing environment, interoperable communications protocols are fundamental to allowing competitive marketplaces to exist for software and service providers. Just as fundamentally, access to low cost, widely available software development infrastructure is equally necessary for the process of innovation, or else the owner of that infrastructure will be able to extract rents from all innovation and thereby impede its progress.

my project in a nutshell --

Using the history of the Unix operating system as an example, this paper will argue that, for certain classes of software, the copyright law with published source code may be the most appropriate form of protection for certain classes of software — in particular, software such as Unix that provides the necessary infrastructure for future innovation. I will argue that copyright protection is incompatible with trade secrecy from a legal standpoint. Furthermore, application of trade secrecy to widely disseminated infrastructure software may have severe negative consequences on innovation and may inhibit rather than encourage improvement of the protected software.

Unix is an operating system and integrated programming environment developed by AT&T Bell Labs. It was created in 1969 and has been continually enhanced ever since. It was the first operating system written in a high level programming language. As a result, it was possible to port it to a wide variety of machines. Unix was consequently the first operating system to be available on a wide variety of computer systems from competing hardware vendors. Today, Unix or Unix like operating systems are in use on everything from wristwatches to supercomputers.

---

[1] ??

[2] Jane Ginsberg's celestial jukebox.

[3] FIXME: mention oft' ignored fact that Apple, Sun and Oracle began by selling free software.

Beginning in 1973, AT&T distributed the Unix source code to universities on a non-commercial basis under restrictive trade-secret agreements. From that point forward, AT&T moved more and more towards a more traditional commercial distribution mode. Nevertheless, there have always been multiple vendors for Unix, licensing the core system from AT&T, and then relicensing their enhancements to their own customers. In addition, educational licenses granted to Unix permitted an entire generation of computer scientists to study its principles. Throughout all this, AT&T walked a thin line between copyright and trade secret protection. In practice, their right behaved like a classical copyright — the expressive textual element Unix (the source code) was widely disseminated and studied, while AT&T collected revenues from this dissemination. This led to rapid improvements from both academic and commercial licensees. This in turn led to widespread adoption of and dependence on the system.

> The propagation of Unix has always survived on a tightrope between the interests of a changing commercial world and the research and academic communities. Remarkably liberal policies for distribution of the system's source code encouraged the seminal developments during the 1970's and 1980's; in particular, the widely available work on Unix at the University of California at Berkeley was significant in building both today's Internet and the workstation industry. At the same time, the corporate guardians worried continually that they might be giving away all rights.[4]

Software development is a discursive process. Open sharing of fundamental software breeds collaborative use, and that breeds user-driven innovation. I will argue that copyright law must demand openness from software vendors that seek to dominate the tools of interoperability and innovation. This is fundamentally incompatable with trade secrecy. Unix is a compelling case study because, while its creators attempted to assert their ownership through both copyright and trade secrecy law, in actual practice its wide publication in comprehensible form (i.e. source code) resulted in just the sort of discursive production of follow-on innovation that one might expect from other cultural products such as music or fiction — cultural products which are commonly subject only to copyright laws. The issues regarding source-code dissemination versus secrecy are of crucial importance to the current crop of companies such as Microsoft and Sun who are attempting to benefit from "open source" concepts while simultaneously retaining both trade secret and copyright control.

2. Early Computing

2.1. Rented Hardware and Metered Use

The modern business of computing traces its origin back to the US Census of 1880.[5] A 21 year old engineer named Herman Hollerith worked briefly at the Census Bureau on methods of analyzing the large amounts of data that needed to be collected and tabulated by hand. That census was nearly a disaster, taking 7 years to complete and costing an enourmous amount of money. Hollerith experienced first hand the need for calculating machines that could perform the mechanical portions of the work. Joining the faculty of MIT in 1882, he quickly set about building such a machine, applying for his first patent in 1884. Part of Hollerith's inspiration came from the Jacquard loom, developed in 1804, which was capable of automatically weaving patterns by sensing holes punched in cards.[6] His tabulating machine used similar cards, employing pins which made electrical contact with a pool of mercury below the card, thereby actuating an electric counter.

Hollerith's tabulating machine was selected over two competing systems for use in the 1890 US census. It succeeded in completing its calculations in three months — significantly faster than

[4] Dennis Ritchie, forward to Lions, p x.

[5] See http://www-history.mcs.st-and.ac.uk/~history/Mathematicians/Hollerith.html; http://accounting.rutgers.edu/raw/gsm/1indus.htm; Salus, 25 Years of Unix

[6] In those days, card decks were sometimes stolen by competing textile mills, an early (perhaps the earliest) example of software piracy. http://www.columbia.edu/~fdc/jacquard.html

the two years estimated for a hand count, while at the same time saving the government an estimated $5 million. Hollerith's machines were used again for the 1900 census, but he was by then more fully exploiting his monopoly, having raised his prices substantially in the intervening years. Very soon, the Census Bureau began to look elsewhere, enlisting James Powers to design around Hollerith's patents and come up with a competing machine.[7] Although Powers eventually won the contract for the 1910 census, Hollerith had by that time diversified his company, offering his machines for accounting, sales forecasting, and a variety of other commercial purposes.

Thomas J. Watson, fired from National Cash Register in 1914, soon joined Hollerith's company as CEO. He changed the company's name to International Business Machines in 1924. James Powers' tabulating machine company dominated the market for some time, but eventually lost in the marketplace due to IBM's marketing ability and it's substantial investments in research and development. It was acquired by Remington-Rand in 1927.

2.2. Hollerith's Business Model

One of Hollerith's early innovations was his business model, already in place by the 1890 census — rent the machines, but sell the patented cards.[8] IBM, Rand, and others would carry this business model from the world of tabulating machines to the world of programmable electronic computers, which became commercially available in 1951. In addition, they would supplement their patent control over the cards with exclusive dealing arrangements.

Punched cards were, for decades, the preferred medium for inputting data and software into general purpose computers.[9] Of course, they still looked very similar to those used in the Jacquard loom 150 years before. After all, they're just pieces of paper with holes in them, albeit specially shaped pieces of paper with special holes. IBM's tabulating card patents and exclusive supply agreements made them the sole supplier of cards for use with their tabulating machines and computers. Likewise, Remington Rand was the sole supplier of cards for Remington Rand hardware.

Since by their nature, punched cards are not reusable, the number of cards consumed by a user was directly proportional to the amount of work the machine did. By charging supra-competitive prices for those cards, IBM could "meter" the use of its machines, deriving greater revenues from heavier users.[10] Presumably, the customers who used their machines the least were the ones who valued them the least. Therefore, such a strategy of price discrimination should in principle have allowed IBM to charge those customers less than what otherwise would have been the profit maximizing price. A simplified economic model shows that this leads to economically efficient results.[11] The United States Supreme Court has not generally been convinced by these arguments. In 1936, it found the tying arrangements employed by IBM and Rand llegal under the antitrust laws,[12] and recent typing cases have not overturned that precedent.[13]

2.3. Metering is Bad

On January 25, 1956, the United States settled another major antitrust suit against IBM. Much of the decree was intended to block IBM from using its dominant market position to

[7] Among other differences, Powers used cards with round rather than square holes. Dickinson v. Philadelphia, 73 Pa. D. & C. 523 (Common Pleas Court of Philadelphia County, 1951).

[8] Presumably IBM was able to preserve its patent rights by "improving" the cards about every 20 years. By 1936, IBM was selling 3 billion cards per year, accounting for nearly 25% of its tabulating machine revenue. International Business Machines Corp. v. United States, 298 U.S. 131 (1936) at 136.

[9] See, e.g. Gordon B. Davis, Introduction to Electronic Computers at 220 (1971) ("The most common input medium for introducing source data into computer systems is the Hollerith 80-column punched card. The second most common card input type is the 96-column card for IBM System/3.")

[10] See generally Roger D. Blair, The Individual Coercion Doctrine and Tying Arrangements: An Economic Analysis, 10 Fla. St. U.L. Rev. 531 (1983).

[11] See, e.g. Eleanor M. Fox and Lawrence A. Sullivan, Cases and Materials on Antitrust, p 662.

[12] IBM v. United States, 298 U.S. 131 (affirming an order enjoining IBM from including exclusive supply terms in its leasing agreements.)

[13] See e.g. Jefferson Parish, 466 U.S. 2 (1984). But see O'Connor's dissent (footnote 4, recognizing economic literature on the potential beneficial effects of price discrimination).

monopolize the service bureau industry.[14] However, a close analysis of the ruling reveals that it was in fact designed to prevent IBM from subjecting its customers to metered pricing. It had provisions designed to limit IBM's ability to enter long term leases,[15] prevent IBM from encouraging customers to lease rather than buy machines,[16] and prevent dominance over the repair and maintenance business.[17] IBM's standard rental agreement only permitted the customer to use the machine for a specified amount of time, with additional time costing up to 40% more. Likewise, the cost of a maintenance agreement also varied depending on use,[18] and fees for service bureau work were keyed solely to the amount of use. Additionally, the settlement imposed a variety of terms aimed at forcing IBM to permit competition in the market for tabulating cards,[19] which they apparently continued to dominate despite the 1936 judgment.[20] Finally, the court required that IBM organize its service bureau business into a subsidiary which would have no unfair advantage over service bureaus not owned by IBM.[21] Thus, the bulk of the settlement seems to be aimed at the ability of IBM to meter.

Despite later economic literature to the contrary, I believe the government's intuitions in the case were correct. While the static effect of IBM's practices may have been efficient, the dynamic effect of metering is to deter user driven inovation and therefore is inefficient over the long term.[22]

2.4. Hulking Giant

Throughout the 50's and 60's, the business of computing seems to have been modeled around the rigid limitations in the earlier tabulating machines themselves.[23] IBM computers at the time cost millions of dollars and could not operate without the support of an entire bureaucracy. Programming such a machine required that you first give your code and data to a keypunch operator who would, later in the day, hand you back a stack of punched cards. Another team of technicians would be responsible for actually loading your program into the computer. Programs were then executed in batches to guarantee that the CPU was never idle. If there were any problems with your code, you could fix it and try again the next day. While early computers had an effective response time measured in hours or days, many studies of interactive systems have shown that decreases in response time below one second can result in marked improvements in productivity over slower interactive systems.[24] Clearly, while batch oriented systems may be very efficient for well understood tasks, they are not the optimal environment in which to use a computer for creative purposes. Perhaps few people thought that end users were going to come up with any useful software anyway, though subsequent history has certainly proven that they do. Between the metering effect of the punched cards, the usage-based leasing fees, and the massive necessary staff of technicians and engineers, IBM was guaranteeing that no customer would ever use their

[14] The Service bureau business was defined as "the preparation with tabulating and/or electronic data processing machines of accounting, statistical and mathematical information and reports for others on a fee basis." United States v. IBM, 1956 U.S. Dist. LEXIS 3992 4 (S.D.N.Y. 1956).

[15] Id. at 12-13.

[16] Id. at 6-12.

[17] Id. at 17-18.

[18] Sullivan, Monopolization: Corporate Strategy, the IBM Cases, and the Transformation of the Law, 68 Texas L.Rev. 587, 599-604 (1982).

[19] Id. at 18-27.

[20] IBM v. United States, 298 U.S. 131.

[21] United States v. IBM, 1956 U.S. Dist. LEXIS 3992 at 13-17.

[22] **FIXME needs to be fleshed out. Should follow this up in a later section.

[23] See generally Stephen Levy, Hackers, p 19.

[24] See "A comparative Study of System Response Time on Program Developer Productivity" by G. N. Lambert, IBM Systems Journal, Volume 23, No 1, 1984, page 36. ("Based on the results of this study and on the project statistics for the MAT Stage 2 development, the facilities of subsecond response time should be made available to as many developers as feasible. Higher productivity encourages developers to do more work at the terminal. Thus each developer should have a terminal as part of his work station, in the same way that a desk and a telephone are provided." See also ?? Miller, R. B. (1968). Response time in man-computer conversational transactions. Proc. AFIPS Fall Joint Computer Conference Vol. 33, 267-277.

computer for anything other than the safest, short term highest value use. Thus a practice that was both economically efficient and eminently sensible from a business perspective had the effect of severely inhibiting any meaningful possibility of user driven innovation.[25] The innovation inhibiting effect is due to the way in which the use of the computer is structured by constraints that increase risk of experimentation. I will argue that proprietary constraints on widely deployed infrastructure software can have a similar effect. IBM's practices may have at least been justified by the physical constraints inherent in the earliest electronic computers. Today we have no such excuse.

2.5. MIT and DEC

The Massachusetts Institute of Technology was one of the early users of IBM computers.[26] However, to the computer hackers in MIT's Tech Model Railroad Club (TMRC) in the 50's and 60's, the stultifying effect of using the Hulking Giant, as they called it, was intuitively obvious.[27] A $3 million TX-0 computer, "loaned" to MIT from Lincoln Labs, offered an alternative. The machine, one of the first fully transistorized computers, was designed by Ken Olson and Harlan Anderson as part of the SAGE project,[28] an ambitious Defense Department project run by IBM and Jay Forrester. SAGE demanded such a massive system that smaller computers were required just to test it. The TX-0 was one of those test computers. Nevertheless, it was quite capable for the time, and offered a fertile ground for the community at MIT to explore the possibilities of computing. Most significantly, the TX-0 was interactive.

The hacker culture at MIT demanded above all hands on use of technology. Unlike MIT's IBM computer, the TX-0 was set up to allow the programmer to sit down directly in front of the console and feed their program (on paper tape, as the TX-0 did not use cards) directly into the machine. The results would appear in real time, not the next day, giving the programmer the ability to immediately fix problems or make improvements. Whenever someone wrote a program they were particularly proud of, they put a copy in the drawer next to the console of the TX-0 so that the next person who came along could build on all of the work that had been done before. Programs that were repeatedly improved in this way achieved a level of quality that would have been impossible for any one programmer working alone.

John McKenzie, the technician at MIT who was in charge of the TX-0, recognized that "the interactive nature of the TX-0 was inspiring a new form of computer programming, and the hackers were its pioneers."[29] The TX-0 had a very small amount of memory and no disk drive, and came from Lincoln Labs with virtually no system software. MIT professor Jack Dennis introduced the TMRC members to the TX-0. A former member of the TMRC himself, Dennis and others quickly set about writing system software to aid in programming the new machine. Successful programs were of course kept in the drawer within easy reach of any of the computer's users. The drawer next to the TX-0 may have been the closest thing this computer had to an operating system, providing a standard base, a common starting point for new development, and a place to put tools that abstract and simplify tasks in a generally agreed upon way.[30]

Ken Olson and Harlan Anderson, the original designers of TX-0, started their own business in 1957, intending to "beat [IBM] at their own game."[31] At the time, a recent study had predicted

[25] IBM is used in this discussion simply as an example. Virtually every computer manufacturer before the "minicomputer" revolution used these tactics.

