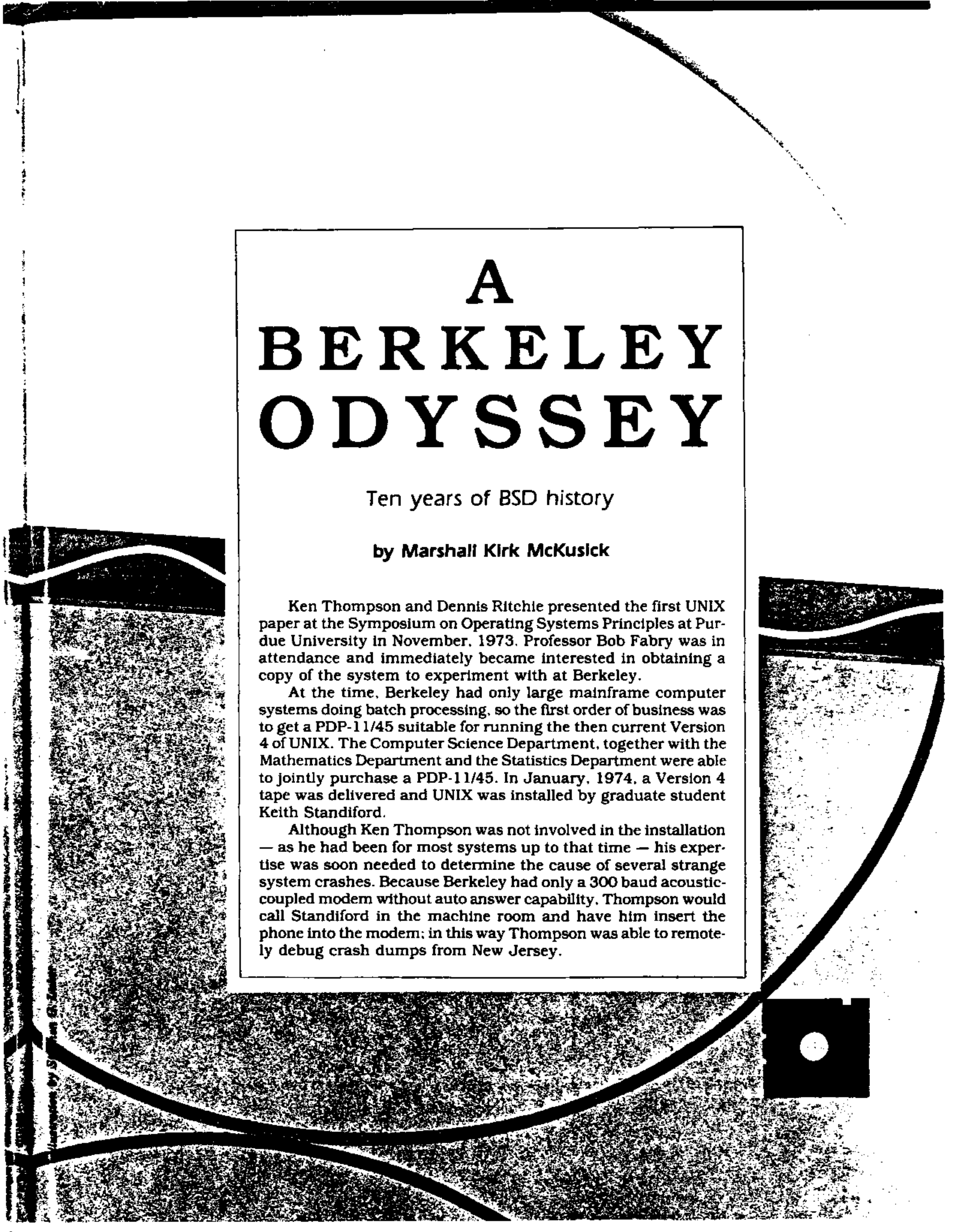


A BERKELEY ODYSSEY

Ten years of BSD history

by Marshall Kirk McKusick

Ken Thompson and Dennis Ritchie presented the first UNIX paper at the Symposium on Operating Systems Principles at Purdue University in November, 1973. Professor Bob Fabry was in attendance and immediately became interested in obtaining a copy of the system to experiment with at Berkeley.

At the time, Berkeley had only large mainframe computer systems doing batch processing, so the first order of business was to get a PDP-11/45 suitable for running the then current Version 4 of UNIX. The Computer Science Department, together with the Mathematics Department and the Statistics Department were able to jointly purchase a PDP-11/45. In January, 1974, a Version 4 tape was delivered and UNIX was installed by graduate student Keith Standiford.

Although Ken Thompson was not involved in the installation — as he had been for most systems up to that time — his expertise was soon needed to determine the cause of several strange system crashes. Because Berkeley had only a 300 baud acoustic-coupled modem without auto answer capability, Thompson would call Standiford in the machine room and have him insert the phone into the modem; in this way Thompson was able to remotely debug crash dumps from New Jersey.

Many of the crashes were caused by the disk controller's inability to reliably do overlapped seeks, contrary to the documentation. Berkeley's 11/45 was among the first systems that Thompson had encountered that had two disks on the same controller! Thompson's remote debugging was the first example of the cooperation that sprang up between Berkeley and Bell Labs. The willingness of the researchers at the Labs to share their work with Berkeley was instrumental in the rapid improvement of the software available at Berkeley.

Though UNIX was soon reliably up and running, the coalition of Computer Science, Mathematics, and Statistics began to run into problems; Math and Statistics wanted to run DEC's RSTS system. After much debate, a compromise was reached in which each department would get an eight-hour shift; UNIX would run for eight hours followed by 16 hours of RSTS. To promote fairness, the time slices were rotated each day. Thus UNIX ran 8 am to 4 pm one day, 4 pm to midnight the next day, and midnight to 8 am the third day. Despite the bizarre schedule, students taking the Operating Systems course preferred to do their projects on UNIX rather than on the batch machine.

Professors Eugene Wong and Michael Stonebraker were both stymied by the confinements of the batch environment, so their Ingres database project was among the first groups to move from the batch machines to the interactive environment provided by UNIX. They quickly found the shortage of machine time and the odd hours on the 11/45 intolerable, so in the Spring of 1974, they purchased an 11/40 running the newly available Version 5. With their first distribution of Ingres in the Fall of 1974, the Ingres project became the first group in the Computer Science department to distribute its software. Several hundred Ingres tapes were shipped over the next six years, helping to establish Berkeley's reputation for designing and building real systems.

Even with the departure of the Ingres project from the 11/45, there was still insufficient time available for the remaining students. To alleviate the shortage, Professors Michael Stonebraker and Bob Fabry set out in June, 1974, to get two instructional 11/45s for the Computer Science department's own use. Early in 1975, the money was obtained. At nearly the same time DEC announced the 11/70, a machine that appeared to be much superior to the 11/45. Money for the two 11/45s was pooled to buy a single 11/70 that arrived in the Fall of 1975. Coincident with the arrival of the 11/70, Ken Thompson decided to take a one-year sabbatical as a visiting professor at his alma mater. Thompson, together with Jeff Schriebman and Bob Kridle, brought up the latest UNIX, Version 6, on the newly installed 11/70.

Arriving in the Fall of 1975 were two unnoticed graduate students, Bill Joy and Chuck Haley.

Also arriving in the Fall of 1975, were two unnoticed graduate students, Bill Joy and Chuck Haley; they both took an immediate interest in the new system. Initially they began working on a Pascal system that Thompson had hacked together while hanging around the 11/70 machine room. They expanded and improved the interpreter system to the point that it became the programming system of choice for students because of its excellent error recovery scheme and fast compile and execute time.

With the replacement of Model 33 teletypes by ADM-3 screen terminals, Joy and Haley began to feel stymied by the constraints of the **ed** editor. Working from an editor named **em** that they had obtained from Professor George Coulouris at Queen Mary's College in London, they worked to produce the line-at-a-time editor **ex**.

With Ken Thompson's departure at the end of the Summer of 1976, Joy and Haley begin to take an interest in exploring the internals of the UNIX kernel. Under Schriebman's watchful eye, they first installed the fixes and improvements provided on the "fifty changes" tape from Bell Labs. Having learned to maneuver through the source code, they suggested several small enhancements to streamline certain kernel bottlenecks.

FIRST DISTRIBUTION

Meanwhile, interest in the error recovery work in the Pascal compiler brought in requests for copies of the system. Early in 1977, Joy put together the "Berkeley Software Distribution". This first distribution included the Pascal system and — in an obscure subdirectory of the Pascal source — the editor **ex**. Over the next year, Joy, acting in the capacity of distribution secretary, sent out about 30 free copies of the system.

With the arrival of some

ADM-3a terminals offering screen-addressable cursors, Joy was finally able to write **vi**, bringing screen-based editing to Berkeley. He soon found himself in a quandary. As is frequently the case in universities strapped for money, old equipment is never replaced all at once. Rather than support code for optimizing the updating of several different terminals, he decided to consolidate the screen management by using a small interpreter to redraw the screen. This interpreter was driven by a description of the terminal characteristics, thus spawning the now famous **termcap**.

By mid-1978, the software distribution clearly needed to be updated. The Pascal system had been made markedly more robust through feedback from its expanding user community, and had been split into two passes so that it could be run on PDP-11/34s. The result of the update was the "Second Berkeley Software Distribution" that was quickly shortened to *2 BSD*. Along with the enhanced Pascal system, **vi** and **termcap** for several terminals was included. Once again, Bill Joy single-handedly put together distributions, answered the phone, and incorporated user feedback into the system. Over the next year, nearly 75 tapes were shipped. Though Joy moved on to other projects the following year, the 2 BSD distribution continued to expand. Today the latest version of this distribution, 2.9 BSD, is a complete system for PDP-11s.

VAX UNIX

Early in 1978, Professor Richard Fateman began looking for a machine with a larger address space that he could use to continue his work on Macsyma that had started on a PDP-10. The newly announced VAX-11/780 seemed to fulfill the requirements and was available within budget. Fateman and 13 other faculty members put together an NSF proposal that they combined with some departmental funds to purchase a VAX.

Initially the VAX ran DEC's operating system VMS, but the department had gotten used to the UNIX environment and wanted to continue using it. So, shortly after the arrival of the VAX, Fateman obtained a copy of the 32/V port of UNIX to the VAX by John Reiser and Tom London of Bell Labs.

Initially the VAX ran DEC's operating system VMS, but the department had gotten used to the UNIX environment and wanted to continue using it.