[26] My focus on MIT is not meant to imply that very significant developments in computing did not occur elsewhere. Rather, it is meant to offer a view of the user driven style of innovation, in contrast to the software development style practiced at IBM and other computer manufacturers.

[27] See generally Levy, Hackers (1984).

[28] "Semi-Automatic Ground Environment."

[29] Levy, page 12.

[30] Such a drawer or cabinet was a common feature of computing facilities, usually called a library. Some large computing facilities employed "librarians" to catalog and organize these programs. The word "library" is still used to refer to common subroutines supplied with an operating system, though of course human librarians are a thing of the past.

[31] Salus, p. 18.

that the entire market for computers would be around 100. Thus the newly formed Digital Equipment Corporation (DEC) would instead call their computers Programmable Data Processors (PDP's). Their first completed PDP was shipped to defense contractor Bolt, Baranek and Newman in 1961. The second was donated to MIT where it quickly displaced the TX-0 as the preferred tool of the burgeoning hacker community. BBN wrote some of the early system software for the PDP. A great deal of other software was written at MIT by the former TX-0 hackers. Some of these programs were later distributed by DEC. Compensation for this work was not on the minds of the MIT programmers. Steven Levy described the attitudes in the lab at the time.

> As for royalties, wasn't software more like a gift to the world, something that was reward in itself? The idea was to make a computer more usable, to make it more exciting to users, to make computers so interesting that people would be tempted to play with them, explore them, and eventually hack on them. When you wrote a fine program you were building a community, not churning out a product.[32]

This culture of sharing would later be common to many Unix developers, and would be significant to its history.

In 1965, DEC's PDP-8 was introduced. This computer has been called the Model-T of computing because it was mass produced and very inexpensive. This machine and other "minicomputers" would help usher in a new era in low cost computing. DEC was able to undercut IBM and others by building smaller, cheaper, and more accessible machines. Much later, the microcomputer would turn the tide, with cheap Apples and IBM PCs undercutting the minicomputer market. Ken Olson and Harlan Anderson's Digital Equipment Corporation has been carved up and sold to Intel and Compaq, two companies which owe their success to the microcomputer revolution which displaced the minicomputer in the same way the minicomputer displaced the mainframe.

3. AT&T

> The phone company is nothing but a computer.[33]
> — John Draper (a.k.a. Captain Crunch)

3.1. Antitrust

International Business Machines was not the only corporate giant to settle a major government antitrust suit in January of 1956. The day before the consent decree in United States v. IBM[34] was issued, a final judgment was handed down against Western Electric and American Telephone and Telegraph.[35] This decree would directly influence the nature of software development within the telephone company, and the unusual software licensing strategy employed by AT&T for the Unix system owes its origin to this final judgment.

The suit, filed in 1949, alleged that AT&T's practice of purchasing telecommunications equipment exclusively from Western Electric was illegally exclusionary, and sought to sever the close relationship between the regulated monopoly businesses of telephone service and the ostensibly competitive market for telecommunications equipment.[36] Seven years later, the Eisenhower administration settled this suit under terms that, at the time, seemed very favorable to the defendants. The 1956 consent decree stipulated that, while AT&T would remain the regulated telephone monopoly, they would be enjoined from conducting any business other than the provision of

[32] Levy, p 56.

[33] Quoted in Esquire magazine, 1971. 249.

[34] 1956 U.S. Dist. LEXIS 3994.

[35] United States v. Western Electric Company and American Telephone and Telegraph Company. 1956 U.S. Dist. LEXIS 4076.

[36] Krattenmaker, Telecommunications Law and Policy, page 355, quoting Roger G. Noll & Bruce M. Owen, The Anticompetitive Uses of Regulation: United States v. AT&T, in The Antitrust Revolution.

common carrier communications services[37] or the manufacture of telecommunications equipment.[38] Lawyers at both companies interpreted the decree to permit experiments involving computers, but to disallow commercial sale or lease of computers.[39] In addition to those restrictions, the defendants agreed to license their patents on "reasonable and non-exclusive" terms.[40] Perhaps the most important of these patents covered the transistor, for which three Bell Labs scientists would win that year's Nobel Prize. The agreement proved to have unintended consequences as AT&T's telephone switching facilities evolved from hardwired switches to software running on computers.

3.2. Digitization

By this time, the technology of telecommunications had come a long way since Alexander Graham Bell first telephoned Thomas Watson. Increasingly sophisticated systems were developed to improve the transmission of signals and to mechanize their interconnection. While the earliest telephone systems required operator intervention to connect even local calls,[41] by 1951, it was possible in some areas for a telephone customer, using only a pulse dialing phone, to connect long-distance calls within the United States without assistance.[42]

In those days, signals were transmitted from one telephone to another over analog lines. Long distance transmission was made possible by vacuum tube amplifiers which regenerated the signal every few miles. AT&T had purchased the rights to the vacuum tube based audion amplifier, enabling the demonstration of the first transcontinental telephone circuit at a 1915 Exposition. The audion amplifier was also an enabling technology for broadcast radio, and much later, vacuum tubes would power the first generation of general purpose electronic computers. The vacuum tube also made possible the use of multiplexing, whereby many telephone signals could be transmitted from one central office to another over the same physical wires.

Interconnection was accomplished with a variety of complex electromechanical systems, housed in a hierarchy of central offices. The pulses generated by a rotary telephone would directly actuate the switches within these central offices, enabling one telephone to be automatically connected to any other.[43] The "control" portion of the switch was mechanical and directly coupled to the user's telephone. The lines thus connected created an end-to-end analog transmission path for the duration of the call.

Commercial deployment of the transistor would, within five years of the consent decree, cause the two basic functions of transmission and interconnection to undergo fundamental changes. Although the transistor was first deployed commercially within the Bell System simply as a replacement for vacuum tube based products, it also made possible digital transmission and electronic switching. The first digital transmission facility was deployed in 1962,[44] and the first transistor based electronic switch began service in 1965. The Number 1 ESS, as the new switch was called, was designed "to be far more flexible than any preceding system, to anticipate the need for new services, and to avoid the high costs associated with modifying existing systems."[45]

---

[37] 1956 U.S. Dist. LEXIS 4076 at 6.

[38] 1956 U.S. Dist. LEXIS 4076 at 5.

[39] 1956 U.S. Dist. LEXIS 4076, page 7 ("The defendant AT&T is enjoined and restrained from engaging ... in any business other than the furnishing of common carrier communications services; provided, however, that this Section V shall not apply to ... (b) experiments for the purpose of testing or developing new common carrier communications services ..."). In 1956, there was no commercial market for computer software, but AT&T would later interpret the decree to preclude competition in that market as well.

[40] 1956 U.S. Dist. LEXIS 4076 at 9-23.

[41] The first telephone exchange was installed in 1878. The four telephone operators knew all 21 subscribers by name, thus telephone numbers were not used. http://www.voicendata.com/aug98/milestone.html

[42] http://www.bell-labs.com/history/75/timeline.html

[43] W. F. Brinkman and D. V. Lang, Physics and the communications industry www.bell-labs.com/history/physicscomm/Physics_Com_wFig.pdf

[44] Physics and the Communications Industry page 8.

[45] Engineering and Operations in the Bell System, page 245.

In fact, it had at its core a general purpose stored program computer. Thus technological developments from the vacuum tube to the transistor amplified the similarities between the two fields of computing and communications,[46] and the legal boundaries between them thrown up by the 1956 decree began to blur.

3.3. The Computer Utility

AT&T had a number of good reasons to advance the science of computing. First of all was the growing complexity of the telephone network itself. Increased automation had long been recognized as the primary solution to this problem. Given the long service life of telephone switching equipment, only a flexible stored program based switch like the No. 1 ESS could economically adapt to provide new services. If you came up with a new service, simply rewrite the software and the entire network could be upgraded with minimal expense. Later, computers would be employed to automate the administrative tasks of servicing and maintaining the vast array of facilities deployed in the Bell System.

In 1961, MIT professor John McCarthy predicted the emergence of a technology that could have greatly increased the demand for telecommunications services: the computer utility. He said that "computation may someday be organized as a public utility, just as the telephone is a public utility."[47] He suggested that customers could connect to computing service companies over telephone lines, with each subscriber paying only for the capacity that they use. Given the very high price of computers, it was reasonable to think that there weren't going to be very many of them around, and a computer utility accessed by a telecommunications link would be the ideal way to allow people to use a computer who were geographically distant from the nearest one. In addition, the high costs of purchasing and operating computers, from the large full-time technical staff to the air-conditioning units and huge electricity bills, could be consolidated and shared among many end users.[48] Demand for data communications services was rapidly increasing in 1964. This was due in part to the arrival of the highly successful IBM 360 computer, which allowed its users to submit programs from a remote terminal over a telecommunications link.[49]

3.4. Time-Sharing

The 360 ran a batch oriented operating system, which meant that programs would be put into a queue and executed one at a time. Therefore, there could be a long delay between the submission of the job and the availability of the result. Multiple users could not simultaneously talk directly to the machine, which put a damper on the use of the 360 for "computer utility" purposes. The technique of allowing multiple people to interactively use the same computer at the same time is called time-sharing. The computer rapidly moves from one task to another, giving the illusion that all tasks are running simultaneously. By connecting several terminals to a computer, many users could simultaneously engage in interactive sessions, each feeling as if they had the computer to themself.[50]

AT&T was not the only organization looking for a time-sharing system. At a time when the expense of a computer precluded each user from having their own, time-sharing was the only way to truly answer the hands on imperative of the hacker ethic. It promised to greatly expand access to interactive computing, providing a fundamental shift in the way programs were created. Fredrick Brooks from IBM would later write about the impact of time-sharing on programmer productivity: "most observers credit time-sharing with a major improvement in the productivity

[46] For a timeline of Bell Labs contributions to computer science, see http://www.bell-labs.com/history/unix/blcontributions.html.

[47] John McCarthy, quoted in Steve Bickerstaff, Shackles on the Giant: How the Federal Government Created Microsoft, Personal Computers, and the Internet, 78 Tex. L. Rev. 1 (1999), footnote 11.

[48] See generally Shackles on the Giant, 78 Tex. L. Rev. 1 (1999) pp 4-8.

[49] Milgo Electronics Corp. v. United Telecommunications, Inc. 1976 U.S. Dist. LEXIS 17204 at 34.

[50] The No. 1 ESS, built by Western Electric and deployed in 1965, utilized a time-sharing operating system, although not for the purpose of providing multiple interactive sessions. The single central control of the No. 1 ESS could simultaneously manage up to 65,000 telephone lines by rapidly shifting its attention from one to another. Engineering and Operations in the Bell System, page 247.

and in the quality of their product."[51]

The Department of Defense also saw the value of time-sharing and in 1963 promised MIT $3 million a year for project MAC (which stood for Machine Aided Cognition or Multiple Access Computers). MIT through project MAC would develop some of the earliest functioning time-sharing systems and would soon team up with AT&T to develop a new system called Multics.[52] GE was chosen to provide the hardware for Multics in the form of the GE-645 computer, and became the third collaborator on that project. AT&T's lawyers viewed this project as "experimental" work, thus excluded from the prohibition on engaging in businesses other than common carrier communications services. AT&T intended Multics to replace its own BESYS operating system as the basic computing environment for all telephone company services.

Multics was also intended to form the basis of a computer utility. As F. J. Corbató of MIT and V. A. Vyssotsky of AT&T Bell Labs stated in a paper presented to the 1956 Fall Joint Computer Conference

> One of the overall design goals is to create a computing system which is capable of meeting almost all of the present and near-future requirements of a large computer utility. Such systems must run continuously and reliably 7 days a week, 24 hours a day in a way similar to telephone or power systems, and must be capable of meeting wide service demands: from multiple man-machine interaction to the sequential processing of absentee-user jobs; from the use of the system with dedicated languages and subsystems to the programming of the system itself; and from centralized bulk card, tape, and printer facilities to remotely located terminals. Such information processing and communication systems are believed to be essential for the future growth of computer use in business, in industry, in government and in scientific laboratories as well as stimulating applications which would be otherwise undone.[53]

To Corbató and Vyssotsky, the economic value projected for the computer utility was not the primary impetus for time-sharing.

> Principally for economic reasons, batch processing of computer jobs has been developed and is currently practiced by most large computer installations, and the concomitant isolation of the user from elementary cause-and-effect relationships has been either reluctantly endured or rationalized.

The solution was time-sharing. The paper continues:

> The impetus for time-sharing first arose from professional programmers because of their constant frustration in debugging programs at batch processing installations.... However, at Project MAC it has turned out that simultaneous access to the machine, while obviously necessary to the objective, has not been the major ensuing benefit. Rather, it is the availability at one's fingertips of facilities for editing, compiling, debugging, and running in one continuous interactive session that has had the greatest effect on programming. Professional programmers are encouraged to be more imaginative in their work and to investigate new programming techniques and new problem approaches because of the much smaller penalty for failure.[54]

In other words, time-sharing encourages risk taking, and the result is quality and innovation. The paper goes on to describe the most significant effect of the Project MAC system: the fact that it inspires people to use computers to find new solutions to problems in many different fields. Innovative research could be conducted that would never have been attempted in a more restrictive

[51] Fredrick Brooks, No Silver Bullet, Information Processing 1986, pp 1069-76. Brooks was the project manager for IBM's OS/360 operating system, completed in 1964 and one of the most massive software projects of its time. Even by 1975, Brooks did not fully appreciate the significance of time-sharing on programmer productivity, saying that "there is not yet much evidence available on the true fruitfulness of such apparently powerful tools." The Mythical Man Month, p 136.

[52] MULTiplexed Information and Computing Service.

[53] F. J. Corbató, V. A. Vyssotsky, Introduction and Overview of the Multics System (1965).