Although 32/V provided a Version 7 UNIX environment on the VAX, it did not take advantage of the virtual memory capability of the VAX hardware. Like its predecessors on the PDP-11, it was entirely a swap-based system. For the Macsyma group at Berkeley, the lack of virtual memory meant that the process address space was limited by the size of the physical memory, initially 1 MB on the new VAX.

To alleviate this problem, Fateman approached Professor Domenico Ferrari, a member of the systems faculty at Berkeley, to investigate the possibility of having his group write a virtual memory system for UNIX. Ozalp Babaoglu, one of Ferrari's students, set about to find some way of implementing a working set paging system on the VAX; his task was complicated because the VAX lacked reference bits.

As Babaoglu neared the completion of his first cut at an implementation, he approached Bill Joy for some help in understanding the intricacies of the UNIX kernel. Intrigued by Babaoglu's approach, Joy joined in helping to integrate the code into 32/V and then with the ensuing debugging.

Unfortunately, Berkeley had only a single VAX for both system development and general production use. Thus for several weeks, the tolerant user community alternately found themselves logging into 32/V and "Virtual VAX/UNIX". Often their work on the latter system would come to an abrupt halt, followed several minutes later by a 32/V login prompt. By January of 1979, most of the bugs had been worked out, and 32/V had been relegated to history.

Joy saw that the 32-bit VAX would soon obsolete the 16-bit PDP-11 and began to port the 2 BSD software to the VAX. While Peter Kessler and I ported the Pascal system, Joy ported the editors **ex** and **vi**, the C shell, and the myriad other smaller programs on 2 BSD. By the end of 1979, a complete distribution had been put together. This distribution included the virtual memory kernel, the standard 32/V utilities, and the additions from 2 BSD. In December, 1979, Joy shipped the first of nearly a hundred copies of 3 BSD, the first VAX distribution from Berkeley.

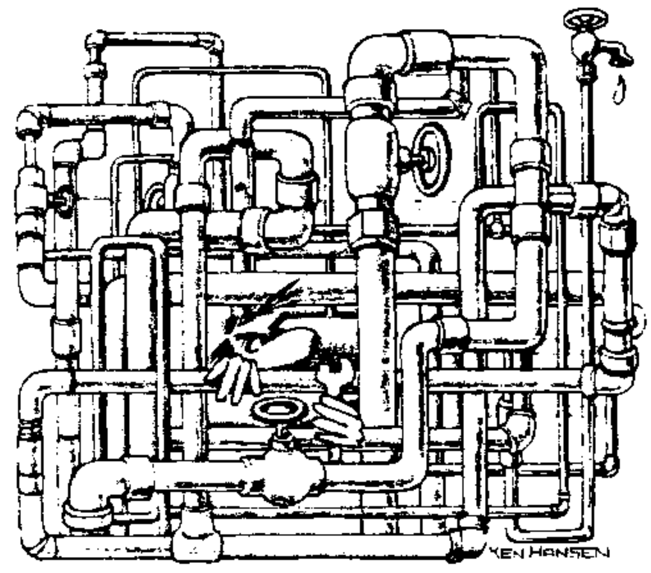
DARPA SUPPORT

Meanwhile, in the offices of the planners for the Defense Advanced Research Projects Agency, DARPA, discussions were being held that would have a major influence on the work at Berkeley. One of DARPA's early successes had been to set up a nationwide computer network to link together all its major research centers. At that time, DARPA was finding that many of the computers at these centers were reaching the end of their useful lifetime and had to be replaced. The heaviest cost of replacement was the porting of the research software to the new machines. In addition, many sites were unable to share their software because of the diversity of hardware and operating systems.

Choosing a single hardware vendor was impractical because of the widely varying computing needs of the research groups and the undesirability of depending on a single manufacturer. Thus, the planners at DARPA decided that the best solution was to unify at the operating systems level. After much discussion, UNIX was chosen as a standard because of its proven portability.

In the Fall of 1979, Bob Fabry responded to DARPA's interest in moving towards UNIX by writing a proposal suggesting that Berkeley develop an enhanced version of 3 BSD for the use of the DARPA community. Fabry took a copy of his proposal to a meeting of DARPA image processing and VLSI contractors, plus representatives from Bolt, Beranek, and Newman, the developers of the ARPAnet. There was some reservation whether Berkeley could produce a working system, but the release of 3 BSD in December, 1979, assuaged most of the doubts.

With the increasingly good reputation of the 3 BSD release to validate his claims, Bob Fabry was able to land an 18-month contract with DARPA beginning in April, 1980. This contract was to add features needed by the DARPA contractors. He immediately hired Laura Tong to handle the project administration. With the negotiations for the contract on track, Fabry turned his attention to finding a project leader to manage the software development. Fabry had assumed that since Joy had just passed his Ph.D. qualifying examination, he would rather concentrate on completing his degree than assume the software development position. But Joy had other plans. One night in early March he



Without SoftShell,™ learning to live with UNIX™ could be a real trap.

SoftShell is a convenient interface which guides you gracefully through UNIX. If you're new to UNIX, or have the task of training new users, SoftShell simplifies its complexity. If you're already a UNIX fan, you'll love SoftShell because it augments your system. You'll find yourself further exploring the great depth and versatility of UNIX. Available for UNIX System-V, Berkeley 4.2 and IBM's PC/IX, SoftShell is an invaluable tool for users at all levels of expertise. For information, contact Logical Software, Inc., 17 Mount Auburn Street, Cambridge, MA 02138, (617) 864-0137.

UNIX is a trademark of AT&T Bell Laboratories.

Logical Software Inc.

Circle No. 16 on Inquiry Card

See us at UniForum Booth # 1279

phoned Fabry at home to express interest in taking charge of the further development of UNIX. Though surprised by the offer, Fabry took little time to agree.

The project started promptly. Tong set up a distribution system that could handle a higher volume of orders than Joy's previous distributions. Fabry managed to coordinate with Bob Guffy at AT&T, and lawyers at the University of California to formally release UNIX under terms agreeable to all. Joy incorporated Jim Kulp's job control, added auto reboot, a 1K block file system, and support for the latest VAX machine, the VAX-11/750. By October, 1980, a polished distribution that also included the Pascal compiler, the Franz Lisp system, and an enhanced mail handling system was released as 4 BSD. During its nine-month lifetime, nearly 150 copies were shipped. The license arrangement was on a per institution basis rather than a per machine basis, thus the distribution ran on about 500 machines.

With the increasingly wide distribution and visibility of Berkeley UNIX, several critics began to emerge. David Kashtan at Stanford Research Institute wrote a paper describing the results of benchmarks he had run on both VMS and Berkeley UNIX. These benchmarks showed several severe performance problems with the UNIX system for the VAX. Setting his future plans aside for several months, Joy systematically began tuning up the kernel. Within weeks he had a rebuttal paper written showing that Kastan's benchmarks could be made to run as well on UNIX as they could on VMS. Rather than continue shipping 4 BSD, the tuned up system with the addition of Robert Elz's auto configuration code was released as 4.1 BSD in June, 1981. Over its two-year lifetime about 400 distributions were shipped.

4.2 BSD

With the release of 4.1 BSD, much of the furor over performance died down. DARPA was sufficiently satisfied with the results of the first contract that a new two-year contract was granted to Berkeley with funding almost five times that of the original. Half of the money went to the UNIX project, the rest to other researchers in the Computer Science department. The contract called for major work to be done on the system so the DARPA research community could better do its work.

With the release of 4.1 BSD, much of the furor over performance died down.

Based on the needs of the DARPA community, goals were set and work began to define the modifications to the system. In particular, the new system was expected to include a faster file system that would raise throughput to the speed of available disk technology, would support processes with multi-gigabyte address space requirements, would provide flexible interprocess communication facilities that would allow researchers to do work in distributed systems, and would integrate networking support so that machines running the new system could easily participate in the ARPAnet.

To assist in defining the new system, Duane Adams, Berkeley's contract monitor at DARPA, formed a group known as the "steering committee" to help guide the design work and ensure that the research community's needs were addressed. This committee met twice a year between April, 1981 and June, 1983, and included Bob Fabry, Bill Joy, and Sam Leffler of the University of California at Berkeley; Alan Nemeth and Rob Gurwitz of Bolt, Beranek, and Newman; Dennis Ritchie of Bell Laboratories; Keith Lantz of Stanford University; Rick Rashid of Carnegie-Mellon University; Bert Halstead of Massachusetts Institute of Technology; Dan Lynch of The Information Sciences Institute; Duane Adams and Bob Baker of DARPA; and Jerry Popek of the University of California at Los Angeles. Beginning in 1984, these meetings were supplanted by workshops that were expanded to include many more people.

An initial document proposing facilities to be included in the new system was circulated to the steering committee and other people outside Berkeley in July, 1981, sparking many lengthy debates. In the Summer of 1981, I became involved with the project and took on the implementation of the new file system. During the summer, Joy concentrated on implementing a prototype version of the interprocess communication facilities. In the Fall of 1981, Sam Leffler joined the project as a full-time staff member to work with Bill Joy.

When Rob Gurwitz released an early implementation of the TCP/IP protocols to Berkeley, Joy integrated it into the system and tuned its performance. During this work, it became clear to Joy and Leffler that the new system would need to provide support for more than just the DARPA standard network protocols. Thus, they redesigned the internal structuring of the software, refining the interfaces so that multiple network protocols could be used simultaneously.

With the internal restructuring completed and the TCP/IP protocols integrated with the prototype IPC facilities, several simple applications were created to provide local users access to remote resources. These programs, **rcp**, **rsh**, **rlogin**, and **rwho**, were intended to be temporary tools that would eventually be replaced by more reasonable facilities (hence the use of the distinguishing **r** prefix). This system, called 4.1a, was first distributed in April, 1982 for local use; it was never intended that it would have wide circulation, though bootleg copies of the system proliferated as sites grew impatient waiting for the 4.2 release.

The 4.1a system was obsolete long before it was frozen. However, its construction and feedback from users provided valuable information that was used to create a revised proposal for the new system called the "4.2 BSD System Manual". This document was circulated in February, 1982 and contained a concise description of the proposed user interface to the system facilities that were to be part of 4.2 BSD.

Concurrent with the 4.1a development, I completed the implementation of the new file system, and by June of 1982 had fully integrated it into the 4.1a kernel. The resulting system was called 4.1b and ran on only a few select development machines at Berkeley. Joy felt that with significant impending changes to the system, it was best to avoid even a local distribution, particularly since it required every machine's file systems to be dumped and restored to convert from 4.1a to 4.1b. Once the file system proved to be stable, Leffler proceeded to add the new file system-related system calls, while Joy worked on revising the interprocess communication facilities.

In late Spring 1982, Joy announced he was joining Sun Microsystems. Over the summer, he split his time between Sun and Berkeley, spending most of his time polishing his revisions to the interprocess communication facilities and reorganizing the UNIX kernel sources to isolate machine dependencies. Pauline Schwartz was hired to take over the distribution duties. David Mosher was hired as a technical manager to resolve problems from users in the field and to handle ordering, installation, and running of the project's hardware.

With Joy's departure, Leffler took over responsibility for completing the project. Certain deadlines had already been established and the release had been promised to the DARPA community for the Spring of 1983. Given the time constraints, the work remaining to complete the release was evaluated and priorities were set. In particular, the virtual memory enhancements and the most sophisticated parts of the interprocess communication design were relegated to low priority (and later shelved completely). Also, with the implementation more than a year old and the UNIX community's expectations heightened, it was decided an intermediate release should be put together to hold people until the final system could be completed. This system, called 4.1c, was distributed in April, 1983; many vendors used this release to prepare for ports of 4.2 to their hardware.

In June, 1983, Bob Fabry turned over administrative control of the project to Professors Domenico Ferrari and Susan Graham to begin a sabbatical free from the frantic pace of the previous four years. Leffler continued the completion of the system, implementing the new signal facilities, adding to the networking support, redoing the standalone I/O system to simplify the installation process, integrating the disk quota facilities from Robert Elz, updating all the documentation, and tracking the bugs from the 4.1c release. In August, 1983, the system was released as 4.2 BSD.

When Leffler left Berkeley for Lucasfilm following the completion of 4.2, he was replaced by Mike Karels. Karels's previous experience with the 2.9 BSD software distribution provided an ideal background for his new job. The popularity of 4.2 BSD was impressive; within 18 months, more copies of 4.2 BSD had been shipped than of all the previous Berkeley software distributions combined.

As with 4 BSD, commentary of the vociferous critics was quick in coming. Most of the complaints indicated that the system ran too slowly. The problem, not surprisingly, was that the new facilities had not been tuned and that many of the kernel data structures were not well suited to their new uses. Karels' first year on the project was spent tuning and polishing the system. An anticipated release of the polished system early in 1985 is expected to quell many of the performance complaints — much as the 4.1 BSD release quelled many of the complaints about 4 BSD.

After completing my Ph.D. in December 1984, I joined Mike Karels on the project. We hope that other researchers will continue to share their work with Berkeley. By incorporating the work of other researchers uniformly into the UNIX system at Berkeley, we can continue to offer the UNIX community a widely available state-of-the-art UNIX system.

ACKNOWLEDGEMENTS

I thank Bill Joy, Sam Leffler, Eric Allman, and Professors Bob Fabry, Richard Fateman, and Domenico Ferrari for providing the historical information presented in this article. Unfortunately, space constraints prevent me from describing the enormous amount of work that went into the user level software. I thank those people whose accomplishments have not been mentioned, but nevertheless have contributed much to the system's vitality. For the last five years, I have been supported by the National Science Foundation under grant MCS80-05144, and the Defense Advanced Research Projects Agency (DoD) under ARPA Order No. 4031 monitored by Naval Electronic System Command under Contract No. N00039-82-C-0235.

Kirk McKusick is involved in the development of Berkeley UNIX as a Research Computer Scientist for the Computer Systems Research Group at the University of California. While a graduate student, he implemented the fast file system distributed on 4.2 BSD and worked on the Berkeley Pascal system. ■

FEAR AND LOATHING ON THE UNIX TRAIL '76

Notes from the underground

by Doug Merritt
with Ken Arnold and Bob Toxen

It was 2 am and I was lying face down on the floor in Cory Hall, the EECS building on the UC Berkeley campus, waiting for Bob to finish installing our bootleg copy of the UNIX kernel. If successful, new and improved terminal drivers we had written would soon be up and running.

We were enhancing the system in the middle of the night because we had no official sanction to do the work. That didn't stop us, though, since UNIX had just freshly arrived from Bell Labs, where computer security had never been an issue. The system was now facing its first acid test — exposure to a group of intelligent, determined students — and its security provisions were failing with regularity.

I was lying face down because I'd gone without sleep for over two days, and the prone position somehow seemed the most logical under the circumstances. Bob was still working because he'd napped not 30 hours before, giving him seniority under the "Hacker-best-able-to-perform" rule of our informal order. We might have called our group "Berkeley Undergraduate Programmers for a Better UNIX", or, less euphemistically, "Frustrated Hackers for Our Own Ideas". But, in truth, our group was never named. It was simply a matter of Us versus Them.

Illustration by Erik Jorgensen

"Them" was the bureaucracy — the school administrators, the system administrators, most professors, some grad students, and even the legendary Implementors themselves at Bell Labs.

"Us" was a small, self-selected group of undergraduates with a passion for UNIX. We were interested in computers and in programming because it fascinated us; we lived for the high level of intellectual stimulation only hacking could provide. Although some in our group never expressed an interest in breaking computer security, others invested thousands of fruitful hours in stealing accounts and gaining superuser access to various UNIX systems. Our object? To read system source code.

For the most part we stayed

out of trouble, although one of our rank once had his phone records subpoenaed by the FBI — after a minor incident with a Lawrence Livermore National Laboratory computer. The Feds seemed to think our comrade had been diddling with top secret weapons research, but he actually hadn't.

Our group could probably best be characterized by its interest in creating and using powerful software, regardless of the source of the idea. Our battle cry, thanks to Ross Harvey, was "FEATURES!!!", and we took it seriously. Well, Ross may have been a little sarcastic about it, since he was referring to superfluous bells and whistles. But I used the expression as shorthand for "elegant, powerful, and flexible". We were always bugging Them to add "just one more feature" to some utility like the shell or kernel. Although They accepted some suggestions, They didn't think twice about most.

One example stands out. In early 1977, Ross, Bob, and I spent months collaborating on a new and improved shell, just before Bill Joy had started on what is now known as the C shell. The most historically significant features we designed were Ross's command to change the shell's prompt, Bob's command to **print** or **chdir** to the user's home directory, and my own **edit** feature, which allowed screen editing and re-execution of previous commands. What we did was smaller in scope than what Bill later included in the C shell, but to Us it was unarguably better than what was then available. We ceased work on our projects only when it became clear that Bill was developing what would obviously become a new standard shell. Our energies then were re-focused on persuading him to include our ideas. Some of our features ultimately were incorporated, some weren't.

We modified the kernel to support asynchronous I/O, distributed files, security traces, "realtime" interrupts for subprocess multitasking, limited screen editing, and various new system

It was simply a matter of Us versus Them.

calls. We wrote compilers, assemblers, linkers, disassemblers, database utilities, cryptographic utilities, tutorial help systems, games, and screen-oriented versions of standard utilities. User friendly utilities for new users that avoided accidental file deletion, libraries to support common operations on data structures such as lists, strings, trees, symbol tables, and libraries to perform arbitrary precision arithmetic and symbolic mathematics were other contributions. We suggested improvements to many system calls and to most utilities. We offered to fix the option flags so that the different utilities were consistent with one another.

To Us, nothing was sacred, and We saw a great deal in UNIX that could stand improvement. Much of what We implemented, or asked to be allowed to implement, is now a part of System V and 4.2 BSD; others of our innovations are still missing from all versions of UNIX. Despite these accomplishments, it seemed that whenever We asked The Powers That Be to install Our software and make it available to the rest of the system's users, We were greeted with stony silence.

Fred Brooks, in The Mythical Man-Month, describes the NIH (Not Invented Here) Syndrome, wherein a group of people will tend to ignore ideas originated outside their own social group. However, there was a stronger force at work at Berkeley, where a certain social stratification prevails that finds Nobel Laureates and department chairs ranking as demigods, professors functioning as high priests, graduate students considered as lower class citizens, and undergraduates existing only on sufferance from the higher orders — and suffered very little at that. Now, the individuals cannot be blamed for what is, in essence, an entire social order. But this is not to say that we did not hold it against them — for we most assuredly did. Unfortunately, it took time for us to appreciate the difficulties of Fighting City Hall.

This is why We were frustrated. This is why We felt We HAD to break security. Once We did, We simply added Our features to the system, whether The Powers That Be liked it or not. Needless to say, They didn't. This is why We felt like freedom fighters, noble figures even when found in the ignoble position of lying face down on the floor of Cory Hall at two in the morning.

We were on a mission that morning to install our new terminal driver. With the old, standard terminal driver, the screen gave you no indication that the previous character had been deleted when you pressed the erase character. You had to accept it on faith. This remains true on many UNIX systems today. Most people on Cory Hall UNIX changed their erase character to backspace so that later characters would overwrite the erased ones, but even that was not sufficient. This was especially true when erasing a backslash, which counter-intuitively required two erase characters. We wanted the system to show that the character

Continued to Page 108

Continued from Page 44

was gone by blanking it out. We also wanted the line-erase character to display a blanked-out line. Some UNIX systems such as 4.2 BSD and System V now support this, but it was not then available anywhere under UNIX Version 6.

Bob and I had argued, somewhat sleepily, for hours as to the correct method of erasing characters, and Bob had started putting our joint design into effect just as I collapsed on the floor for "a short nap". I awoke around dawn to find Bob asleep over the terminal. When he woke up, he said he was pretty sure he'd finished the job before falling asleep, but neither of us had enough energy to check. It was time for food and 14 hours of sleep.

When we finally checked our handiwork the next day, we found some serious flaws in the implementation — not an uncommon situation with work performed under extreme conditions. But the system was up and running, and although the new features were flawed, they didn't seem to cause any problems, so we forgot about it for the time being. A week later, I was consulting in Cory — we all offered free programming help to other students in the time-honored tradition of hackers everywhere — when Kurt Schoens called me over to the other side of the room.

"Hey Doug," he said. "Look at this. It looks like someone tried to put character deletion into the terminal drivers, but only half finished."

My heart raced. Did he suspect me? Or was he just chatting? I could never tell whether Kurt was kidding; he had the most perfect poker face I had ever seen. But he quickly made the question academic, and proved again that he was one of Them.