[54] Id.

computing environment.[55] Ithiel de Sola Pool has commented that it was the development of time-sharing systems that ultimately made the regulatory distinction between computing and communications, fundamental to both the AT&T and IBM settlements, untenable.[56] Time sharing systems were the hosts for the first online communities[57] and were the sociological and technological parents of today's computer networks.[58]

In 1965, a prototype of the Multics based computer utility was expected to be operational by 1966.[59] However, this was not to be. By 1969, Multics could barely support 3 simultaneous users.[60] The system was years behind schedule, and, because of the different priorities of the three groups involved, AT&T could see no way to bring the project back on track. They withdrew in April 1969.[61]

[55] Corbató and Vyssotsky may have had in mind projects like Ithiel de Sola Pool's computer analysis of the 1960 and 1964 presidential elections, or Terminal Oriented Social Science (TOSS) by Ithiel de Sola Pool, J. C. R. Licklider and Douwe B. Yntema, implemented first on a Lincoln Labs TX-2 and ported to Multics in 1969. See http://www.multicians.org/history.html; Pool, Ithiel de Sola, R. Abelson, and S. Popkin, Candidates, Issues, and Strategies: A computer simulation of the 1960 and 1964 presidential elections, Cambridge, MA: MIT Press (1965). Licklider would later be profoundly influential in the formation of the ARPANET, the predecessor to the Internet.

[56] Technologies of Freedom, p. 45.

[57] FIXME: Is this true?

[58] See generally Michael Hauben and Ronda Hauben, Netizens, available at http://www.columbia.edu/~rh120.

[59] Vyssotsky and Corbató, Structure of the Multics Supervisor (1965) ("Mutics must be a useful product and it is to be in operational use in 1966").

[60] Salus, p 6.

[61] Salus, p 29. Although Multics did not become a commercial system until years later, The knowledge gained from the project proved to be profoundly influential on later systems. For historical material on Multics, see generally http://www.multicians.org.

> Perhaps the most important achievement of UNIX is to demonstrate that a powerful operating system for interactive use need not be expensive either in equipment or in human effort: UNIX can run on hardware costing as little as $40,000, and less than two man-years were spent on the main system software.
> — Dennis Ritchie and Ken Thompson[62]

> No one User wrote me. I'm worth a couple million of their man-years!
> — Master Control Program (Tron)[63]

4. Unix

5. Space Travel

The cancellation of the Multics project was a blow to many of the computer scientists at Bell Labs. The group of researchers most affected were those who had grown accustomed to the convenience and efficiency of interactive computing and were faced with the fact that no suitable commercial substitute existed. Those researchers also missed the social aspects alluded to in the last chapter.

> What we wanted to preserve was not just a good environment in which to do programming, but a system around which a fellowship could form. We knew from experience that the essence of communal computing, as supplied by remote-access, time-shared machines, is not just to type programs into a terminal instead of a keypunch, but to encourage close communication.[64]

Several proposals to purchase a new machine on which to develop an operating system were rejected. Stung by the Multics experience, AT&T management was not interested in spending $100,000 on a new computer for operating system development.

Even without a computer, however, some operating system research did get done at Bell Labs in the summer of 1969. While working on the Multics project, Bell Labs researcher Ken Thompson had written a detailed simulation of the solar system called "Space Travel," in which a player could fly around, landing on various planets and moons. At first, Space Travel ran on Multics and GECOS,[65] but the GE machines were so expensive that a single game might cost $75 in computer time. Furthermore, those big systems were actually not very good at providing the smooth displays and interactivity required for a video game. Ken Thompson and Dennis Ritchie soon rewrote the game to run on to an old and neglected DEC PDP-7, in the process (and out of necessity) building up a library of programming tools for that machine. Also that summer, the two, along with Rudd Canaday (a coworker) sketched out the ideas for a file system, which Thompson soon implemented for the PDP-7. The file system provided the basis on which a self sustaining system could be built.[66] Very soon the PDP-7, although much less capable than GE machines, was supporting a simple but self sufficient programming environment for two simultaneous users. As a pun on Multics, the system was named UNICS, for UNiplexed Information and Computer Service. The name was soon shortened to Unix.

---

[62] The UNIX Time-Sharing System, Communications of the ACM, Volume 17, Number 7 (July 1974)pp 365-375.

[63] Disney, 1983?. In addition to being a diabolical artificial intelligence mastermind, the Tron MCP was the paradigmatic bad operating system, criticized throughout the movie for disallowing communication between users and programs (i.e. interactivity).

[64] Dennis M. Ritchie, The Evolution of the Unix Time-sharing System, AT&T Bell Laboratories Technical Journal 63 No. 6 Part 2, October 1984 pp 1577-93.

[65] The operating system written by GE for Bell Labs' GE-635.

[66] Technically, this is referred to as "self-hosting," where the system is sophisticated enough to support its own further improvement without the support of any other computer or operating system.

5.1. Word Processing

In the summer of 1970, a brand new DEC PDP-11 was purchased on a proposal to create a word processing system for use within the Bell System. The PDP-11 was a small machine for the time, with prices starting at a little more than $10,000. Partly for that reason, it would prove to be extremely popular, and it would soon establish DEC as the leading minicomputer company.

Word processing was a very good choice for an application in 1970. There were typists all over the Bell System, and what they needed was an inexpensive system that could allow multiple users to interactively edit and print text documents. Their needs were in fact not that much different from the needs of a software development group, who worked exclusively with textual documentation and source code. Unix already did virtually everything that was needed. A simplified text editor (named ed) and text formatting system (named roff) were created, both heavily derived from or inspired by previous programs. The Patent Department of Bell Telephone Labs, recently expanded to meet the disclosure and licensing requirements of the 1956 consent decree, became the first customer of the new system. They would soon be using Unix to license *itself* to others outside of the Bell System.

While work on the word processing system and a variety of other projects progressed, the core operating system was continually enhanced. Fairly soon a design philosophy emerged based around a few core ideas. In contrast to the large scale design style typified by the Multics project, the Unix philosophy stressed simplicity. While the Multics System Programmer's Manual was around 3,000 pages,[67] the manual for the first edition of Unix, dated November 3, 1971, was a slim 194 pages.[68] Of course Unix was much less ambitious than Multics and had far fewer features. One of its strengths on the other hand was its comprehensibility. The core of the Unix philosophy is expressed below:

> Write programs that do one thing and do it well.
>
> Write programs to work together.
>
> Write programs that handle text streams, because that is a universal interface.[69]

Fundamental to the Unix philosophy is a mechanism, introduced in version ??,[70] called a "pipeline" (or "pipe"), which allows a user to easily set up a communications channel between two or more programs. By using pipes, a user could quickly build complicated and innovative systems using nothing more than the tools already supplied with the system. "The power of Unix originated here, from the relationships generated among programs, not from the programs themselves."[71] Fredrick Brooks wrote that "Unix and Interlisp, the first integrated programming environments to come into widespread use, are perceived to have improved productivity by integral factors."[72]

5.2. Sharing Unix

Because the computer science researchers at Bell Labs were the first Unix customers, the system was designed from the start to be good for software development.[73] However, as more and more projects within the Bell System began to depend on Unix, it was necessary to shield those projects from the ever-changing research system. To that end, the Unix Support Group was formed in September, 1973 under the leadership of Berkeley Tague. The aim of the new group was to support a reliable, stable platform for the development of internal Bell System

[67] See http://www.multicians.org/devdoc.html#MSPM.

[68] See http://cm.bell-labs.com/cm/cs/who/dmr/1stEdman.html.

[69] Salus, p 53.

[70] FIXME

[71] Salus, p. 53.

[72] Fredrick Brooks, No Sliver Bullet, Information Processing 1986.

[73] In particular, it was (and still is) very good for *operating system* development.

applications.[74] Unix proved to be well suited to a wide variety of tasks, eventually fulfilling the promise of Multics by becoming the standard computing platform for the entire Bell System.[75]

Meanwhile, outside of the telephone company, word was spreading about this new system that "captured most of the best features of Multics in a small, elegant package."[76] In 1974, Ken Thompson and Dennis Ritchie published a paper about Unix in the Communications of the Association for Computing Machinery.[77] This description was compelling enough to inspire a flood of requests for copies of the source code to the system, predominantly from academic institutions.

5.3. Early Unix Distribution

By 1973, AT&T was faced with the far reaching decision of whether and how to share the Unix system with those who were asking for it. AT&T's antitrust settlement and the state of copyright law combined to form the backdrop for this decision. Lawyers at Bell Laboratories interpreted two prongs of the settlement to bear on the question — the prohibition on engaging in any business other than the furnishing of common carrier communications services, and the requirement to license patents on reasonable, non-exclusive terms. Both requirements were interpreted conservatively by AT&T. In particular, in order to avoid any possible dispute with the Justice Department, AT&T interpreted the licensing requirement to apply to software as well as patents. On the other hand, the requirement to engage only in the communications business precluded any competition in the computer industry, including in the market for software (a market that was non-existent in 1956). Therefore AT&T would agree to license Unix but under terms that would make it clear that they were not pursuing the business of computer software. Otis Wilson, who would become the manager of AT&T's software sales, said "the policy was restated over and over again at every gathering of the faithful — 'As is, no support, payment in advance!' "[78] One result of the decree was that Unix was seen as something AT&T "couldn't make money from."[79] Sending the source code to universities therefore seemed irrelevant from a financial perspective.

Dispite this, the way Unix was licensed during the 1970's shows that AT&T was in fact careful about maintaining property rights over the system. AT&T became more and more resistant to the prohibition on competition in the computer industry, and when a second federal antitrust suit was filed in 1974, AT&T ended the suit by consenting to the 1984 divestiture. AT&T's agreement to divest was largely motivated by the elimination of the restriction on competing in the rapidly growing computer industry. While Unix grew in popularity and quality as an operating system, AT&T was simultaneously straining for the right to exploit it as a commercial product.

Unfortunately, intellectual property law as applied to computer software in the mid 1970's was highly uncertain, and the question of just how to secure strong property rights was difficult one. Even today, there is no clear resolution to the fundamental problem they were seeking to solve — how to walk the fine line between the sharing of their software and the maintenance of the highest level of intellectual property rights. AT&T chose to protect their software with trade secrets, and in fact appears to have made a conscious choice to obtain trade secret rights even if it meant putting their ability to enforce copyright protection at risk. However, in practice, their protection operated much like a copyright. They licensed copies on a per-machine basis with fees based on the intended use (e.g. academic, administrative, or commercial) and whether or not source code was provided. Furthermore, in the 1980's when they began to grant sublicensing rights, they charged the licensee for each copy they relicensed. Over time, vast numbers of computer scientists and students used and studied Unix in great detail because of its widely available

[74] Automating Telephone Support Operations: An Interview with Berkeley Tague, The Amateur Computerist, Volume 6 No. 1 (1994).

[75] Cite?

[76] Berkeley Tague, Id.

[77] The UNIX Time-Sharing System, Communications of the Association for Computing Machinery, 17, No. 7 July 1974, pp. 365-375.

[78] Salus, p 59.

[79] Sandy Fraser, quoted in Salus p. 58.

source code. Dispite the claimed "secret" nature of the system, Unix has profoundly shaped operating system research for a quarter of a century.

The early versions of Unix evidence some confusion over the proper form of protection, particularly in the interaction between trade secret and copyright regimes. While documentary evidence from the earliest Unix versions is largely unavailable, scattered copyright notices are present, some dating back as early as 1972. By the 5th Edition of Unix, released in 1974, copyright notices were present on most source files, but missing from many of the small utilities and from the C library. Some of this inconsistency can be attributed to the fact that Unix at this time was not seen as a project of major significance within AT&T. In 1974, there were only on the order of 50 Unix installations worldwide.

Things got more complicated around Version 6. As the release was prepared, lawyers for Bell Labs asked the authors of the software to make sure that copyright notices appeared on every file that was going out, including data files. A flurry of activity ensued, which included modifying the programs that could not properly read the copyright adorned data files. At this point a few tapes were sent out to the first licensees. Almost immediately, the Bell Labs lawyers reversed their position, fearing the specter of trade secret preemption, and asked the programmers to go back and remove all of the notices. The few licensees who had already received tapes soon got a follow-up letter promising a quick "upgrade" if the tape was returned. The upgrade, the official Version 6 distribution, was ready three weeks after the first batch was prepared, on July 18 1975. Even the online manual that was sent out with this copy of Version 6 lacks a copyright notice, and instead bears this warning

```
The enclosed UNIX documentation is supplied
 in accordance with the Software Agreement
you have with the Western Electric Company.
```

Steve Johnson summed up the incident this way.

> In my view, this era was the first, and very extreme, case of something that has happened many times since then — a severe clash between the speed at which legal matters move and the speed of software technology. AT&T lawyers, and AT&T generally, did planning to a 20-year timeframe before divestiture. It seemed perfectly appropriate to AT&T lawyers, when faced with any question, to wait five years and see if it went away. Needless to say, this wasn't much help to technologists.[80]

The conflict is of course the technologists desire for wide dissemination versus the lawyers' desire to maintain strong control. Even after 1980, when Congress enacted legislation ensuring copyright protection for computer software,[81] AT&T continued to distribute its source code under trade secret protection, and in fact would make that protection more explicit in their licensing agreements at the same time the copyrightability of software was gaining acceptance. The copyright notices, hastily removed in the summer of 1976, did not reappear until 1984.

Unix shipped with extensive online documentation detailing how to use and program for the system. The standard license agreement for 6th Edition Unix treats documentation and source code identically. Licensees were expected to keep the online and printed documentation secret to the same degree that that they agreed to protect the source code itself. That is to say, licensees were required to "hold the LICENSED SOFTWARE in confidence" and disclose it only to employees and students "to whom such disclosure is necessary to the use for which rights are granted hereunder." Furthermore, the license requires that each licensee "shall appropriately notify each employee and student to whom any such disclosure is made that such disclosure is made in confidence and shall be kept in confidence by him."[82]

---

[80] email message on March 31, 2001.

[81] Computer Software Copyright Act, Pub. L. No. 96-517 (1980) (codified at 17 U.S.C. § 117).

[82] Unix Educational Software License Agreement for Unix 6th Edition §4.05, dated May 12, 1977, on file with author.