"I showed this to Bill, and he wanted to fix it", Kurt said. "Oh, really?" I stammered. "Sounds good to me," thinking that it was a real stroke of luck that Bill Joy would be interested in the half-completed project. If Bill finished it, then it would be in the system on legitimate grounds, and would stay for good.

Kurt paused for effect. "Yeah, he was all fired up about it, but I talked him out of it, and I just deleted it from the system instead."

Oh, cruel fate! Kurt must know that I was involved; he just

UNIX*

Marketing and Technical Professionals
UNIFORUM - Dallas January 21-25

The Hottest UNIX* Opportunities Are With Gould's Firebreathing Team in Florida. Our family of 32-bit minicomputers blast the competition into oblivion. The Firebreather Team has the following opportunities:

UNIX* Development
- Drivers
- Communications
- Internals
- "C"

Sales Support
- Los Angeles
- Chicago
- Boston
- Washington, D.C.
- Ft. Lauderdale

Marketing/Management
- Product Market
- Market Development
- Promotion
- Product Marketing

If you're contemplating a career move and plan to attend the UNIFORUM Conference in Dallas, call Al Smith collect at (305) 797-5658 to arrange a local interview with our technical staff. If unable to attend, please call or send resume to Al Smith, Dept. UR185, **GOULD INC.**, Computer Systems Division, 6901 W. Sunrise Blvd., Ft. Lauderdale, FL 33313. An Equal Opportunity Employer, M/F/H/V.

**UNIX is a trademark of AT&T Bell Laboratories*

GOULD
Electronics

Circle No. 63 on Inquiry Card

wanted to see me jump when he said "boo!"

Although I'm sure Kurt thought the whole incident very funny, all I could think of was that yet another of my features had gone down the drain. I discussed this latest setback with others in the group, and we shared a sense of frustration. More than ever before, we were determined to get our contributions accepted somehow.

Kurt was both a graduate student and a system administrator, but I liked him all the same — chiefly because of his practical jokes. We had recently cooperated in a spontaneous demonstration of Artificial Intelligence at the expense of an undergraduate named Dave who had joined Them as a system administrator. Dave had watched Kurt as he typed **pwd** to his shell prompt and received **usr/kurt/mind** as the response. His next command had been **mind -i -l english** . During all this time, Kurt was double-talking about psychology and natural language processing and some new approach to simulating the human mind that he'd thought of. Dave looked dubious, but was willing to see how well Kurt's program worked.

What Dave didn't realize was that Kurt had not been typing commands to the system at all; although we were sitting not 10 feet apart, Kurt and I had been writing to each other and chatting for half an hour, and as a joke I had been pretending I was Kurt's shell, sending him prompts and faking responses to commands. Dave had walked in at just the right time. So when Kurt typed **mind -i -l english** , I had naturally responded with:

```
"Synthetic Cognition System, version 17.8"
"    Interactive mode on,
         Language = english"
"Please enter desired conversational
     topic: (default:philosophy)"
```

Dave couldn't help looking a little impressed; Kurt's "artificial intelligence" system was off to a great start. Kurt had talked to his budding mind for several minutes, and Dave of course had grown more and more impressed. Kurt and I faced the greatest challenge of our lives in keeping a straight face during the demonstration, but we eventually made the mistake of making the mind altogether TOO smart to be believable, in effect sending Dave off to tackle more serious work.

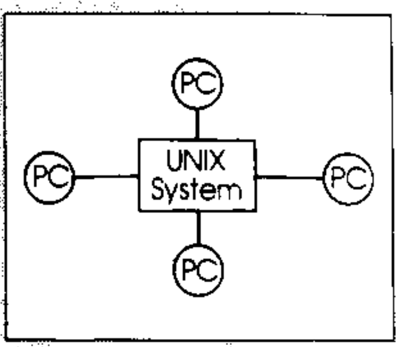
There was one practical joke that was notable for the length of time it was supported by the entire group. The target was system administrator Dave Mosher. Dave had been suspicious of bugs in our system's homebrewed terminal multiplexer for some time. Ross decided to persecute Dave by having random characters appear on his screen from time to time, which of course convinced Dave that the terminal multiplexer did indeed have problems. To help Ross with the prank, each of us sent Dave some garbage characters at random intervals whenever any one of us was on the system. We had settled on the letter "Q" so that Dave would be sure it was always the same bug showing the same symptom. Since Dave had these problems no matter which terminal he was on, day or night, no matter who else was logged onto the system, he was positive there was a problem, and he spent much time and effort trying to get someone to fix it.

Unfortunately for Dave, he was the only one who ever saw these symptoms, so everyone thought he was a little paranoid. We thought it was pretty funny at first, but after a few months of this, it seemed that Dave was really getting rattled, so one day Ross generated a capital "Q" as big as the entire terminal screen and sent it to Dave's screen. This made it pretty obvious to poor Dave that

PC*works*™

WORKS!

Here at BASIS, we've checked PC*works* carefully. And it does just what it claims to do: it makes your IBM PC™ (or compatible) a part of the UNIX™ revolution.



Now you and your PC can:

- Access UNIX programs
- Transfer PC & UNIX files
- Use the UNIX printer
- Back up PC files on UNIX
- Read, Create, & Send mail
- Access public & private networks

CALL or WRITE for More Information

B.A.S.I.S.

UNIX Software, Hardware, & Timesharing

1700 Shattuck #1
Berkeley, CA 94709
(415) 841-1800

Circle No. 64 on Inquiry Card

someone, somehow, really *had* been persecuting him, and that he wasn't paranoid after all. He had an understandably low tolerance for practical jokes after that.

The numerous practical jokes we played were probably a reaction to the high level of stress we felt from our ongoing illicit operations: it provided some moments of delightful release from what was, at times, a grim battle. There were many secret battles in the war: if Our motto was "Features!", Theirs was "Security for Security's Sake" and the more the better. We were never sure how long our victories would last: on the other hand, They were never sure whether They had won. The war lasted almost three years.

We were primarily interested in the EECS department's PDP 11/70 in Cory Hall, since that was the original UNIX site and continued to be the hotbed of UNIX development, but We "collected" all the other UNIX systems on campus, too. One peculiar aspect of the way the Underground had to operate was that we rarely knew the root password on systems to which we had gained superuser access. This is because there were easier ways to get into, and stay into, a system than guessing the root password. We tampered, for instance, with the su program so that it would make someone superuser when given our own secret password as well as when given the usual root password, which remained unknown to us. In the early days, one system administrator would mail a new root password to all the other system administrators on the system, apparently not realizing that we were monitoring their mail for exactly this kind of security slip. Sadly, they soon guessed that this was not a good procedure, and we had to return to functioning as "password-less superusers", which at times could be a bit inconvenient.

Late one night on Cory Hall UNIX, as I was using my illegitimate superuser powers to browse through protected but interesting portions of the system, I happened to notice a suspicious-looking file called *usr/adm/su*. This was suspicious because there were almost never any new files in the administrative *usr/adm* directory. If I was suspicious when I saw the filename, I was half paralyzed when I saw it contained a full record of every command executed by anyone who had worked as superuser since the previous day, and I was in a full state of shock when I found, at the end of the file, a record of all the commands that I'd executed during my current surreptitious session, up to and including reading the damning file.

It took me perhaps 10 minutes of panic-stricken worry before I realized that I could edit the record and delete all references to my illicit commands. I then immediately logged out and warned all other members of the group. Since nothing illicit ever appeared, the system administrators were lulled into a sense of false security. Their strategy worked brilliantly for us, allowing us to work in peace for quite a while before the next set of traps were laid.

The next potential trap I found was another new file in */usr/adm* called *password*, that kept track of all unsuccessful attempts to login as root or to su to root, and what password was used

Great-looking TROFF output from low-cost laser printer!

For several years, Textware has been licensing **TPLUS** software to process the output of TROFF and DITROFF for a wide variety of typesetters, laser printers, letter-quality and dot-matrix printers, and high-resolution displays. Our device-independent approach permits **TPLUS** to exploit the specific capabilities (and to work around the limitations) of all the imaging devices we support.

Now we are excited to announce the availability of **TPLUS** for the Hewlett-Packard **LaserJet** printer. The low-cost **LaserJet** is a remarkable value on its own—8 page per minute output speed, 300 dot per inch resolution, and typesetter-quality fonts. **TPLUS** gives you access to all this ***and more*** from your own system. We support all the characters and accents needed by TROFF and EQN; in addition, special characters (Ⓢ; logos too) can be supplied or generated to meet specific requirements. Our precise handling of rules and boxes allows you to take full advantage of TBL for forms, charts, etc.

EQN examples	
$\lim_{x \to \pi/2} (\tan x)^{\sin 2x} = 1$	
$\frac{\alpha+\beta}{\sin(x)}$	$\prod_{k \geq 1} e^{S_k x^k / k}$

If you haven't been impressed with the output from low-cost laser printers in the past, take another look. While even **LaserJet** output is not in the same class as the best of phototype, it is certainly well suited to documentation and a broad range of other applications. When you do have a need for phototypeset images, **TPLUS** and the **LaserJet** will help save you time and money. *Preview mode* lets you proof *all* aspects of your documents conveniently, in-house, before sending out for finished phototypesetting (from our **UNI•TEXT** service).

The HP **LaserJet** printer is not only inexpensive—it is an exceptional value! Want proof? This entire ad was set in position using **TPLUS** on the **LaserJet**!

Also available for:

- AM 5810/5900 & 6400, APS 5 & µ5, CG 8400 & 8600, Mergenthaler 202
- Xerox 2700, 8700 & 9700
- BBN, Sun & 5620 CRTs
- Diablo, Qume & NEC LQPs
- C Itoh & Epson dot-matrix

For further information, please write or call.

TEXTWARE
INTERNATIONAL
PO Box 14 Harvard Square
Cambridge, MA 02238
Telephone:
(617) UNI-TEXT

in the attempt. Since none of us had known the root password for months and therefore weren't going to become superuser by anything as obvious as logging in as root, this wasn't particularly threatening to us, but it was very interesting. The first few days that we watched the file it showed attempts by legitimate system administrators who had made mistakes of various sorts. One of Them once gave a password that We discovered, through trial and error, to be the root password on a different system. Several of Them gave passwords that seemed to be the previous root password. Most of them were misspellings of the correct root password. Needless to say, this was a rather broad hint, and it took Us less than five minutes to ascertain what the correct spelling was.

One might think that, since we had several ways to become superuser anyway, it wouldn't make any real difference whether or not we knew the actual root password as well. The problem was that our methods worked only so long as nothing drastically changed in the system; the usual way that They managed to win a battle was to backup the entire system from tape and recompile all utilities. That sometimes set Us back weeks, since it undid all of our "backdoors" into superuserdom, forcing us to start from ground zero on breaking into the system again. But once we knew the root password, we could always use that as a starting place.

We worked very hard to stay one step ahead of Them, and we spent most of our free time reading source code, in search of either pure knowledge or another weapon for the battle. At one time, I had modified every single utility that ran as superuser with some kind of hidden feature that could be triggered to give us superuser powers. Chuck Haley once sent a letter to Jeff Schriebman commenting that he "had even found the card reader program" to show signs of tampering. I thought that I had disguised it well, but it was extremely difficult to keep things hidden from a group of system administrators who were not only very intelligent, but also highly knowledgeable about the inner

UNIX™ VALIDATION

formerly THE SYSTEMS GROUP

PERENNIAL

Validation Suite versions available

- UNIX System III and System V (tm Bell Labs)
- BSD 4.1 and BSD 4.2
- /usr/group Standard
- XENIX (tm Microsoft)

Validation Suites Include

- Source code for entire suite
- Template for adding tests
- Installation support service
- User documentation
- Audit trail for all tests
- Test driver control program
- System exercisor—stress tests
- Automatic and manual tests
- Touch tests of all utilities
- Functional tests for all system calls

PERENNIAL SOFTWARE SERVICES GROUP

3130 De La Cruz Blvd. • Santa Clara, California 95054 • (408) 727-2255

SOFTWARE DEVELOPMENT, TESTING, MAINTENANCE AND SUPPORT SERVICES

Circle No. 66 on Inquiry Card

workings of UNIX. As an indication of the caliber of the people we were working against, I should note that Chuck Haley is now a researcher at Bell Labs; Bill Joy is VP of Engineering at Sun Microsystems; Kurt Schoens is a researcher at IBM; Jeff Schriebman is founder and President of UniSoft; and Bob Kridle, Vance Vaughn, and Ed Gould are founders of Mt. Xinu.

This was an unusual situation; system administrators are not usually this talented. Otherwise, they'd be doing software development rather than administration. But at the time, there was no one else *capable* of doing UNIX system administration.

As a result, we had to move quickly, quietly, and cleverly to stay ahead, and planting devious devices in the midst of standard software was our primary technique. Normally trusted programs which have been corrupted in this way are called "Trojan Horses", after the legend of the Greeks who were taken in by a bit of misplaced trust. One of our favorite tricks for hiding our tracks when we modified standard utilities was the diddlei program, which allowed us to reset the last change time on a modified file so that it appeared to have been unchanged since the previous year. Bob modified the **setuid** system call in the UNIX kernel so that, under certain circumstances, it would give the program that used it root privileges. The "certain circumstances" consisted simply of leaving a capital "S" (for Superuser) in one of the machine's registers. Bob was bold enough to leave this little feature in the system's source code. We usually put our Trojan Horses in the system executables only — to decrease the chance of it being noticed. But Bob took the chance so that the feature would persist even if the system were recompiled. Sure enough, it lasted for several months and through more than one system compilation before Dave Mosher noticed it (undoubtedly with a sense of shock) as he was patiently adding comments to the previously undocumented kernel.

This sort of battling continued for several years, and although They were suspicious of most of Us at one time or another, none of Us was ever caught red-handed. It undoubtedly helped that we never performed any malicious acts. We perhaps flouted authority, but we always enhanced the system's features. We never interfered with the system's normal operation, nor damaged any user's files. We learned that absolute power need not corrupt absolutely; instead it taught us restraint.

This is probably why we were eventually accepted as members of the system staff, even though by then several of Us had confessed to our nefarious deeds. Once we were given license to modify and improve UNIX, we lost all motivation to crack system security. We didn't know it at the time, but this has long been known to be one of the most effective ways of dealing with security problems: hire the offenders, so that there is no more Us versus Them, but simply Us.

It worked well in our case; under the auspices of the System Development and Research Group, created by the ever-industrious Dave Mosher, we went happily to work on UNIX development. The development of UNIX at Berkeley, always fast-paced, exploded once everyone — including undergraduates — were participating.

The only fly in the ointment was the introduction a short while later of UNIX Version 7. While it was a vast improvement in many ways over the Version 6 that we had been working with, most of the enhancements we had developed were lost in the changeover. Some were reimplemented under Version 7 by those of the group who remained at Berkeley, but by then many of us were leaving school, and the impetus behind our ideas left with us.

Ken Arnold is, perhaps, the most famous of our original group. He stayed at Berkeley longer than any of the rest of us, and became well known for such contributions as **Termlib**, **curses**,**fortune**, Mille Bourne, and of course his co-authorship of Rogue. But somehow it seemed a Pyrrhic victory even for Ken; much of his best work in the early years never saw the light of day.

We could not help but feel that we had passed through a sort of Dark Age for UNIX development, and even with the Renaissance in full bloom. We ponder what might have been, and bewail the features that UNIX will now never have.

Doug Merritt became one of the earliest UNIX users outside Bell Laboratories while attending UC Berkeley in 1976. He helped to debug ***termcap*** *and contributed to the development of* ***vi*** *and* ***curses****. Mr. Merritt now works as a consultant in the San Francisco Bay Area.*

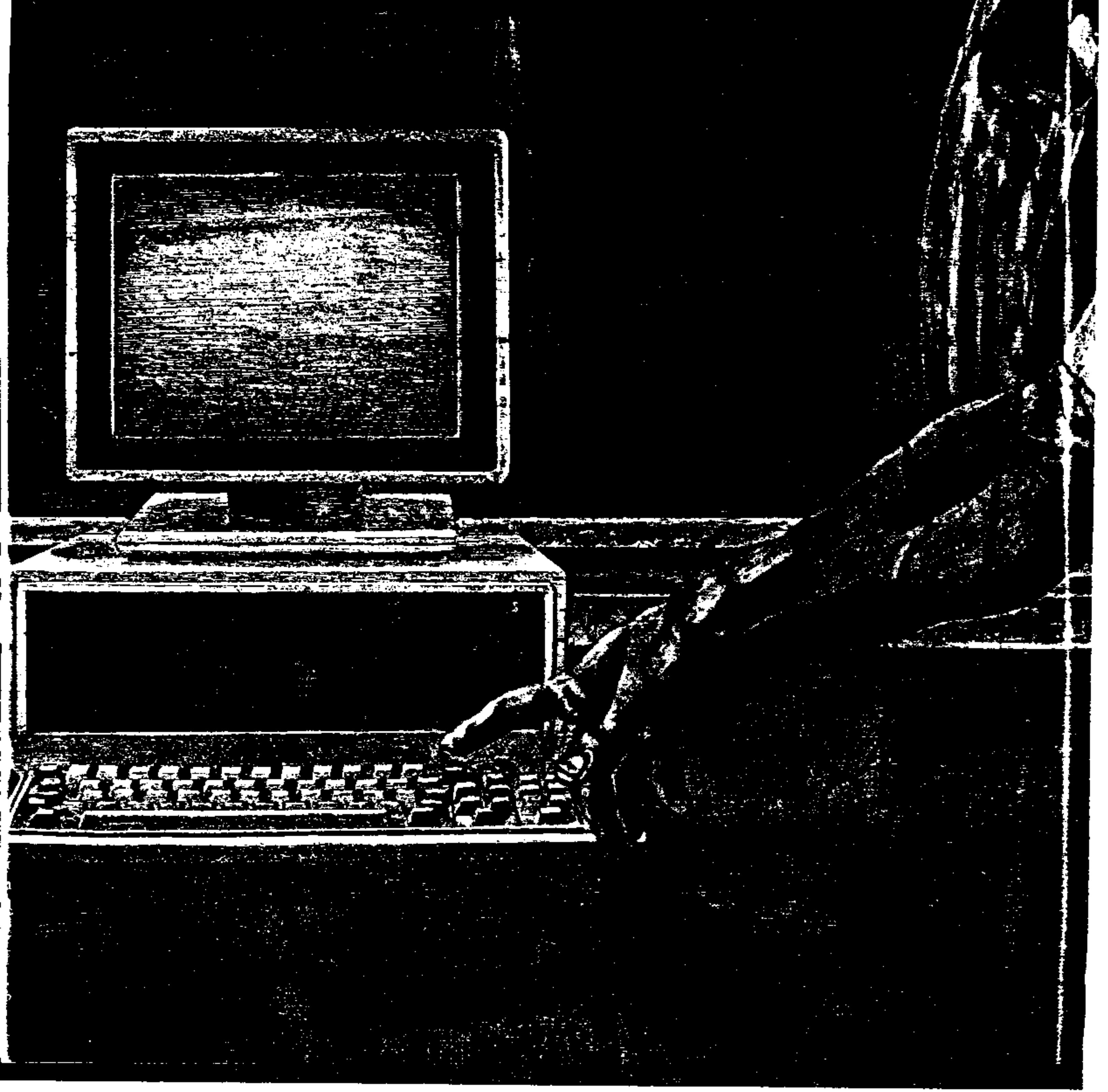
Bob Toxen is a member of the technical staff at Silicon Graphics, Inc., who has gained a reputation as a leading expert on ***uucp*** *communications, file system repair and UNIX utilities. He has also done ports of System III and System V to systems based on the Zilog 8000 and Motorola 68010 chips.*

Best known as the author of curses and co-author of Rogue, Ken Arnold was also President of the Berkeley Computer Club and the Computer Science Undergraduates Association during his years at UC Berkeley. He currently works as a programmer in the Computer Graphics Lab at UC San Francisco and serves as a member of the UNIX REVIEW *Software Review Board.* ■

THE GENESIS STORY

An unofficial, irreverent, incomplete account of how the UNIX operating system developed

by August Mohr



This is, so to speak, a history of how UNIX evolved as a product: not the "official" history of who was responsible for what features, and what year which milestones were crossed, but the "political" history of how decisions were made and what motivated the people involved. Most of the readers of this magazine are familiar with the system itself, so I don't want to go into great detail about how the system got to be what it is internally, but rather how it got to be at all.

Three years ago, I was one of many espousing the idea that Bell Laboratories and Western Electric had engaged in a masterful piece of long-term strategy by first releasing UNIX to universities and then later releasing it to the commercial world. How clever, I thought, to get universities involved in the development of the system and get a whole crop of UNIX experts trained in the process. But after talking to some of the people involved, I have reversed my opinion. Bell, it seems, was dragged kicking and screaming into providing UNIX to the world.