5.4. Unix Patents

Interestingly, given AT&T's early advocacy of software patents,[83] only two patents were ever filed on the core Unix system. The first covered the original file system designed in the summer of 1969. However, this patent never issued — AT&T stopped pursuing it following resistance from the PTO.[84] The second patent, filed on July 9, 1973, was granted on January 16, 1979.[85] This patent, titled "Protection of Data File Contents," covers an important technique used in the Unix security model and is implemented in every non-trivial variant or clone of the Unix system. Nevertheless AT&T abandoned the patent to the public domain during the 1980's. The inventor, Dennis Ritchie, said that

> It wasn't until the Consent Decree of about '82 (effective '84) that AT&T decided to get into the computer biz, and dedicating the patent looks and looked in retrospect fairly silly, although it wasn't until much later that serious IP issues began to happen — with or without the patent, there weren't any other serious Unix clones for quite a while.[86]

5.5. How to Teach a Trade Secret

If one thing ought to be clear under traditional state trade secrecy law, it is that you can't teach a trade secret in a University course and expect it to stay a secret. This is exactly what happened in Australia in 1976.[87] Ken Robinson at the University of New South Wales read Thompson and Ritchie's article in the Communications of the ACM and was one of the many people who requested a copy of Unix from Bell Labs. A copy arrived in December, 1974, at a cost of $150.[88] His colleague at UNSW, John Lions, quickly saw how valuable Unix could be for teaching. Lions had been teaching operating systems to advanced undergraduates, but was disappointed with the options he had for his courses. Before Unix arrived, he had used the "general principles" approach to operating systems education, in which students are exposed to the fundamental principles of operating systems without actually getting to see the inner workings of a real system. He found this preferable to the "building block" approach where students would attempt to build their own "toy" operating system, because the practical limitations of such a course would not allow the students to be exposed to the full complexity of a real system. However, Lions would have much preferred to use the case study approach in which students would "undertake a detailed analysis of an existing system."[89] In 1976, the Unix operating system was put to use for that purpose with the introduction of two new one semester undergraduate courses. Unix was seen by Lions as perfect for teaching because it was the only system small enough to be comprehensible yet powerful enough to be interesting.[90] Lions said that the Unix license his university had was not explicit enough to forbid teaching it in this way.[91]

In order to facilitate the teaching of those courses, John Lions wrote a two volume book. The first volume contained most of the source code to the 6th Edition of Unix, with the addition of detailed line-by-line comments. The second volume, completed in 1977, was a commentary on the system, giving both an overview of the implementation and a description of the components of the design, down to the level of individual subroutines and data structures. These books were distributed to students in Lions' case study courses, and were intended to supplement the

---

[83] See, e.g., Susan H. Nycum, Legal Protection for Computer Programs, The Computer/Law Journal, Volume 1 (1978-1979). "It can safely be said that Bell Labs has been in the vanguard of the forces arguing for patent protection [for software]." Id. at 74.

[84] FIXME: cite?

[85] U.S. Pat. No. 4,135,240.

[86] Email message on April 4, 2001.

[87] See generally Rachel Chalmers, "Code critic," Salon Magazine. http://salon.com/tech/feature/1999/11/30/lions/index.html

[88] Michael Hauben and Ronda Hauben, Netizens, Chapter 9. Available at http://www.columbia.edu/~rh120.

[89] Lions, i-1.

[90] Id.

[91] Salus, p 130. The license in question does not appear to be available.

documentation that was provided by the original authors of Unix. The new books filled a gap in documentation, providing a description of the system at the implementation level, rather than at a level appropriate to an end user or programmer.[92]

The researchers at Bell Labs gave high praise to the book, and were very pleased to see a large number of well trained Unix users coming out of John Lions' classes and others like them. However, from the perspective of those within AT&T who were increasingly seeing Unix as a potentially profitable product, "the very value and vividness of the Lions commentary compelled caution."[93] AT&T did not allow the book to be published. By the time the 7th Edition of Unix was released in 1979, AT&T had changed their standard license agreement to prohibit University teaching of its source code. The clause read

> LICENSEE shall not include in its curriculum any course of instruction in which the source code or other presentation of the internal operation of the LICENSED SOFTWARE is disclosed or discussed, or prepare or publish any documentation disclosing or describing such code or presentation.[94]

All subsequent Unix Educational licenses contained this "John Lions clause." AT&T found the books to be very valuable and incorporated them for use in their internal training programs. AT&T assumed responsability for the distribution of the books, and did so under extremely restrictive terms. Each licensee of the Unix system was allowed to obtain a single copy of the book, direct from AT&T. Copying by licensees was strictly prohibited. Even employees of Bell Labs were not allowed to make copies of the book. A copy was issued to them with their name and a serial number, under the requirement that it be returned when no longer needed.[95]

Dispite the restrictions on the book and the prohibition on university teaching, the Lions commentary remained extremely popular. Michael Tilson reports, in the preparatory notes to a recently published version of Lions' commentary, that this book educated "a generation of operating systems designers."[96] Through photocopying, it became "one of the most widely distributed underground computer science documents."[97]

AT&T's choice of intellectual property regime plays into this story significantly. Volume 1 of the Lions book is a slightly modified and commented listing of the Unix 6th Edition source code. Because of its overwhelming textual similarity to the code distributed by AT&T, this volume would undoubtedly qualify as a derivative work that could have been suppressed under copyright law. On the other hand, the second volume is merely an extremely detailed, exhaustive "literary criticism" of Unix, arguably precisely the kind of work privileged by the fair use rights of comment, criticism, and teaching.[98] Additionally, for many readers, it was volume 2 that offered the most value, because most readers already had convenient access to the source code. Unix

[92] The introduction to the programmer's manual for the 6th Edition of Unix, written by Ken Thompson and Dennis Ritchie and dated May 1975, recognized this gap in documentation, saying that "[This manual] provides neither a general overview — see "The Unix Time-sharing System (Comm. ACM 17 7, July 1974, pp. 365-375) for that — nor details of the implementation of the system, which remain to be disclosed."

[93] Dennis Ritchie, forward to Lions, p x.

[94] Educational Software Agreement for Unix Version 7, Dated February 20, 1981, on file with author.

[95] See Arpanet unix-wizards mailing list post titled "Lion's Book" by Brian Redman on June 14, 1981. Available from the Usenet A-News archive at http://communication.ucsd.edu/A-News/index.html.

[96] Michael Tilson, Lions, p ix. Tilson was president of The Santa Cruz Operation, Inc. (SCO) which, in 1995, acquired the rights to Unix and consented to allow the book to be published for the first time. The source code it contains, though long out of date, still has instructional value.

[97] Ken Thompson, Lions, p x. In this way, the Lions book is strikingly similar to the Real Book, an illegal, underground musical transcription book which helped train a generation of jazz musicians when good quality commercial substitutes were unavailable.

[98] 17 U.S.C. §107.

installations at universities tended to maintain the source code to the system online. Therefore, most any student with access to a Unix system could read the source code online or print it out and take home a copy.[99] Although the operating system had a facility for controlling access to files on a per-user basis, this was rarely applied to protect access to the source code. Besides, AT&T's licensing agreements arguably granted permission to show the source code to virtually any student studying computer science. Given widespread copying of the Lions book and massive student accessibility to online copies of the source code, it certainly appears that in actual practice, AT&T's intellectual property right behaved like a copyright in that they maintained the ability to control distribution, in the form of the number of operational Unix systems, without inhibiting widespread comprehension and criticism.

5.6. Bug Fixes

Shortly after the release of the 6th Edition of Unix, Ken Thompson discovered a set of fairly serious bugs, which he had been able to fix in the Bell Labs research version of Unix. He was about to go on sabbatical to work as a visiting professor at Berkeley, and he wanted to share these fixes with other Unix users. Lou Katz told this story:

> A large number of bug fixes was collected, and rather than issue them one at a time, a collection tape ("The 50 fixes") was put together by Ken. Some of the fixes were quite important, though I don't remember any in particular. I suspect that a significant fraction of the fixes were actually done by non-Bell people. Ken tried to send it out, but the lawyers kept stalling and stalling.
>
> Finally, in complete disgust, someone "found a tape on Mountain Avenue" which had the fixes. [The address of Bell Labs is 600 Mountain Avenue, Murray Hill, NJ.]
>
> When the lawyers found out about it, they called every licensee and threatened them with dire consequences if they didn't destroy the tape, after trying to find out how they got the tape. I would guess that no one would actually tell them how they came by the tape (I didn't).[100]

This episode clearly demonstrates the conflict between "the speed at which legal matters move and the speed of software technology" that Steve Johnson mentioned regarding the copyright notices on the 6th Edition. It also shows the growing tension between the developer community and the owner of Unix.

5.7. BSD

One important effect of AT&T's early refusal to offer support for Unix was that it forced users to band together in support of one another. As early as 1974, Unix users were holding informal gatherings around the world where they could share tips and bug fixes that they had developed. One of the universities that licensed Unix very early on was the University of California at Berkeley. The extraordinarily important work done at Berkeley would shape the course of Unix until the early 1990's when budget cuts and the rising cost of a Unix source code license caused them to stop further development on the system.[101] While many Unix licensees were modifying and adding to their copies, Berkeley was extremely successful in sharing its modifications with others. In addition, Berkeley served as a centralized place for the integration of software and modifications made around the world. The students and faculty at Berkeley did this by way of "Berkeley Software Distributions" which were created between 1977 and 1994 and sent out on tape to Unix licensees for a nominal fee. Bill Joy, who had been using Unix at Berkeley since 1975, was in charge of these distributions until 1982 when he left to become an employee of Sun Microsystems. The first two distributions were comprised entirely of software created at Berkeley.

---

[99] Regardless of the legality of these activities under copyright law, they were widely and routinely practiced.

[100] Salus, p 139. Lou Katz was an early Unix user at Columbia University.

[101] However, the development of Berkeley Unix continues to this day with an ever changing mixture of commercial and volunteer support.

In December, 1977, Bill Joy was involved in discussions with AT&T over the legality of the first software distribution, which contained a number of Berkeley developed tools but no AT&T owned code. A letter from the director of patent licensing at Bell Labs clarified the legal situation. It said that Berkeley's license agreement placed no restrictions on the use or distribution of software which did not contain any proprietary information belonging to AT&T. For software that did contain such information, Berkeley was within its rights so long as they restricted their distribution to verified holders of a Unix source code license. In addition, Berkeley was not allowed to use the name Unix or any other AT&T trademark.

A simple one page license agreement was prepared, entitled "conditions under which the computer software described below is furnished by the University of California at Berkeley."[102] Under the license, the licensee agreed 1) to pay for the cost of duplication, 2) not to use the software for commercial purposes or distribute it without permission, 3) to give credit to the authors and the University when the software is published or used, 4) to accept a disclaimer of warranty, and 5) to warrant that they held a current Unix source code license from AT&T. Bill Joy sent out about thirty tapes containing the distribution, which consisted of a new editor called ex, an implementation of the Pascal programming language, a "star trek" game and a variety of other programs. The tape was distributed in exchange for a $50 duplication charge (not a license fee). A label on the tape read

> `The contents of this tape are distributed to UNIX licensees only, subject to the software agreement you have with Western Electric and an agreement with the University of California.`[103]

An update soon occurred in the middle of 1978 which was called the "Second Berkeley Software Distribution," or 2BSD. Bill Joy again managed the distribution and shipped nearly seventy-five copies over the next year.

Starting with the 3rd Berkeley Software Distribution in December 1979 (3BSD), Berkeley began shipping a full operating system rather than just a collection of software. The new system was derived from the Unix/32V distribution from Bell Labs. 32V was significant because it was the first version of Unix to run on the new VAX[104] computer from Digital Equipment Corporation. At this time, although the portability of Unix had been demonstrated, it was still predominantly tied to the 16-bit DEC PDP-11 line of computers. The VAX was the successor to the PDP-11, and as a 32-bit machine, it could utilize far larger amounts of memory.[105] Four Bell Labs engineers ported Unix to the VAX and had a working system by August 1978. By the time it was running, people were already asking for copies of it. Distributing it proved to be difficult, however. Charlie Roberts, the manager of the porting project, said

> I went to Roy Lipton and Al Arms in Patents and Licensing about getting it out. After a lot of back-and-forth they decided that we could give it to one university for research purposes and that Al would set up a "special research agreement" with that institution.... So, with the blessings of BTL Area 11 management, we sent 32V to Berkeley. It was October or November, 1978.[106]

Letter from E.G. Baldwin to Berkeley professor Susan L. Graham, on file with author, written in response to a letter from Susan L. Graham, dated January 18, 1978, on file with author.

[102] License agreement on file with author.

[103] Id. BSD distribution tape provided to the Unix Heritage Society by Keith Bostic. Western Electric was AT&T's subsidiary in charge of handling Unix licensing at that time.

[104] "Virtual Address Extension."

[105] "The most significant architectural enhancement that the VAX-11/780 provides over its predecessor, the PDP-11, is the very large address space made available to user programs." Ozalp Babaoglu, William Joy and Juan Porcar, "Design and Implementation of the Berkeley Virtual Memory Extensions to the Unix Operating System." This paper was distributed with Third Berkeley Software Distribution, on file with the author.

[106] Charlie Roberts, quoted in Salus, p 154. I do not have any other evidence of the "special

Berkeley made substantial improvements to 32V, most notably the support for the virtual memory capabilities of the VAX architecture. The third Berkeley Software Distribution (3BSD) was sent out towards the end of 1979 and proved to be very popular. Since AT&T's 32V system did not support virtual memory, users who wanted to use it had two choices: 3BSD or Digital Equipment Corporation's brand new VMS[107] operating system. Many early VAX customers who were upgrading their PDP-11 Unix systems naturally moved to BSD, and BSD became a significant rival to VMS. John Gilmore said that "what we did with BSD was to essentially take the VAX market from DEC." While VMS remained a popular choice for commercial users until the demise of the VAX product line, Unix was (and continues to be) favored in many educational and research settings.

The "duplication charge" for 3BSD was $200, a tiny amount at the time for a powerful 32-bit operating system. In fact, distribution of BSD at cost was probably mandated by Berkeley's license agreement. However, probably as a result of the short lived "special research arrangement," 3BSD was also licensed as a trade secret.[108] The 3BSD license agreement was written in much more formal language and runs to four pages. The transaction was explicitly characterized as a lease rather than a sale, and the licensee was entitled to only one installation of the software. The licensee was also required to hold the software in confidence, and in addition,

> THE SOURCE FORM OF LICENSED MATERIAL SHALL NOT BE DISCLOSED TO OTHER LICENSEES WHETHER OR NOT SUCH OTHER LICENSEES HAVE CURRENT VERSIONS OF THE LICENSED MATERIAL.[109]

In other words, the licensee was prohibited, by Berkeley, from doing just what AT&T had granted Berkeley permission to do.