Illustration by Kelly Davies

IN THE BEGINNING, THERE WAS MULTICS

Multics, in some ways UNIX's predecessor, was a huge project, a combined effort of three of the largest computing centers of the day. All three principles — Bell, GE, and MIT, had already done operating systems, even *time-shared* systems, before, and following Fred Brooks' *Second System Syndrome*, it was felt Multics was going to solve all the problems of its predecessors. Ultimately, the project proved to be too late and Bell dropped out, leaving Ken Thompson, Dennis Ritchie, and Rudd Canaday without a timeshared system to play with.

So they set about building an operating system of their own.

So until this point, timesharing systems had only been developed for big machines costing hundreds of thousands of dollars. It was unlikely that Computer Research was going to get another investment of that sort out of Bell so soon after Multics. In any event, Thompson, Ritchie, and Canaday weren't particularly interested in working on another large scale project, so they set their sights on obtaining a minicomputer on which to build a timesharing system they could use as a program development environment. They wanted the kind of flexibility and power they had worked toward in Multics, without incurring the expensive rings of protection and interlocks.

The project was short on computing power, though. With only a PDP-7 to work on, the researchers yearned for another machine — preferably a VAX 11/20. To get the necessary funds for a new machine, though, it was clear they would first need to find an application.

Fortunately, the Bell Labs' Legal Department — which was close at hand to Thompson and Ritchie's Murray Hill office — was looking for a word processing system at about this time. A paper-tape system for an old Teletype machine was under consideration.

With a bit of salesmanship, Thompson and Ritchie got the legal team interested in a UNIX-supported system. It proved to be a momentous deal. The UNIX project got the machine it needed and the legal department got the word processing system it wanted — along with some rather impressive local support.

The combination of program development and word processing was to have serendipitous effects.

A number of important barriers thus were crashed. First, the business side of Bell Labs, generally held at arm's length by the people in Research, got a healthy dose of Research assistance. More importantly for UNIX users, Thompson and Ritchie got their first live customer. UNIX word processing suddenly had to be usable by secretarial staff as well as by programming staff. The changes made to accomplish this transition were to have serendipitous effects as UNIX evolved.

Richard Haight, now AT&T Bell Labs Supervisor of Video Systems Software (a UNIX application), had his first exposure to UNIX at about this time. As he remembers it, "I came across Ken Thompson by accident and wanted to do some software development on a PDP-11. He said he had a machine I could use but that it didn't run on a DEC operating system. I had lots of exposure to timesharing systems and it was pretty obvious that this was superior in many ways.

"It was uphill in the beginning. The fact that it was homegrown in the Labs didn't cut anything with the people that I was working with at that time. They went and visited Ken and Dennis in their sixth-floor attic at Murray Hill and all they saw was hardware laying all over the floor and a bunch of guys in T-shirts and sneakers. It wasn't the sort of place that would warm the heart of a manager who had been brought up in a traditional data processing environment. It really was the 'two good guys in a garage' kind of syndrome except that they happened to be in an attic.

"That first time I met Ken Thompson, he had a small bookshelf, maybe 2 1/2 feet long, above an old Teletype Model 37 terminal. On the shelf were hardware manuals from Digital Equipment for the PDP-11, occupying maybe five inches, and the rest of the shelf was nothing but chess books. I think Thompson will tell you himself that he developed UNIX as a good place to develop chess programs. It turns out that it's everybody else's idea of where and how to develop programs too."

Haight has had many years of experience in the Bell Labs environment. Before moving to his current post, he served as part of the UNIX Support group and the PWB (Programmer's Workbench) development group. He believes that the environment of the Labs was a major factor in the creation of UNIX.

"There's a small fraction of people who just go crazy over computers," he explained. We hire a bunch of them and they remain workaholics for several years after coming in. Eventually they get a house and a mortgage and they get married and have kids and settle down to be normal human beings, but it's wonderful to hire these people because you get two or three people's work out of them for several years. Those are the kind of people that give you things like UNIX."

AN INFLUENTIAL FEATURE: PIPES

Dick Haight is one of the people who became hooked on UNIX at the very beginning, and even yet is an outspoken proponent of its value and power — although he will tell you he is still waiting for something better to come along. "There's a lot of complaint about UNIX being terse and for experts only," he said. "I've seen that blamed on the pipe mechanism. If terseness is the price for having pipes, I'll take it any day."

As one of the new system's first users, Haight got to follow many of the changes. "I happened to have been visiting the research crew the day they implemented pipes," he recalled. It was clear to everyone, practically minutes after the system came up with pipes working, that it was a wonderful thing. Nobody would ever go back and give that up if they could help it."

Haight believes that the pipe facility has had a major effect on the evolution of UNIX. Because of the shell and the piping facility, programs running under UNIX can be conversational without being explicitly programmed that way. This is because piping adds to the shell's ability to accomplish this purpose for the system as a whole.

Doug McIlroy is usually credited with the idea of adding pipes, although the idea may have been around since Multics. Haight believes there may have been yet another reason for implementing them. The original UNIX had a file size limitation of 64K and, according to Haight, "one of the people there in research was constantly blowing that size restriction in an intermediate pass of the Assembler." Between McIlroy's lobbying for the idea and this other problem with file size, Thompson and Ritchie were finally convinced to implement pipes.

MOVING INTO THE COMPANY

Independent of what was happening in the research area, Bell was starting to perceive the need for minicomputer support for its telephone operations. It needed Operations Systems, not Operating Systems. With the numbers of systems under consideration, the possibility of being tied to a single vendor, or having each site tied to a different vendor, induced a kind of paranoia. There just had to be another way.

"Naturally I knew that once they got on UNIX they wouldn't be able to get off. It's just like drugs."

The groups responsible for developing operations systems had many people from hardware and applications software backgrounds who were considering writing their own operating system — their first — when Berkley Tague, now head of Bell Labs' Computing Technology Department, suggested they use UNIX to get started.

"I observed that people were starting to put these minis out in the operating company, and saw that it was an area of both opportunity and potential problems," Tague remembered. "I found that some of the people in development had never built an operating system for any computer before; many of them had very little software background. They were coming out of hardware development and telephone technology backgrounds, and yet were starting to build their own operating systems. Having been through that phase of the business myself, it seemed silly to go through it another hundred times, so I started pushing the UNIX operating system into these projects."

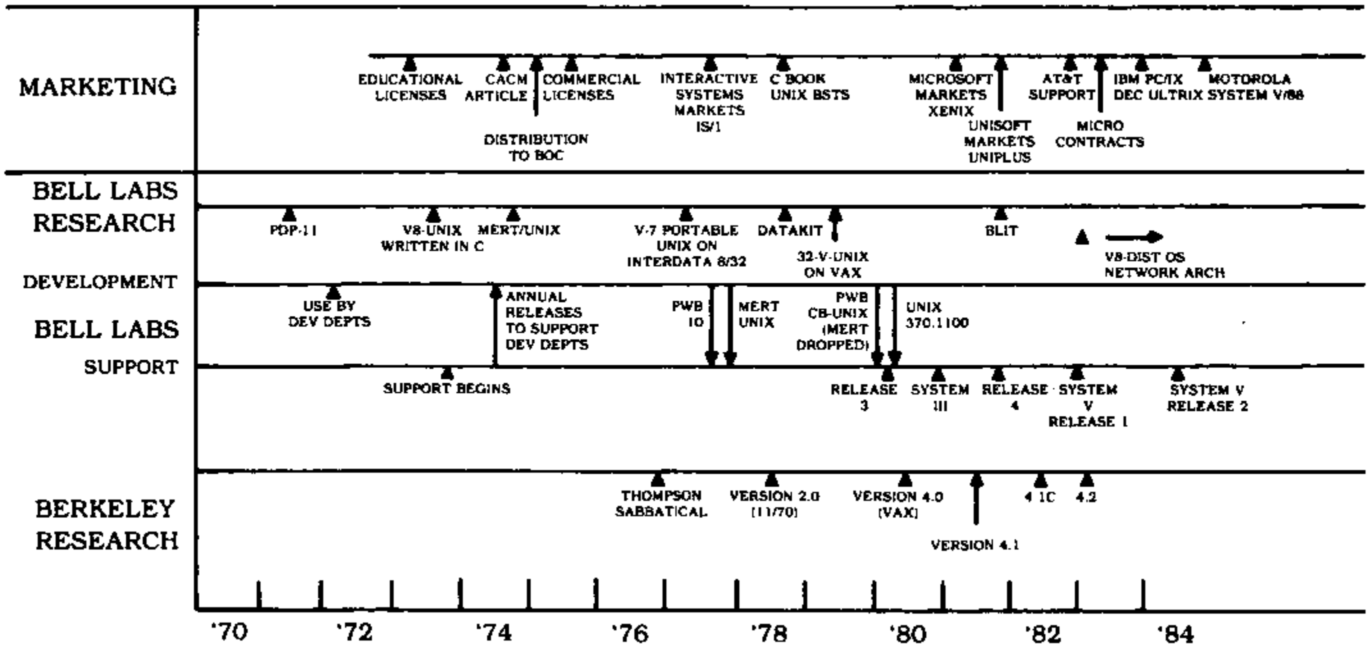
Tague's backing of UNIX, as a development system for operations, was not just a personal preference. "I had every confidence in the people who built it because I'd worked with them on Multics," he explained. "With their experience and training, I figured they could build a much better operating system than somebody who's building one for the first time, no matter how smart that person is."

SUPPORT?

UNIX had been running long enough in research by that time that Tague knew that the system the operations group would get would serve as a very good starting point. Unfortunately, there was no vendor support for it.

The argument Tague made for UNIX was: if the operations people were going to build their

KEY EVENTS IN UNIX SYSTEM EVOLUTION



Courtesy of Larry Crume, AT&T UNIX Pacific.

own system, they were going to have to maintain it themselves. Surely, UNIX could be no worse. They could use it to get started and do the development. If a more efficient or better operating system was needed for a target machine when they got into the field, they could always build it, but UNIX would at least get them off the ground. "Naturally I knew that once they got on this thing they wouldn't be able to get off. It's just like drugs," Tague explained.

Tague also knew it was important to get some field support. "We were starting to put these things in the operating companies all around the countryside and the prospects were that there were going to be several hundred minis over the next few years that were going to have to be maintained with all their software and hardware," he said.