This prohibition applied to documentation as well. 3BSD included detailed descriptions of the implementation of its new features. The copyright notice on that documentation granted permission to copy only "as necessary for licensed use." However, Berkeley was apparently not worried about trade secret preemption due to notice of copyright. The source files for new code created by Berkeley have always had notices since 2BSD.

By August 1, 1981, Berkeley and AT&T had entered into a more liberal license agreement for the Unix 32V source code. The license, dated August 1, 1981, permits the use of the software for research

> provided that the results of such research are not intended primarily for the benefit of a third party and that such results are published and are available to the public for no more than reproduction and shipping charges.[110]

The development of software for commercial sale or license was prohibited.[111] AT&T actually required that the work be "published" and "available to the public," language evidently designed to preclude any claim of trade secrecy over the University's work. The use of the word "published" is curious given AT&T's assertion that the source code to their own 32V system was maintained as an unpublished work. 32V remained unpublished, but Berkeley's modifications to it were required to be published. This is as if the Princeton Review required it's test proofreaders to publish their corrections, which presumably would be meaningless without access to the

research agreement." 32V was soon widely licensed to other universities.

[107] "Virtual Memory System."

[108] The "special research arrangement" was superceded by a license for Unix 32V dated August 1, 1981 (on file with author).

[109] License Agreement for Third Berkeley Software Distribution (3BSD), emphasis in original, on file with author.

[110] Software Agreement for Unix 32V, dated August 1, 1981, §2.01, on file with author.

[111] Id.

underlying secret. The 1981 license still required that Berkeley "hold the licensed software in confidence."[112]

But what would happen if the "corrections" became so extensive as to constitute a complete rewrite of the underlying secret material?

Although AT&T continued to enhance Unix, Berkeley based all subsequent versions of BSD on Unix 32V. As such, it is a direct ancestor of a variety of modern BSD derived operating systems, including Apple's OS X and the free FreeBSD, NetBSD and OpenBSD systems.

No provision prevented Berkeley from maintaining copyright ownership over their modifications. Berkeley would take advantage of this fact four years later when it licensed it's software back to AT&T for incorporation into the "official" version of Unix. However, for a variety of reasons, BSD remained a "competitor" to AT&T Unix. The difficulty was that while Berkeley was adding a great many features, AT&T was often slow to incorporate them. For the most part, academic users were perfectly happy with BSD and didn't care about it's lack of official support. However, as Unix moved towards commercialization, compatibility problems between the different versions began to become an issue. It is widely believed that the slow adoption of user enhancements into AT&T Unix was a major cause of this difficulty. John Gilmore, who was employed by Sun Microsystems in the mid-80's, said that "if it weren't for the incompetence of AT&T in accepting fixes, there would have been no BSD." Instead, improvements from Berkeley and the wider user community would have been quickly and steadily incorporated into the main AT&T system and there would have been no reason to maintain a distinct "branch" called BSD. Berkeley in effect acted as a "publisher" for Unix, accepting patches and additions from the community and rapidly putting together a coherent package. Throughout the 1980's BSD would rival AT&T Unix in popularity, and although the split was sometimes acrimonious, there is no question that the contributions made by Berkeley were extremely significant both for Unix and for the computer industry as a whole. The dynamics of software development communities of varying degrees of openness often resemble this structure. Licensing considerations are significant in so far as they permit competing project managers to usurp control from incompetent originators, but licenses are mere agreements, and they are not, by themselves, the determining factor in the long-term evolution of a product.[113]

In the mean time, Berkeley's work on the Unix system attracted the attention of the Department of Defense Advanced Research Projects Agency (DARPA). Berkeley was granted a contract to improve the performance of the system, and later to add TCP/IP networking, a suite of networking protocols designed to facilitate the interconnection of local and wide area computer networks. The fact that Unix was portable and was in use on more than one hardware platform was a big factor in DARPA's decision to support Unix. Unix was extraordinarily popular around universities and research institutions by the early 1980's. Integration of TCP/IP would allow the many Unix using universities around the world to connect to a growing number of wide area networks, including the ARPANET, which adopted TCP/IP in 1983. ARPANET, created in the late 1960's, formed the first backbone of the Internet at that time.

Unix and the Internet grew together throughout the 1980's.[114] A related Berkeley contribution from this time was Sendmail, written by Eric Allman, which today runs the majority of Internet email servers.[115] Sendmail was written to route email between a wide variety of networks that were in use at Berkeley, during a time when mail routing protocols were undergoing rapid change. It's flexibility allowed it to grow along with the Internet to enjoy it's now dominant position.

The graduate students at Berkeley and the researchers at Bell Labs maintained a close working relationship. In 1980, Bill Joy flew to New Jersey to help install 4BSD at Bell Labs. The

[112] Id., §5.06.

[113] FIXME — Flesh this idea out into a section?

[114] ARPANET supported many other systems including ones from IBM and DEC. However, Unix was a very significant test bed for new networking technologies, many of which were created or enhanced at Berkeley.

[115] See http://www.sendmail.com/company/overview/. There are an estimated 2 million copies of Sendmail installed worldwide. http://www.sendmail.com/partner/.

significant technical contributions Berkeley made, including its DARPA funded work in networking, soon led AT&T to seek a license for the commercial use of BSD. First on March 9th, 1983, and then again on March 4th, 1986, AT&T entered licensing agreements for Berkeley software.[116] Berkeley's restrictions were simple. In exchange for a $1,000 distribution fee, Berkeley granted AT&T an unlimited license to modify and sublicense the 4.3 version of BSD. While the title to the software would remain with Berkeley, AT&T was not bound by any trade secret provision. The only significant limitation on AT&T was in paragraph 8, requiring "proper credit and recognition." AT&T was required to acknowledge the contributions of Berkeley and BSD's "Other Contributors" in their printed documentation.[117]

5.8. Commercial Unix

By the early 1980's, AT&T was well on its way to settling the 1974 antitrust suit. AT&T management agreed to accept divestiture in exchange for the right to enter unregulated markets, particularly computers. The management of Unix during this time was based around the plan that commercial exploitation would take place as soon as it was permitted. The modified final judgment was handed down by district judge Green in August 1982 and became effective in 1984. Thus, during those years, AT&T was preparing to enter the computer business, and Unix was a major part of its strategy. Although Unix's transition to a commercial system had been gradual, AT&T had adhered to the "no advertising, no support" directive. While most Unix users were in university or research environments, AT&T had been granting licenses for commercial use since the mid 1970's. In addition, AT&T had sold licenses giving permission for other companies to commercialize Unix. Microsoft bought such a license in 1979 and began to market Xenix, which ran on microcomputers and was based on Unix version 7. Xenix in turn was marketed by The Santa Cruz Operation (SCO) and became one of the most commercially important varieties of Unix.

As AT&T moved towards commercialization of Unix, they began to offer binary-only licenses, at a reduced cost compared to the increasingly expensive source code licenses. In 1983, AT&T annouced the availability of Unix System V as a commercial product, and the introduction of a line of microprocessor based personal computers to run it. Fairly soon, there was a dramatic proliferation in the number of companies supporting and selling Unix. By 1988,

> Apollo, DEC, Eakins, Gould, Integrated Solutions, Masscomp, mt Xinu, NSC and Wollongong were among the companies marketing Berkeley Unix. Among those marketing AT&T System III or System V derivatives were: AT&T, Altos, Apollo, Compaq, Convergent, HP, Honeywell, IBM, ITT, Intel, Interactive, Masscomp, Microport, Microsoft, Motorola, NCR, NUXI, Opus, SCO, Silicon Graphics, Sperry, Sun, Tandy, UniSoft, and Wollongong. Furthermore, Amdahl, Apollo, Apple, Cray, DEC, Data General, HP, IBM, Intel, Motorola, Unisys and a host of others offer proprietary versions of Unix, several of which are 4.2BSD-based.[118]

Despite heroic efforts at standardization, these systems inevitably evolved incompatibilities. Competitive pressures made it difficult for so many companies to work together, and the necessities of licensing made it difficult for technology to be shared among them. On the other hand, as the list above indicates, the wide availability and relative standardization of Unix made it an ideal platform for launching a new company.

Founded in 1982, Sun Microsystems combined a cheaply available microcomputer design (the Stanford University Network (SUN) board), and a cheaply available operating system (Unix). Andy Bechtolsheim, the designer of the SUN board, was one of the founders of Sun Microsystems. The young company soon hired Bill Joy from Berkeley and announced their intention to switch from a commercial version of Unix to 4.2BSD. Sun bought a commercial license to Unix System

[116] The 1983 agreement appears to be confidential, but it is referenced in the 1986 agreement, on file with the author. The discussion here is based on that license.

[117] FIXME — What did the other BSD licenses look like at this time? Sun? All the other commercial BSD vendors? non-commercial BSD licenses?

[118] Salus, p 210.

III (later, System V) from AT&T to cover their commercial use and relicensing of the software. Then they bought a single copy of BSD from Berkeley. When the Unix tape arrived from AT&T, they simply threw it away. BSD was a full system, and all they needed from AT&T was the right to use and sell it.

The June 1982 issue of the USENIX[119] newsletter mentioned Bill Joy's new job, and the new company's stance on its relationship to Berkeley.

> While SMI [Sun Microsystems, Inc.] may need to develop proprietary software in certain specialized areas, Bill expects fixes to the shared base of 4.2BSD programs which are made at SMI can be distributed by Berkeley. The current cooperative efforts between CSRG and various industrial groups are seen as a model for the relationship....[120]

Sun would contribute significant software to the Unix community, but they soon changed course to pursue a more proprietary approach.

5.9. Unix Licensing in the early 1980's

One of the new features in AT&T's educational license agreement around this time was the way it dealt with derivative works. Computer scientists had long known that users of the system were very valuable for their ability to fix bugs and add features. This was especially true of Unix, whose customers were not ordinary users. They were predominantly other computer scientists who had the source code and were forced by AT&T's lack of support to provide techical support for themselves and for each other. Also, AT&T's experience with BSD showed them the potential commercial value of user driven innovation, and therefore the value (to AT&T) of the right to capture that innovation. AT&T's new licensing agreements explicitly granted the right to create derivative works based on Unix (a right that was understood but never explicitly stated in the earlier agreements discussed above). The new agreements also required the licensee to treat derivative works as if they were a part of the software originally provided by AT&T.[121] This last practice was discontinued early in 1985, possibly to avoid the appearance of a grant back agreement, which could be a questionable practice from an antitrust perspective.[122] AT&T did not want to be claiming ownership over original work that just happened to be done on a Unix system. However, they wanted to make sure that granting the right to make derivative works did not eliminate AT&T's rights in the portion of those works owned to AT&T. The sharing behavior of the Unix licensees was influenced by AT&T's true legal rights, AT&T's explicitly negotiated contractual rights, and AT&T's public behavior representing their willingness to enforce those rights. This clause acted, in a way, as a valve, by which AT&T could attempt to adjust the balance between sharing and control. In this particular instance, AT&T's fear of overreaching their true legal rights encouraged them to publicly clarify their position, and perhaps for that reason a little bit more invention was able to take place in the Unix community.

Another change that took place was the reversal of AT&T's stance on copyright notices. An AT&T memorandum dated July 16, 1984 introduced the practice of affixing copyright notices to all source code files.[123] The copyright notices that were added marked AT&T's Unix source code as a copyrighted but unpublished work.[124] This is in keeping with the general software industry

[119] USENIX is an international Unix users association.

[120] Salus, p 200. This quote mirrors one from an early Apple computer advertisement which read "Our philosophy is to provide software for our machines free or at minimal cost." Apple and Sun both reversed their liberal positions in due time, however. Soon after Joy spoke, Apple won a landmark legal victory in Apple v. Franklin, 714 F.2d 1240, establishing for the first time the copyrightability of binary code stored in a read only memory. Today, both Apple and Sun remain proprietary hardware vendors but produce software under a variety of licenses (Darwin — license?, Star Office, SCSL, SISL, etc.).

[121] Educational Software Agreement for AT&T System V, dated July 1, 1983, on file with author.

[122] The change in the licensing agreement was spelled out in a May 15, 1985 letter to the University of California at Berkeley, on file with author. Presumably similar letters were sent to other licensees.

[123] AT&T Memorandum For Record, cited in 1993 U.S. Dist. LEXIS 19503, note 2.

[124] See the introduction to chapter 7.

understanding that trade secrets and copyrights on computer software are not incompatible.

6. Results

The widespread use and profound influence of Unix suggest that, although AT&T failed to become a major player in the computer industry, their relatively liberal stance on intellectual property does not seem to be the reason for that, and in fact it was quite likely more beneficial than harmful.

> As things have turned out, the ultimately generous arrangements AT&T made over the years didn't work out all that badly from an IP standpoint. AT&T fared poorly in the computer biz not because of this kind of issue, but just because we didn't have and support the right product line.[125]

AT&T's licensing scheme allowed Unix to become an industry standard. Because of the wide distribution of its source code, people relied on it to be continually at the cutting edge of software technology. The relative simplicity of Unix, coupled with its implementation in a high level language, meant it could be ported from one hardware platform to another with relative ease. The lack of centralized technical dictatorship allowed niche variants of Unix to survive when a traditional operating system vendor might have killed them off to avoid support costs. By not depending on the network effects that would have compelled a single company to consolidate its users on a single hardware and software platform, Unix was able to grow in many directions, forming the basis for a large number of new ideas and new companies.

[125] Dennis Ritchie, email on April 4, 2001.

7. Software Copyright

7.1. Copyright Revision and CONTU

Throughout this century, there has been a continuing tension between the development of technology and copyright law. It seems as if every advance in communications or computing technology has been accompanied by a new strain on the copyright statute.[126] By 1955, technological changes, from sound recordings to motion pictures and broadcast radio and television, had convinced Congress that the 1909 Copyright Act was woefully out of date. A 21 year effort to completely rewrite copyright law culminated in the 1976 Copyright Act. This was perhaps only possible because two of the most significant technological developments of the time were explicitly excluded from consideration: photocopying and computers.[127] Instead of allowing these issues to derail copyright reform, Congress created the National Commission on New Technological Uses of Copyrighted Works (CONTU). In order to give the Commission time to present its recommendations, Congress explicitly wrote into the 1976 Act that rights in computer related works would not be modified, but instead would be handled by courts under the 1909 Act.[128] Presumably, Congress had the intention of waiting for CONTU's recommendations.