Bell had already gained some field support experience maintaining electronic switching machines and their software. Supporting a network of minicomputers would be a significantly different problem, though. Maintaining an operating system is not at all like maintaining an electronic switching system. The minicomputers had different reliability demands, requiring a different support structure in the organization — one that did not yet exist in any form. In many ways, the operations group was breaking new ground.

Up to this point, Tague had served as head of the Computer Planning department responsible for systems engineering. After gaining support for UNIX in the operations group over the course of 1971 and 1972, he made a push for two significant changes. The first was to make UNIX an internal standard and the second was to offer central support through his organization. In September, 1973, he was permitted to form a group called UNIX Development Support, the first development organization supporting a "Standard UNIX". While this group worked closely with Bell Labs Research, its concerns sometimes diverged.

One area the two groups could agree on, though, was portability. By 1973, it was already on the horizon. Tague foresaw the possibility of UNIX becoming an interface between hardware and software that would allow applications to keep running while the hardware underneath was changing.

From the support point of view, such a capability would solve a very important problem. Without UNIX and its potential portability, the people building the operations support systems were faced with selecting an outside vendor that could supply the hardware on which to get their development done. Once that was complete, they would be locked into that vendor. Portability obviated this limitation and offered a number of other advantages. When making a hardware upgrade, even to equipment from the same vendor, there are variations from version to version. That could cost a lot of money in software revisions unless there were some level of portability already written into the scenario. Fortunately, the integral portability of the system developed by Research proved adequate to make UNIX portable over a wide range of hardware.

The first UNIX applications were installed in 1973 on a system involved in updating directory information and intercepting calls to numbers that had been changed. The automatic intercept system was delivered for use on early PDP-11s. This was essentially the first time UNIX was used to support an actual, ongoing operating business.

To Tague, at this time, "our real problem was pruning the tree". There were so many different sites using UNIX that each would come up with different answers to the same problems of printer spooling, mail, help, and so on. "The customers would invent this stuff and make it work and our problem was to get the slightly different variations together, get the best of all of those worlds, put it in the standard, and get it out again." This was, in many ways, a political process. Tague credits the "technical underpinnings" for making the process easier than expected. "That made it easy to get the right stuff in without upsetting the whole world. I didn't have to go to all of my customers and tell them that this was now my new version and that nothing they had out in the field would run again," he said.

To provide a standard UNIX system, the support group had to establish what version it would back. This was a process of negotiation and compromise with the UNIX-using community — not a unilateral decision. The support team and customers often ended up arguing things out until everybody understood the issues and a suitable compromise was made. "As one of the local gurus put it to me one time," Tague said, "one of the problems in UNIX is that everybody wants to carve his initials in the tree." When the choice came down essentially to tossing a coin, Tague, as arbitrator, tried to make sure that each group got at least one pet contribution into the system.

From the very beginning within Bell, UNIX followed what has become a familiar pattern of users leading their management.

Fortunately, UNIX is flexible enough that even the particularly traumatic decisions, such as the ones concerning standard shell versions, could be patched in slowly — at the user's discretion.

A FAMILIAR PATTERN

From the very beginning within Bell, UNIX followed what has become a familiar pattern of users leading their management. While this is not the most common marketing strategy in the commercial world, it is typical of Bell Labs' "bottom-up" organization. According to Rudd Canaday, now head of Bell Labs' Artificial Intelligence and Computing Environments Research Department, change within the Labs often comes from the people doing the work. "UNIX spread throughout Bell Laboratories because people loved to use it," he said.

Canaday first experienced UNIX, although it hadn't yet been named, while working with Thompson and Ritchie. He left that group and later became head of a group building large, mainframe-oriented systems. Because of his previous exposure to UNIX, he wanted to bring it to his new group.

Canaday found lots of support among the programmers who had already tasted UNIX. One of those, Richard Haight, recalled, "I was trying to get my management to get UNIX and we dreamed up the idea of using it as a common timesharing interface to different kinds of host computers."

Initially, the project involved interfacing with big IBM and Univac machines, and later expanded to interfacing with RCA, Xerox, and others. The basic idea was to edit programs and work with files under UNIX, but instead of compiling and executing under UNIX, you could send the remote job off to a big machine. This way, the programmer didn't have to deal with complicated IBM JCL sequences since he could just give the UNIX utility the parameters it needed to know. The masses of printout could then come back as a file under UNIX and, as Dick Haight put it, "save cutting down a tree." It also saved having to retrain programmers for a variety of host systems.

This original Programmer's Workbench system was built on a PDP 11/45. The system eventually offered lots of utilities, including ones for analyzing host machine dumps on the UNIX system.

While work proceeded on the PWB system, an interesting discovery was made. The designers had assumed that the majority of the work cycle would involve the host computer. Users were thought of as editing a file, sending it to a host computer, getting the print file back, looking at it, and doing that over and over again. As it turned out, samples taken from different kinds of work groups on different systems showed people tended to use the text formatter, **nroff**, five times as often as they submitted Remote Job Entry programs.

This unexpected result might not have happened had UNIX not had fairly sophisticated word processing facilities available to programmers. The original development for Bell's legal department could hardly be called "incredible foresight", but happily for UNIX, word processing was to become the single most commonly used computer application. Once the facilities were there, programmers made massive, unexpected use of them. This happened, according to Haight, because programmers have to be able to document programs on the same machine used for development. "Things like pipes and the power of the shell are not to be slighted, but what's really important is the fact that you can do your documentation and your programming on the same machine," he said. You can be editing your documentation and break away from that to edit the source. When you're finished with that, you can submit a compile in the background and go back to editing your documentation while the compile happens."

Flushed with the success of the PWB and the Remote Job Entry facility, Canaday and his group set about showing people what was possible. Once the users were convinced, Canaday said to management, "Well, if you want to keep on using this, you're going to have to start buying machines to do it." He knew that "once you let people get their hands on UNIX, they just won't let go."

A key piece in the rapid spread of UNIX within Bell Labs was the low price of minicomputers relative to mainframes. A department head's urging was generally sufficient for purchase of a VAX. Mainframe purchases were considerably more sticky. A VAX had sufficient power to reasonably serve the needs of a department, so VAXen became increasingly commonplace.

More and more departments were becoming convinced that UNIX was part of the path toward

Continued to Page 117

REFLECTIONS ON SOFTWARE RESEARCH

Can the circumstances that nurtured the UNIX project be produced again?

by Dennis M. Ritchie

The UNIX operating system has suddenly become news, but it is not new. It began in 1969 when Ken Thompson discovered a little-used PDP-7 computer and set out to fashion a computing environment that he liked. His work soon attracted me; I joined in the enterprise, though most of the ideas, and most of the work for that matter, were his. Before long, others from our group in the research area of AT&T Bell Laboratories were using the system; Joe Ossanna, Doug McIlroy, and Bob Morris were especially enthusiastic critics and contributors. In 1971, we acquired a PDP-11, and by the end of that year we were supporting our first real users: three typists entering patent applications. In 1973, the system was rewritten in the C language, and in that year, too, it was first described publicly at the Operating Systems Principles conference; the resulting paper [8] appeared in *Communications of the ACM* the next year.

Thereafter, its use grew steadily, both inside and outside of Bell Laboratories. A development group was established to support projects inside the company, and several research versions were licensed for outside use.

The last research distribution was the seventh edition system, which appeared in 1979; more recently, AT&T began to market System III, and now offers System

Continued to Page 118

Continued from Page 28

V, both products of the development group. All research versions were "as is," unsupported software; System V is a supported product on several different hardware lines, most recently including the 3B systems designed and built by AT&T.

UNIX is in wide use, and is now even spoken of as a possible industry standard. How did it come to succeed?

There are, of course, its technical merits. Because the system and its history have been discussed at some length in the literature [6, 7, 11], I will not talk about these qualities except for one. Despite its frequent surface inconsistency, so colorfully annotated by Don Norman in his *Datamation* article [4] and despite its richness, UNIX is a simple, coherent system that pushes a few good ideas and models to the limit. It is this aspect of the system, above all, that endears it to its adherents.

Beyond technical considerations, there were sociological forces that contributed to its success. First, it appeared at a time when alternatives to large, centrally administered computation centers were becoming possible; the 1970s were the decade of the minicomputer. Small groups could set up their own computational facilities. Because they were starting afresh, and because manufacturers' software was, at best, unimaginative and often horrible, some adventuresome people were willing to take a chance on a new and intriguing, even though unsupported, operating system.

Second, UNIX was first available on the PDP-11, one of the most successful of the new minicomputers that appeared in the 1970s, and soon its portability brought it to many new machines as they appeared. At the time that UNIX was created, we were pushing hard for a machine, either a DEC PDP-10 or SDS (later Xerox) Sigma 7. It is certain, in retrospect, that if we had succeeded in acquiring such a machine, UNIX might have been written but would have withered away. Similarly, UNIX owes much to Multics [5], I have described [6, 7], it eclipsed its parent as much because it does not demand unusual hardware support as because of any other qualities.

Finally, UNIX enjoyed an unusually long gestation period. During much of this time (say 1969-1979), the system was effectively under the control of its designers and being used by them. It took time to develop all of the ideas and software, but even though the system was still being developed, people were using it, both inside Bell Labs, and outside under license. Thus, we managed to keep the central ideas in hand, while accumulating a base of enthusiastic, technically competent users who contributed ideas and programs in a calm, communicative, and noncompetitive environment. Some outside contributions were substantial, such as those from the University of California at Berkeley. Our users were widely, though thinly, distributed within the company, at universities, and at some commercial and government organizations. The system became important in the intellectual, if not yet commercial, marketplace because of this network of early users.

What does industrial computer science research consist of? Some people have the impression that the original UNIX work was a bootleg project, a "skunk works". This is not so. Research workers are supposed to discover or invent new things, and although in the early days we subsisted on meager hardware,

UNIX* and C TRAINING

UNIX Overview (1)	$145
UNIX Fundamentals (3 days)	$595
C Fundamentals (3 days)	$595

Austin Schedule

Feb. 6-8	UNIX
Feb. 13-15	C
Mar. 9-11	UNIX
Mar. 16-18	C

On-site training available

Call or write to register or to get on our mailing list.

TELOS CONSULTING SERVICES
815 Brazos, Suite 504
Austin, TX 78701
512/478-2072

*UNIX is a trademark of Bell Laboratories.