In 1974, the 93rd Congress enacted a bill creating the commission,[129] giving them three years to compile data and make their recommendations. Their field of study was to be

> (1) the reproduction and use of copyrighted works of authorship —
>
> (A) in conjunction with automatic systems capable of storing, processing, retrieving, and transferring information, and
>
> (B) by various forms of machine reproduction, not including reproduction by or at the request of instructions for use in face-to-face teaching activities;
>
> and
>
> (2) the creation of new works by the application or intervention of such automatic systems of machine reproduction.[130]

The commissioners were selected by Gerald Ford and included representatives from "copyright owners, copyright users, and the public"[131]

There were no representatives of the computer or photocopying industry on the commission. Nevertheless, CONTU interpreted its charter broadly and spent a great deal of its time investigating the issue of copyright protection for computer software, even though that issue was not mentioned in its charter.[132] Of the 87 pages of their Final Report devoted to the issue of computers and copyright, 70 were devoted to analysis of the software copyright issue, with the remainder discussed databases and new works created with the use of a computer. While the copyright office

[126] See generally Final Report of the National Commission New Technological Uses of Copyrighted Works (CONTU) (1978) pp 5-9.

[127] Id. at 6.

[128] Id. at 11, Citing 17 U.S.C. §?? (1976 version).

[129] Pub. L. No. 93-573.

[130] CONTU Final Report at 8.

[131] CONTU Final Report, p 9. The copyright owners were John Hersey, President of the Authors League of America; Dan Lacy, Senior Vice President, McGraw Hill, Inc.; E. Gabriel Perle, Vice President-Law, Time, Inc. and Hershel B. Sarbin, President, Ziff-Davis Publishing Co. The copyright users were William S. Dix, Librarian Emeritus, Princeton University (Commissioner Dix died on February 22, 1978.); Arthur R. Miller, Professor of Law, Harvard Law School; Robert Wedgeworth, Executive Director, American Library Association and Alice E. Wilcox, Director, Minnesota Interlibrary Telecommunications. The members of the public were George D. Cary, retired Register of Copyrights; Stanley H. Fuld, retired Chief Judge of the State of New York and the New York Court of Appeals; Rhoda H. Karpatkin, Executive Director, Consumers Union and Melville B. Nimmer, Professor of Law, UCLA Law School.

[132] May be in the congressional record. look up Senate Report 94-473 (mentioned in CONTU Meeting 5).

had been accepting registrations for computer software copyrights since 1964,[133] they did not express an opinion on their validity, leaving that decision up to the courts. CONTU ultimately recommended that computer software be copyrightable, and Congress enacted that recommendation into law in 1980.[134]

Although personal computers had only been available for a very short time in 1978, CONTU recognized their significance, and the changes they would bring to the software industry. CONTU was influenced by two related trends:

> Computers have become less cumbersome and expensive, so that individuals can and do own computers in their homes and offices with more power than the first commercial computers, while at the same time, programs have become less and less frequently written to comply with the requirements imposed by a single-purpose machine.[135]

Fredrick Brooks wrote in 1986 that the incredible drop in computer hardware costs brought about by the microcomputer changed the dynamics of the software industry. "The personal computer revolution has created not one, but many, mass markets for software."[136] He went on to state his belief that the ability to purchase software on the mass market was "the most profound long-run trend in software engineering."[137]

7.2. Classic Use of Copyright

The Copyright Act of 1976 protects an original work of authorship fixed in any tangible medium of expression[138] and grants to that author the exclusive right to reproduce the work, prepare derivative works, and perform or display the work.[139] These rights derive from a constitutional grant of power to Congress to "promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[140]

A legal system usually justifies a copyright statute in one of three ways.[141] Some systems base the right on the idea that an author, as the creator of a work, has a moral right to control its use and dissemination. Regimes that favor the moral rights theory generally provide authors with broad control over the creation of derivative works. The other two predominant theories are often related in American law, each finding strong support in the language of Article I, §8 of the U.S. Constitution. One theory justifies copyright as a bargain. In exchange for creating a work and making it available to the public, the public in return grants the author a series of exclusive rights. The core notion of this theory is public availability: if the public cannot benefit from access to the author's expression, there is no bargain and the author deserves no copyright, although a right of trade secrecy may still apply. The third theory is economically based and justifies copyright as an incentive. Congress' power to promote the progress of science is justified by any measure deemed to increase the net output of works of authorship. Virtually any type of control that provides a monetary benefit to an author may be justifiable under this theory.

In the context of a book, for most useful purposes (and excluding the moral rights view), copyright amounts to granting the author control over how many copies of the book will be in existence. If you want one more copy of the book to exist, you need to ask the author to create one, or negotiate a license so that you can create one yourself. Under that interpretation, a copyright may be described as primarily a distribution right. The author gets paid in proportion to the number of times a new copy of the work is distributed to the public. Assuming I am not

---

[133] Copyright Office Circular 31D (January 1965).

[134] Computer Software Copyright Act, Pub. L. No. 96-517, (1980) (codified at 17 U.S.C. § 117).

[135] CONTU Final Report at 24.

[136] Brooks, No Silver Bullet at 197.

[137] Brooks' argument to buy, rather than build software is even more compelling in a free software context which bypasses virtually all avoidable transaction costs of software acquisition.

[138] 17 USC §102.

[139] 17 USC §106.

[140] U.S. CONST. art I, §8, cl. 8.

[141] Cite?

making any new copies of the work available to the public, there is no reason to think that the author would care what I do with a book once I've purchased it. Additionally, the author is not trying to prevent me from knowing the content of the work until I have paid. I am perfectly free to read it in a library or borrow it from a friend. People pay for books for the convenience and satisfaction of owning their own copy.

Book and record producers place their goods into the stream of commerce, relying on their exclusive rights to protect their revenue stream. Copyright promises that each copy sold will result in income. The expressive content of these works is meant to be comprehended by the end user, and as such no special restrictions are placed on its use or enjoyment beyond those that would create additional competition for the author. Copyright is convenient in this context because it does not require any contractual relationship between author and end-user.

7.3. Realities of the Software Business

Thinking about software in this context makes one thing immediately clear. The business of software is fundamentally different from the business of producing all other copyrighted works. This is because software is not meant to be read or listened to. It is not meant to be comprehended. It is meant to do something.[142]

Long standing business practices in the software industry reveal this conflict. Philip Dorn presented some of these facts to CONTU. The software industry, unlike the book publishing industry, is not made up of individual authors who make money by selling copies of published, copyrighted works.

> ... the problem is not protecting individuals and their right to publication; it is not the problem that most of us are faced with ... the problem that we are faced with is the ability to sell a program or to lease a program to a buyer and make sure he doesn't steal it, copy it and go on beyond that. I would suggest to you gentlemen that we have been doing that for considerably long periods of time; it's perfectly routine, conventional contract work.[143]

In fact, software has almost always been sold (or "leased") under terms of confidentiality. Furthermore, measures such as the concealment of source code and contractual prohibitions on reverse engineering are frequently employed to impede comprehension of the copyrighted work. This practice is widely shared in the software industry, dispite the fact that their enforceability has almost never been tested.[144] Regardless of the contractual enforceability of such provisions, modern American law seems to find no problem granting copyright protection to a work specifically designed to be incomprehensible.[145]

The practice of software development reveals an equally significant distinction between software and other copyrighted works. Virtually all reasonably complicated software depends on other software for its use, whereas a book "does not require the publication of six other books in order to be useful."[146] While a book is generally a complete work, software is constantly

[142] See e.g. Pamela Samuelson, Contu Revisited: the Case Against Copyright Protection for Computer Programs in Machine-Readable Form, 1984 Duke L.J. 663. Of course, while the primary purpose of software is to perform some function, it is essential for the developers of a program to be able to read and understand its source code. Likewise, access to source code is an important prerequisite to learning, commentary and improvement by others.

[143] Philip Dorn, CONTU Commission Meeting 16, September 15, 1977, page 23.

[144] "To judge by the flux of law review articles discussing shrinkwrap licenses, uncertainty is much in need of reduction--although businesses seem to feel less uncertainty than do scholars, for only three cases (other than ours) touch on the subject, and none directly addresses it." ProCD v. Zeidenberg, 86 F.3d 1447 (7th Cir. 1996) at 14. In ProCD, Judge Easterbrook assumed, but did not hold, that the database covered by the shrinkwrap license at issue was not copyrightable.

[145] "the Copyright Office now accepts permissive registrations based on deposits of small 'identifying portions' of a program, and recognizes a further exception from disclosure if the copyright owner claims trade secrets in a program." Miller, Anything New Since Contu, 106 Harv. L. Rev. 977 at 989, citing 37 C.F.R. § 202.20(c)(2)(vii)(A)-(B) (1991).

[146] Philip Dorn, CONTU Commission Meeting 16, September 15, 1977, page 19-20.

undergoing revision and can never be said to be truly "finished."

> There has always been the theory in the industry that we put the program out forty per cent of what it ought to be, and we let the users in the community tell us what's wrong, and we fix it according to their needs, and requirements.[147]

The creation of software is in a sense a discursive process, built on a base of other software, constantly directed by user feedback. While this is commonly understood to be the case for free software, it is also true for virtually all proprietary software as well.[148]

Of course, given the option, authors will protect their work in all possible ways. Judicially untested shrink wrap agreements purport to create confidentiality obligations at no cost to the software vendor and without any negotiation, and Copyright Office practices do not let the deposit requirement interfere with any trade secrets contained in the work. Trade secret law is traditionally thought of as a "leaky" form of protection, applying only to parties that are contractually bound to maintain it and susceptible to accidental disclosure and other problems. Simultaneous application of Federal copyright law which allows for triple damages and the recovery of attorneys fees, can plug these leaks. It is therefore not surprising that these agreements have not been tested in court. Today, software publishers effectively get all of the benefits of trade secrets with none of the risks.

7.4. What Happened?

So copyright does not look like copyright when applied to software. Instead, it is a safety valve on a trade secret. CONTU was well aware of this conflict, but declined to dispose of it. The Software Subcommittee report, published in 1977 (??) discussed the problem of a set of "more or less mutually exclusive forms of protection"[149] and the need for clearly defined boundaries between them. A subsection on preemption stated that, under 17 U.S.C §301,[150] "trade secrecy protection could not be asserted where its purpose was to prevent the copying of a work. One who seeks revenues from proprietary software "ought not to be entitled to allege that his work, although available to users, is somehow secret.... as seems self evident, trade secrecy could not be relied upon if a program were widely marketed or otherwise broadly distributed."[151] The Subcommittee report went on to express the desirable effect of any reduction in secrecy, both in terms of reducing the need for duplication of inventive work, and the reduction in the complexity created merely for the purpose of making software incomprehensible.

Most statements to this effect were weakened before publication of the final report. The Commission recognized that trade secret "is inappropriate for protecting works that contain the secret and are designed to be widely distributed" and went on to recognize the confusion caused by the lack of uniform national trade secret law. However, they failed to resolve these issues or to recommend any constructive solution for Congress, opting instead to preserve an unstable status quo.[152] They argued for the preference of the use of copyright over trade secret,[153] but did not ask

[147] Dorn, CONTU Meeting 16, p 21.

[148] While it is certainly true that conventional literary works build on other works and may be more enjoyable or valuable when read in context, the nature of software is of a different character. A Windows program is probably worthless to a Macintosh user. Software typically has absolutely no value when its dependencies (in this example the various components of the Windows operating system) are missing from a consumer's computer system.

[149] Software Subcommittee Report, footnote 6.

[150] §301 expressly preempts state trade secret law to the extent that it is not "different in kind from copyright infringement."

[151] Software Subcommittee Report , B21-B22.

[152] CONTU Final Report, pp 42-45.

[153] The Final Report noted many advantages of copyright over trade secrets, including the non-uniform nature of trade secret law, the requirement for entering a contractual relationship, the higher cost of obtaining, maintaining and enforcing protection, and the possibility of eternal protection coupled with the high risk of unexpected immediate loss at any time. CONTU Final Report pp 45-47. Note that many of these advantages accrue to society as a whole, not to the individual proprietors who will be making the selection of intellectual property regimes.

Congress to enforce that preference, assuming that courts would immediately perceive trade secrecy as incompatible with a mass-market software business and that those businesses would voluntarily give up their trade secrets and opt for copyright protection instead.

```
/*

 *      Copyright (c) 1984, 1985 AT&T
 *      All Rights Reserved

 *      THIS IS UNPUBLISHED PROPRIETARY SOURCE
 *      CODE OF AT&T.
 *      The copyright notice above does not
 *      evidence any actual or intended
 *      publication of such source code.

 */
```

— AT&T UNIX System V release 2 (1985)

8. Software Law Part 2

The uncertainties regarding trade secret and copyright law protection over computer software were so great during the 1970's that a variety of mutually contradictory strategies could have been justified. The following discussion assumes a software maker that would have liked to have both copyright and trade secret protection on their work. I will also assume that, like AT&T in the case of Unix, the software proprietor would like to encourage user contributions by distributing the source code along with the software product.[154]

8.1. Notice Plus Registration

First of all, you could have shipped the code with copyright notices and deposited copies with the Library of Congress. This is what computer manufacturer Burroughs Corporation did.[155] Frank H. Cullen, the manager of Patent Headquarters at Burroughs, believed that registration and distribution of a software work was not necessarily a "divestive publication" resulting in the loss of trade secret protection.[156] Burroughs did in fact sell their software with copyright notices affixed. They deposited their works for public display at the copyright office, but then entered into restrictive trade secret licenses with their customers. Cullen believed that any trade secrets in his company's software were adequately obscured by the fact that any one program represents an "absolute mass of material, and it's so great that it requires some automatic memory to memorize it." The software was so incomprehensible that even reading its source code would not extract its secrets — nothing short of wholesale copying could do so. Its value lay not in the techniques and algorithms embodied, but rather in the fact that, as a whole, it could do useful work inside the computer. The trade secret was not what the code taught, but what it did. In fact Cullen's admission that only wholesale copying would compromise any "secrets" in his software amounts to an admission that trade secret law was being invoked solely to prevent such wholesale copying, which is contradictory to the express trade secret preemption of the copyright act.[157] Besides, this strategy could never work for a software product like Unix. Version 6, released in May 1975, was only 11,000 lines of code (less than 170 pages), and it was eminently comprehensible.

Mr. Cullen did note an exception to the deposit requirement. In a few cases, programs were so massive that they would constitute "a stack of papers six or eight feet high." Burroughs had a

[154] This was not atypical during that time. IBM, for example, shipped source code with all of their software.

[155] By January 1, 1977, Burroughs and IBM represented the vast majority of software copyright registrations. Out of a total of 1205 programs registered, IBM and Burroughs accounted for 971. CONTU Final Report, 85.

[156] Frank Cullen's testimony appeared in CONTU Meeting 16, pp 74-110.

[157] The 1976 act, through 17 U.S.C. §301, preempts "all legal and equitable rights that are equivalent to any of the exclusive rights within the general scope of copyright as specified by §106."

"special arrangement" whereby they could deposit only the first 10 or fifteen pages of each major portion of the program, thereby depositing only a small fraction of the code.[158] This argument also seems somewhat disingenuous. Burroughs could very well have had a "special arrangement" allowing them to deposit a tape, micro-fiche, or some other high density storage medium which contained the entire source code. Today, the copyright office permits anyone to deposit a limited amount of code with the justification of allowing the owner to maintain trade secret rights. It appears that, even before the 1976 act went into effect, the copyright office was pulling away from the disclosure policy written into the registration and deposit provisions. In any event, this wouldn't really have worked for Unix either. In the case of Version 6, the small size of the system provided no impediment to deposit.

An alternate strategy was pursued by IBM, a strong proponent of software copyright. IBM was well known for shipping source code with all of its software products. In fact, their software was almost always free, at least until federal antitrust authorities asked them to stop that practice.[159] The practice of sharing source code suggests that IBM was either less concerned about trade secret protection, or they had learned the countervailing value of encouraging end-user support. On the other hand, it is reasonable to assume that their software products were at least as incomprehensible as those of Burroughs. After all, the creation of IBM's OS/360 operating system had taken 5000 man years,[160] compared to 10 man years for Unix version 6.[161] Of course, Unix in the 1970's always shipped with source code. It was a research project, and source code was largely what people wanted it for.

There were a number of serious problems with this strategy, and as stated in the previous chapter, they were not resolved by CONTU. For example, an article by Peter A. Luccarelli, Jr. in the 1981/1982 volume of the Computer Law Journal argued convincingly that state courts ought to dismiss trade secret actions when the subject matter is copyrightable and bears a copyright notice. The owner should be forced register the work, deposit a copy with the Copyright Office, and proceed with a copyright action in federal court.[162]

8.2. Notice Without Registration

An alternate strategy would have been to ship source code with notices, but without registering. This strategy has many of the weaknesses of the previous strategy, plus one additional weakness. The 1976 Copyright Act grants a five year grace period from the time of "publication" to the time when registration and deposit must occur. The meaning of publication in a software context was very unclear at the time.[163] The testimony of Frank Cullen before CONTU exemplified this problem. He (and Burroughs) postulated that you could "publish" to the extent that copyright notice is required, but not to the extent that trade secret protection would be lost.[164] This strategy is a tightrope walk between the two definitions of "publication."

8.3. No Notice

The final option would have been to ship the code without any notices at all and accept that you may be abandoning your copyright and relying on trade secret alone. The Copyright Act of 1976 provides two related escape hatches in this case. Assuming that the initial distribution turned out to be a publication, the proprietor would still have a five year grace period to register.[165] They may lose their trade secrets, but they at least can now claim copyright

[158] Id. at 107.

[159] See 60 Texas L. Rev. 587, 599-604.

[160] Brooks, The Mythical Man Month.

[161] John Lions, Commentary on Unix 6th Edition, Chapter 1.

[162] Peter A. Luccarelli, Jr., The Supremacy of Federal Copyright Law over State Trade Secret Law for Copyrightable Computer Programs Marked with a Copyright Notice, 3 Computer Law Journal 19 at 52. Need to update this. What happened in the courts?

[163] It's still unclear?

[164] CONTU Meeting 16, 85.

[165] Cite? Would they have to claim that they neglected notice by accident?

protection. Note that the publication issue is more favorable to the proprietor here. The defendant could no longer claim that the copyright notice was evidence of publication or intent to publish.[166] The other escape hatch occurs if the initial distribution of software was not a publication at all. A proprietor could wait until infringement before registering for copyright and depositing the work. They would stand the risk of losing their trade secret through registration at this point, but they were of course already facing the loss of the secret through the infringing activity. Copyright at this point may serve to put the genie at least partially back in the bottle.

8.4. The Software Industry in the mid 1970's

CONTU's job was made especially difficult because of the rapidly changing landscape of the computer industry. Advances in semiconductor technology led to the development of the microprocessor, which in turn allowed for the creation of extremely low cost computers. An enthusiastic hobbyist community soon formed around formed around these inexpensive but terribly rudimentary machines. The first commercial microcomputer, the MITS Altair 8800, appeared in 1975. It sported an Intel 8080 microprocessor and cost $397. Over the next decade, the extraordinary advances in computer hardware technology would reshape the software industry. Brooks theorized that because hardware prices fall much faster than software productivity increases, it becomes necessary for a software mass market to develop.[167] If software productivity remains relatively constant, that means that the cost of a custom designed application will not fall significantly over time. Therefore, software costs would soon overwhelm hardware costs in the absence of a mass market.

Even in the late 1970's, with personal computers barely on the horizon and virtually ignored by major computer manufacturers,[168] the computer industry itself was still coming to terms with the proper way to produce and market software. In 1977, Philip Dorn said that— "[Software is] not something that we care to give away for free. We did for many years, I might add. We gave away programs. We didn't realize they had any value, and many of them did not. We are just beginning to learn how to do our business."[169] Whereas early computer manufacturers tended to provide software for free with the purchase of the hardware, by the late 70's people were beginning to see software as an independently viable business.

In 1975, two teenage Harvard students, Bill Gates and Paul Allen, went into business selling commercial system software for the Altair 8800.[170] They were hired by MITS to implement the BASIC[171] programming language. A copy of Altair BASIC on paper tape cost $50. However, Gates and Allen soon learned that the market for microcomputer software was very different from the market for minicomputer software. Microcomputer owners at the time were hobbyists, not business users. They were accustomed to doing things themselves and sharing their work, rather than paying someone else for proprietary software. About 90% of the users of Altair BASIC never paid for it. They copied it from a friend. On February 3, 1976, Bill Gates wrote an open letter to computer hobbyists. He argued that good software would never exist in the microcomputer market if hobbyists do not stop "stealing" the software they use.[172]

As a direct result of this conflict, we have perhaps the first concerted volunteer effort to clone a piece of commercial software and give it away for free. Around this time, the People's Computer Company (PCC), a community organization in Berkeley California providing

[166] See, e.g. comments made by Susan Nycum before CONTU. "Now certain customers of software licensors — in particular agencies in the Federal government — have argued that these two concepts of protection are inimical, and if a copyright notice is affixed to the product, publication will be presumed and that any non-disclosure notice or covenant within the license not to disclose, et. cetera., may safely be ignored." Susan Nycum testifying before CONTU, Meeting 16 (1977) pp 13-14.

[167] Brooks, No Silver Bullet.

[168] The IBM PC was introduced in 1981.

[169] Contu Meeting 16 at 24.

[170] Levy, Hackers, p 225.

[171] Beginners All Purpose Symbolic Instruction Code.

[172] William Henry Gates III, An Open Letter to Hobbyists, http://www.blinkenlights.com/classiccmp/gateswhine.html.

community access to computers, published an article describing a "participatory project" to implement a "Tiny BASIC" interpreter. Soon they were so deluged with source code and bug reports that they spun off a new magazine, the Tiny BASIC Journal, devoted to that discussion. The new newsletter soon changed its name to the Dr. Dobbs Journal and broadened its charter to the discussion of "free and very inexpensive software." The first issue described one solution to the problems posed by Bill Gates: make software so cheap that it is easier to buy than to copy. The community response to Altair BASIC aptly demonstrates the tension between property rights and the drive towards free software alluded to by Dennis Ritchie in the quote in the introduction. It represents the same type of response that can occur when a network of computer programmers work to circumvent an overreaching property right. This is the same type of struggle that occurred a decade later on a much larger scale with Unix.

FIXME: compare to census revolt in 1910 — Powers vs. Hollerith.

> So it would seem to me that if I market a program in 1978 under let's say, the best trade secret agreement I can dream up, and I don't have a copyright notice in it, and let's say it's intentionally not there, so we don't drag in the question of the five year provisions. I may well have published and kissed my copyright away.
> — Joseph Taphorn[173]

9. Contamination

9.1. Standardization

Through the second half of the 1980's, tensions in the commercial and non-commercial Unix worlds were rising. Despite its advantages, the Unix licensing model, which permitted users to create and share new versions, also had a serious disadvantage. The existence of a multitude of commercial variants marketed by competing companies gave rise to fragmentation. Standardization was required to combat the natural tendency of these systems to diverge. The process of standardizing a new feature is itself necessarily a compromise between the sharing and concealing of intellectual property. The Unix standardization model was primarily to agree on a behavioral description of the feature while keeping its implementation proprietary and secret. This method succeeded in some situations, but overall proved insufficient to prevent the fragmentation of the Unix market. Ultimately, many of the most succesful attempts at standardization were instead accomplished through the sharing of non-proprietary source code.

An early example is the standardization of the C programming language. Since the early 1970's, Unix was written in a higher level language called 'C,' the successor to a language called 'B.'[174] Because Unix and all of its utilities were written in C, it was beneficial to AT&T to have a large number of programmers who knew the language. The population of C programmers constitutes an economic network, the growth of which enhances the value of Unix.[175] C is today one of the dominant computer languages, despite the fact that its development has been plagued by fragmentation fueled by property rights.

Steve Johnson at Bell Labs was involved with the implementation of compilers for Unix, and created one of the early C compilers. The part of a compiler called a "lexical analyzer" interprets the formal grammatical structure of the text in the source code of a program. It doesn't actually do any of the hard work of compilation, but it's a prerequisite for interoperability. Steve Johnson said

> I made a strong effort to release the C grammar and lexical analyzer into the public domain, in an attempt to standardize the language. The impulse was the same as the open source movement today — that if the code was shared, the program would be highly portable. And we wanted C to be a portable language.
>
> I failed in this attempt. The lawyers could not bring themselves to "give away" code, even if it was in AT&T's strategic interest to do so. As a result, many PC C compilers were written based on the incomplete and ambiguous specification in the Kernighan and Ritchie book. The resulting incompatibilities irritated the industry for years, and led to the need for an ANSI C standards effort that took several more years.[176]

C was based on B, which was a simplified form of BCPL (an acronym for "Basic CPL"). CPL is the "Combined Programming Language," a derivative of ALGOL 60, which in turn was a derivative of ALGOL 58. C in turn gave rise to C++, which was first developed from 1979 to 1983. Both Java (from Sun) and C# (from Microsoft) are syntactic decendents of C++, released

[173] Chairman of the Software Committee of the Information Industry Association, testifying before CONTU Meeting 16 on September 15, 1977.

[174] There was no 'A.' IBM's APL (an acronym for "A Programming Language") is unrelated.

[175] See, e.g. FIXME anything on network economics?

[176] Steve Johnson, email on March 31, 2001.

in 1995 and 2000 respectively.[177]

9.2. Fragmentation

Each new Unix version from AT&T potentially came with a new set of licensing constraints. As each new innovation from AT&T was introduced, Berkeley had a choice — license AT&T's code, or independently develop or acquire compatible code. Ultimately, due in part to increasingly stringent prices and terms, Berkeley never licensed AT&T code for incorporation into BSD after Unix 32V. Therefore, while many AT&T innovations were duplicated in BSD, but many others were left out or implemented in incompatible ways. On the other hand, a large number of significant innovations were made neither by AT&T or Berkeley. The survival of these innovations had a little bit to do with technical merit and a lot to do with the way they were shared.[178]

Take as a simple example the introduction of a new C library function called 'getopt' in AT&T Unix System III. This function was intended to provide a uniform mechanism for parsing parameters in command line tools, and it was made available as part of the standard library for all subsequent versions of AT&T Unix. Software developers got two benefits out of this — they saved the duplicated effort that would have come from re-implimenting such a common function, and they got some of the advantages of standardization for the user interfaces of their programs. These are two of the classic arguments for having an operating system in the first place. The problem was that Berkeley never licensed System III. Therefore, programmers wishing to use getopt had two options — they could write their software so that it would only run on official AT&T Unix systems and ignore the many users on BSD systems, or they could ship their own version of getopt with their code. A third option, legally prohibited, would have been for software developers to ship AT&T's source code with their program.[179]

BSD was too important to ignore. The getopt function was duplicated and put into the public domain only months after its introduction by AT&T. In October, 1981, Henry Spencer cloned getopt based on its description in AT&T's Unix manual. He shared his code with the Unix community in January.[180] In announcing his contribution, he said

> The following is the source and manual page for a getopt() routine written locally to match the description in a copy of the Unix 3.0 manual that I got to see once. Behavior is believed identical to the Bell one, but this is NOT Bell code and carries no nondisclosure restrictions. Use it on any system you wish.

The code was 60 lines long. The documentation he wrote was about twice that long.

This is not the end of the story, however. It turns out that AT&T's version had a number of bugs in it, or perhaps they were simply undocumented but intended 'features.' It doesn't really matter, because a number of programs relied on the behavior of AT&T's version and broke when used with Henry Spencer's. In 1984, in an attempt to conform the free getopt to the behavior of the proprietary one, Keith Bostic at Berkeley wrote his own version, based on Henry Spencer's but different enough to constitute a substantial rewrite. He posted it to the Usenet as a public domain work.[181]

---

[177] Notice that a patent issued in the year ALGOL 58 was introduced (1958) would not have expired until 1975, three years after the creation of C. A patent issued in 1979 on C++ would not have expired until 1996, a year after the introduction of Java. Those patents might have prevented developments whose true value could not possibly have been forseen by their creators. For example, when the C programming language was devised, Unix was not far out of its "Space Travel" stage. Ken Thompson was not in a position to license a hypothetical Algol 58 patent at that time, and would likely have had to start from scratch.

[178] See the discussion of the X Window System, ?? infra, the Network File System (NFS), ?? infra.

[179] Shipping a pre-compiled version was also possible, but not really feasible given the wide variety of Unix systems available and the lack of binary compatability between them.

[180] Usenet post to net.sources newsgroup, dated January 11, 1982. Spencer's manual page, included with his source code distribution, bears a date of October 28, 1981.

[181] FIXME: When was it posted? There is a repost from December 1984. What was the first version of BSD to include this code? Can find out from CSRG sccs files.

In a classic example of the pace of legal reasoning not keeping up with the software development community, AT&T entered the fray towards the end of 1985.[182] At the 1985 Uniforum conference in Dallas, Texas, they distributed copies of the "official" getopt routine. The code was identical to the code included in the latest and greatest AT&T Unix, except that the copyright and trade secret notice had been removed. The code was posted to Usenet, with some trepidation, soon thereafter.[183] Despite public assurances that the code was no longer secret or copyrighted, the dark shadow of AT&T's notoriously aggresive legal department still hung over the Unix community. Berkeley did not take the official version into the C library for BSD. They continued to use Keith Bostic's version of Henry Spencer's code. The Free Software Foundation standardized on their own greatly enhanced version based on Henry Spencer's original. The version of getopt included in the latest version of the GNU C Library,[184] at over 1000 lines, is obviously substantially different from its 20 year old ancestor, but its lineage is clearly identifiable.

Despite Keith Bostic's rewrite, and despite AT&T's donation, programmers were still not able to rely on the operating system to provide this function in a reliable way. Many programs, in the interest of maintaining portability, still shipped with their own version, only now they had three to choose from. Many other programs around this time were distributed with instructions that they must be compiled one way on AT&T Unix and a different way on BSD. This is not a great burden when there are only a few differences. However, remember that getopt is one of thousands of standard operating system routines, and as the number of divergent platforms increased, so did the number of differences between them. A modern solution to this problem is the GNU Autoconf tool, which automatically scans the system before compilation in order to discover the presence or absence of a wide variety of features. The getopt example illustrates the dynamics at work in any effort to standardize an interface when implementations are kept proprietary. The same forces played out over many other issues, large and small. Berkeley was able to hold the Unix community together for many years by serving as a major source of non-proprietary innovation.[185]

9.3. Decontamination

As time passed, the pressure on Berkeley to upgrade BSD by licensing a newer version of AT&T Unix was mounting.[186] In 1988 Berkeley was asking the Free Software Foundation to incorporate a number free tools created for the GNU project into BSD. However, not only was the price of a source code license rapidly rising, their own contributions had begun to far outweigh the original code in Unix 32V. Already by 1983, the source code had grown to over 700,000 lines in 4.2 of BSD from 170,000 lines in Unix 32V. Much of this code was completely original to Berkeley, sharing no common origin with AT&T Unix at all.

In particular, because of DARPA's patronage, the BSD networking code had become very important to a wide variety of users on the Internet, some of them not even running Unix and therefore not eligible to receive copies of all of BSD. This became very significant during 1987, when problems with the TCP protocol became evident. The exponential growth of the Internet was outpacing the government's ability to deploy new infrastructure, and the protocols, originally tested on the short, fast links of a local area network, proved to be poorly matched to long, slow transcontinental links. A number of people, notably Van Jacobson and Mike Karels from Berkeley, demonstrated a modification to the transmission algorithm that could avoid congestion problems and significantly improve the speed of the network. His new protocol was implemented in a modified BSD system. The July 1987 issue of the Internet Monthly Report summed up the situation:

---

[182] Date??

[183] First to mod.std.unix on [date??], then to mod.sources by John Quarterman on November 2, 1985.

[184] Version 2.2.4, current as of November 2001.

[185] FIXME — include story about standardization of the C library?

[186] See e.g. GNU's Bulletin, vol. 1 no. 5 (11 June 1988), available at http://www.gnu.org/bulletins/bull5.html. FIXME: Is this license available?

> We believe that the work which Van and Mike Karels have been doing represents a very important step forward in TCP implementations, and its wide adoption in the Internet would improve service for all. The task force discussed the mechanics of expediting the incorporation of this new TCP into vendor products which currently use 4.2BSD or 4.3BSD as a base. The essential idea is for Berkeley to freeze on an updated TCP, at an appropriate time which we hope is not too far away, and to make the frozen code available in the public domain. Van is pursuing this strategy.[187]

While congestion on the Internet Backbone was a significant incentive to create a license free Unix, it was not the only one. Richard Stallman had been asking Berkeley for years to free whatever code they could. Even the availability of small utilities would have been a big help for his GNU project. In order to accomodate the process of separating free from "contaminated" code, Berkeley began a practice of marking source code files based on their licensing status. Files were marked as either known to be original to Berkeley, known to be derived from AT&T, or still uncertain.

At the January 1988 USENIX meeting, John Gilmore proposed a project to solve the problem of AT&T's licensing strategy. The project was called "project sift," and its volunteer participants would search through the BSD source code looking for uncontaminated code. While the GNU project was attempting to recreate Unix tools from scratch, project sift pursued the complementary strategy of identifying uncontaminated code from Berkeley Unix. The project was active for three years and worked in close cooperation with Berkeley. Interestingly, for at least a year, AT&T's licensing organization was helping as well. AT&T reviewed source code sent to them and made determinations as to whether or not it infringed any of their intellectual property rights. This practice apparently stopped in the middle of 1988.

9.4. Net/1

The first release of free Unix related software from Berkeley came in 1988. It was called the BSD Networking Release. The April 1988 issue of the Internet Monthly Report contained the following note about its release.

> The BSD networking code, newly freed from distribution restrictions, has been officially announced by Mike Karels. This code implements the TCP performance improvements that Van Jacobson and Mike have developed. All vendors whose products are based on 4.2/4.3BSD are urged to obtain this code and install it.[188]

The release was not a full operating system, but it was still a substantial amount of very useful free software, and it presaged the eventual "liberation" of BSD. It was released on basically the same terms under which AT&T acquired a license to 4.3BSD. However, it had the advantage of not being dependent on a Unix license. Therefore, licensees of the networking release could make copies and share them for free. The code was soon available for free download from a number of sites on the Internet. Nevertheless, several hundred organizations paid the $1,000 distribution fee to Berkeley for an "official" copy.

9.5. Unix Wars

Several years would pass before the next release of code from Berkeley. In the mean time, BSD faced significant challenges in the marketplace. In 1987, AT&T purchased a 20% stake in Sun Microsystems, which had been one of the main commercial proponents of BSD since Bill Joy went to work there in 1982.[189] Sun announced that it would be abandoning its BSD derived operating system in favor of a future version of AT&T System V. The alliance between Sun and AT&T, the two most powerful players in the Unix market, was seen as a grave threat to the rest

[187] Internet Monthly Report, July 1987, available at http://sunsite.utk.edu/ftp/pub/internet-monthly-reports/imr8707.txt. FIXME — Was Cisco using BSD at this time? I should mention the other commercial vendors that were waiting for the release.

[188] Internet Monthly Report, April 1988, available at http://sunsite.utk.edu/ftp/pub/internet-monthly-reports/imr8804.txt

[189] FIXME: check date

of the industry. In response, an organization called the Open Software Foundation (OSF) was formed by a number of major users of BSD, including IBM, DEC, and Hewlett Packard. They announced the aim of eventually producing an AT&T license free derivative of Unix.[190] AT&T, along with supporters of Unix System V, formed a rival group called Unix International. OSF produced a very popular graphical user interface library (Motif) but eventually abandoned their plans for an AT&T free operating system.

9.6. Net/2

At Berkeley, Keith Bostic began to spearhead an effort to create an expanded freely redistributable release.[191] Bostic solicited aid from volunteers on the Internet. Many people contributed some of the small utilities that are part of the Unix system. Bostic, Mike Karels, and Kirk McKusick (all from Berkeley) removed all of the AT&T derived code from the BSD kernel and rewrote the parts that could be done easily. When they were finished, they found that they had a virtually complete system. Only 6 files were missing that could not be easily recreated. Rather than take the time to get a new licensing agreement approved by the University lawyers, they decided to release the code they had as an update to the Berkeley Networking Release. They called it Net/2 and began shipping it in June, 1991.[192]

Although Net/2 was not a full release, two groups soon formed to complete the task. One was led by Bill and Lynne Jolitz, who ported the code to the Intel 386 microprocessor and rewrote the 6 missing files. They released the first version, 386BSD 0.1, on the evening of July 15, 1992.[193] By this time, the popularity of Intel 386 based computers among college students, the lack of a modern commercial multi-tasking operating system for personal computers, and the rapid rise in Internet connectivity on college campuses all conspired to create a massive demand for free Unix based systems. The Jolitz's later told Salon Magazine that 386BSD 0.1 was downloaded 250,000 times.[194] The release of 386BSD created some competition for the Linux operating system, which was first announced almost a year earlier, in August 1991, but was in an extremely skeletal form at that time. 386BSD evolved directly into NetBSD, FreeBSD, and OpenBSD, the three main varieties of freely redistributable BSD systems.

9.7. BSDI

At the same time, a company called Berkeley Software Design, Inc. (BSDI) was formed to commercialize the Net/2 release. They aimed to provide a proprietary distribution, called BSD/386, which would come with source code, as well as commercial support. Their system sold for $995, which was billed as a 99% discount to the price of an AT&T System V source code license. Shortly after their sales campaign began, they were sued by Unix System Laboratories (USL), the division of AT&T responsable at that time for licensing Unix.

The first complaint alleged 1) trademark infringement for BSDI's use of the telephone number 1-800-ITS-UNIX; 2) a Lanham Act violation for false descriptions of origin, source, sponsorship or authorization; 3) dilution of the Unix trademark; and 4) unfair competition and deceptive trade practices under common law. Noticeably absent was an accusation of copyright infringement or trade secret misappropriation. Part of the reason for this is that the copyrights for Unix 32V, from which the Net/2 release was ultimately descended, was not registered until May 15, 1992, three weeks after the complaint was filed. An amended complaint was filed in July, joining the Regents of the University of California as defendants, and adding the copyright and trade secret claims. They sought an injunction against the distribution of Net/2 and BSD/386.

[190] However, OSF software was not intended to be free in any sense other than the absence of AT&T control.

[191] See generally Kirk McKusick, Twenty Years of Berkeley Unix, From AT&T-Owned to Freely Redistributable, Open Sources: Voices from the Open Source Revolution.

[192] FIXME

[193] Usenet post by David J. Hughes, posted to comp.unix.bsd on July 16, 1992. FIXME: but Bill Jolitz worked (at least briefly) at BSDI?

[194] The Unknown Hackers, Salon Magazine, May 17, 2000.

Although they must have known about it, USL did not seek an injunction against Bill and Lynne Jolitz for 386BSD. In fact, the Jolitz's continued active and public development, releasing version 1.0 in December of 1993.[195]

A ruling on March 30, 1993 by Judge Debevois of the District of New Jersey would prove to be very damaging to USL's case.[196] Debevois determined that the distribution of 32V was not a limited publication because the criteria under which distribution was made were "general" rather than "limited." AT&T distributed the 32V source code to anyone who "wanted it, could pay for it, reasonably needed it, and would protect it from redistribution."[197]

A second major setback for USL occurred when Judge Debevois found the University, and therefore the regents, largely immune from suit on the grounds of state sovereign immunity. The University had filed a countersuit in California state court alleging a breach of the "proper credit" clause of the 1986 license agreement. Throughout the 1980's, AT&T and USL had been distributing Unix derived from 4.3BSD without honoring the credit provisions. Also during 1993, AT&T sold Unix System Laboratories to Novell Corporation, the maker of Netware, a popular network operating system. Ultimately, Novell proved willing to settle the case, and on February 4, 1994, an agreement was reached. As a concession, BSDI and the University of California agreed to withdraw the Net/2 release and create a new release called 4.4 BSD(Lite) after the removal of three disputed files.[198]

9.8. Postscript

The issue in the case that created the most widespread fear was the possibility that a programmer, merely by looking at or studying the the Unix source code, would somehow become "mentally contaminated." It was not exactly clear if USL were alleging that a mentally contaminated programmer could not safely work on a competing operating system without raising suspicions. There is reason to believe that the lawsuit drove a number of developers from 386BSD to the budding (and written from scratch) Linux system for fear that their work on the Net/2 derived project would either contaminate them, or simply be wasted effort in the event that 386BSD is withdrawn from distribution.

The attempt to clone Unix was inevitable when AT&T began to price their contributors out of the market. The barrier to entry to run Unix shrank dramatically with the introduction of the Intel 386 and other fast microprocessors. Therefore, many more people were suddenly in a position to benefit from access to its source code. Ironically, AT&T was simultaneously raising, not lowering its prices. Their policies hastened the arrival of a free clone. In 1992, there were few alternatives for someone seeking a high quality but inexpensive operating system for a personal computer. What offerings did exist would soon be obsoleted, for the needs of the average "hobbyist," by Linux and the free BSD variants.

Today, the Unix market continues to be robust and competitive. However, it has consolidated significantly, and successful standardization efforts have rendered incompatibilities much less of a concern. Furthermore, the incredible commercial success of Linux has arguably provided something like a reference implementation. Many vendors now advertise the ability to run Linux binary programs. AT&T used to wield its trademark rights over the Unix name as a strong strateic tool. Unix was originally defined as a system derived from AT&T's proprietary source code for which AT&T had given its blessing. Today, the trademark is owned by The Open Group "in trust for the industry" , and it identifies an industry standard to which any computer operating system, even Microsoft Windows, can conform.

---

[195] Some developers working on Net/2 derived systems were contacted in 1993 (?? FIXME), but no further actions were taken against them.

[196] Unix System Laboratories v. Berkeley Software Design, Inc., 1993 U.S. Dist. LEXIS 19503.

[197] Id. at 13.

[198] Unix System Laboratories was later sold to the Santa Cruz Operation (SCO), the company that had marketed Microsoft Xenix and later its own version of AT&T System V Unix. In 2001, SCO was sold to Caldera, a company founded in 1996 to market Linux.

See http://www.unix.org/what_is_unix.html, visited 1 December 2003.

Although AT&T's contributions to computer science through the Unix system have been extraordinarily far reaching, there is certainly a good amount of pure chance involved in it's history. It certainly raises questions as to whether creation of the acorn entitles one to rights over the resultant oak (or forest). Unfortunately, the lack of clarity over the scope of intellectual property protection on computer software creates serious doubts and uncertainty in the minds of software developers and marketers. The fact that there appears to be a vast discrepancy between software industry practice and judicial application of intellectual property law is especially troubling. Attempts such as UCITA to codify the contract law of information transactions may, by hewing to industry licensing practice, in fact pose serious dangers of interfering with actual practices of software developers that have proven to be successful drivers of innovation in the past.

Table of Contents