NOW GET BERKELEY ENHANCEMENTS FOR YOUR 3B2

Z Systems is now offering its port of the Berkeley cshell for your AT&T 3B2. Through enrollment in the 3B Subscription Service, you will get the first software release which includes:
- Berkeley 4.2 compatible
- csh (less jobs)
- ls (links to lf, lr, . . .)
- more
- strings
- and others

Your annual subscription to 3BSS also includes newsletters and future software releases for a nominal media and handling charge.

The cost is just $200/yr.

To order call 303-526-1633
major credit cards accepted

Z SYSTEMS, INC.
602 Park Point Drive
Genesee Business Center
Golden, CO 80401

Call for information on other software and hardware products.

Circle No. 70 on Inquiry Card

we always had management encouragement. At the same time, it was certainly nothing like a development project. Our intent was to create a pleasant computing environment for ourselves, and our hope was that others liked it.

The Computing Science Research Center at Bell Laboratories to which Thompson and I belong studies three broad areas: theory; numerical analysis; and system, languages, and software. Although work for its own sake resulting, for example, in a paper in a learned journal, is not only tolerated but welcomed, there is strong though wonderfully subtle pressure to think about problems somehow relevant to our corporation. This has been so since I joined Bell Labs around 15 years ago, and it should not be surprising. the old Bell System may have seemed a sheltered monopoly, but research has always had to pay its way. Indeed, researchers love to find problems to work on; one of the advantages of doing research in a large company is the enormous range of the puzzles that turn up. For example, theorists may contribute to compiler design, or to LSI algorithms; numerical analysts study charge and current distribution in semiconductors; and, of course, software types like to design systems and write programs that people use. Thus, computer research at Bell Labs has always had considerable commitment to the world, and does not fear edicts commanding us to be practical.

For some of us, in fact, a principal frustration has been the inability to convince others that our research products can indeed be useful. Someone may invent a new application, write an illustrative program, and put it to use in our own lab. Many such demonstrations require further development and continuing support in order for the company to make best use of them. In the past, this use would have been exclusively inside the Bell System; more recently, there is the possibility of developing a product for direct sale.

For example, some years ago Mike Lesk developed an automated directory-assistance system [3]. The program had an online Bell Labs phone book, and was connected to a voice synthesizer on a telephone line with a tone decoder. One dialed the system, and keyed in a name and location code on the telephone's key pad; it spoke back the person's telephone number and office address (it didn't attempt to pronounce the name). In spite of the hashing through 12 buttons (which, for example, squashed "A", "B" and "C" together), it was acceptably accurate: it had to give up on around 5 percent of the tries. The program was a local hit and well-used. Unfortunately, we couldn't find anyone to take it over, even as a supported service within the company, let alone a public offering, and it was an excessive drain on our resources, so it was finally scrapped. (I chose this example not only because it is old enough not to exacerbate any current squabbles, but also because it is timely: the organization that publishes the company telephone directory recently asked

...enough said.

See us at:
UniForum
Dallas
Jan. 21 to 25
Booth 2959

Fulcrum Technologies Inc.
331 Cooper Street,
Ottawa, Canada
K2P 0G5
(613) 238-1761

FULCRUM

Circle No. 71 on Inquiry Card

us whether the system could be revived.)

Of course not every idea is worth developing or supporting. In any event, the world is changing: our ideas and advice are being sought much more avidly than before. This increase in influence has been going on for several years, partly because of the success of UNIX, but more recently, because of the dramatic alteration of the structure of our company.

AT&T divested its telephone operating companies at the beginning of 1984. There has been considerable public speculation about what this will mean for fundamental research at Bell Laboratories: one report in *Science* [2] is typical. One fear sometimes expressed is that basic research, in general, may languish because it yields insufficient short-term gains to the new, smaller AT&T. The public position of the company is reassuring: moreover, research management at Bell Labs seems to believe deeply, and argues persuasively, that the commitment of support to basic research is deep and will continue [1].

Fundamental research at Bell Labs in physics, chemistry, and mathematics may, indeed, not be threatened: nevertheless, the danger it might face, and the case against which it must be prepared to argue, is that of irrelevance to the goals of the company. Computer science research is different from these more traditional disciplines. Philosophically it differs from the physical sciences because it seeks not to discover, explain, or exploit the natural world, but instead to study the properties of machines of human creation. In this it is analogous to mathematics, and indeed the "science" part of computer science is, for the most part, mathematical in spirit. But an inevitable aspect of computer science is the creation of computer programs: objects that, though intangible, are subject to commercial exchange.

More than anything else, the greatest danger to good computer science research today may be *excessive* relevance. Evidence for the worldwide fascination with computers is everywhere, from the articles on the financial, and even the front pages of the newspapers, to the difficulties that even the most prestigious universities experience in finding and keeping faculty in computer science. The best professors, instead of teaching bright students, join start-up companies, and often discover that their brightest students have preceded them. Computer science is in the limelight, especially those aspects, such as systems, languages, and machine architecture, that may have immediate commercial applications. The attention is flattering, but it can work to the detriment of good research.

As the intensity of research in a particular area increases, so does the impulse to keep its results secret. This is true even in the university (Watson's account [12] of the discovery of the structure of DNA provides a well-known example), although in academia there is a strong counterpressure: unless one publishes, one never

UNICOMP
Technical Type

Typesetting Service

- UNIX*, Troff, Wizard
- Compugraphic 8400 Typesetter
- Quality appearance of books, proceedings, newsletters.
- Specializing in technical documents.
- Documents accepted via magnetic tape, phone lines, or paper.
- For information, samples, or estimates, call

505/662-EDIT (3348)

1580 Camino Redondo
Los Alamos, NM 87544

*Unix is a trademark of Bell Laboratories

Circle No. 73 on Inquiry Card

UNIX*/ C
$25,000 to $50,000
APPLICATIONS and SYSTEMS

National Placement Leader for UNIX/C PROF'Ls

500 OPENINGS NATIONWIDE

Call or send resume

J. ROGERS ASSOCIATES
Dept. C-R
123 Franklin Corner Rd.
Lawrenceville, N.J. 08648
(609) 896-8484
(800) 222-0596

Representing EOE Companies

*UNIX is a trademark of Bell Labs

Circle No. 74 on Inquiry Card

becomes known at all. In industry, a natural impulse of the establishment is to guard proprietary information. Researchers understand reasonable restrictions on what and when they publish, but many will become irritated and flee elsewhere, or start working in less delicate areas, if prevented from communicating their discoveries and inventions in suitable fashion. Research management at Bell Labs has traditionally been sensitive to maintaining a careful balance between company interest and the industrial equivalent of academic freedom. The entrance of AT&T into the computer industry will test, and perhaps strain, this balance.

Another danger is that commercial pressure of one sort or another will divert the attention of the best thinkers from real innovation to exploitation of the current fad, from prospecting to mining a known lode. These pressures manifest themselves not only in the disappearance of faculty into industry, but also in the conservatism that overtakes those with well-paying investments — intellectual or financial — in a given idea. Perhaps this effect explains why so few interesting software systems have come from the large computer companies: they are locked into the existing world. Even IBM, which supports a well-regarded and productive research establishment, has in recent years produced little to cause even a minor revolution in the way people think about computers. The working examples of important new systems seem to have come either from entrepreneurial efforts (VisiCalc is a good example) or from large companies, like Bell Labs and most especially Xerox, that were much involved with computers and could afford research into them, but did not regard them as their primary business.

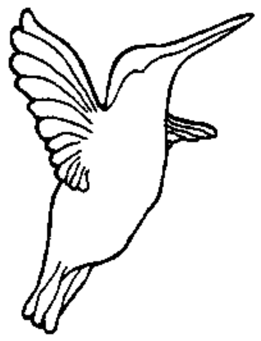
On the other hand, in smaller companies, even the most vigorous research support is highly dependent on market conditions. *The New York Times*, in an article describing Alan Kay's passage from Atari to Apple, notes the problem: "Mr. Kay...said that Atari's laboratories had lost some of the atmosphere of innovation that once attracted some of the finest talent in the industry. "When I left last month it was clear that they would be putting their efforts in the short term," he said..."I guess the tree of research must from time to time be refreshed with the blood of bean counters."[9]

Partly because they are new and still immature, and partly because they are a creation of the intellect, the arts and sciences of software abridge the chain, usually in physics and engineering, between fundamental discoveries, advanced development, and application. The inventors of ideas about how software should work usually find it necessary to build demonstration systems. For large systems, and for revolutionary ideas, much time is required. It can be said that UNIX was written in the '70s to distill the best systems ideas of the '60s, and became the commonplace of the '80s. The work at Xerox PARC on personal computers, bitmap graphics, and programming environments [10] shows a similar progression, starting and coming to fruition a few years later. Time, and a commitment to the long-term value of the research, are needed on the part of both the researchers and their management.

Bell Labs has provided this commitment and more: a rare and uniquely stimulating research environment for my colleagues and me. As it enters what company publications call "the new competitive era", its managers and workers will do well to keep in mind how, and under what conditions, the UNIX system succeeded. If we can keep alive enough openness to new ideas, enough freedom of communication, enough patience to allow the novel to prosper, it will remain possible for a future Ken Thompson to find a little-used CRAY/I computer and fashion a system as creative, and as influential, as UNIX.

REFERENCES

1. Bell Labs: New order augurs well. *Nature 305*, 5933 (Sept. 29, 1983).
2. Bell Labs on the brink. *Science 221* (Sept. 23, 1983).
3. Lesk, M.E. User-activated BTL directory assistance. Bell Laboratories internal memorandum (1972).
4. Norman, D.A. The truth about UNIX. *Datamation 27* (1981).
5. Organick, E.I. The Multics System *MIT Press* Cambridge, M.A. (1972)
6. Ritchie, D. M. UNIX time-sharing system: A Restrospective. *Bell Syst. Tech. J.* 57.6 (1978) (1947-1969.
7. Ritchie, D. M. The Evolution of the UNIX time-sharing system. In Language Design and Programming Methodology. Jeffrey M. Tobias, Ed. *Springer-Verlag.* New York. (1980).
8. Ritchie, D. M. and Thompson, K. The UNIX time-sharing system. *Commun. ACM* 17.7 (July 1974) 365-375.
9. Sanger, D. E. Key Atari scientist switches to Apple. *The New York Times* 133,46,033 (May 3, 1984).
10. Thacker, C. P. et al. Alto, a personal computer. *Xerox PARC Technical Report* CSL-79-11.
11. Thompson, K. UNIX time-sharing system: UNIX implementation. *Bell Syst. Tech J.* 57,6 (1978), 1931-1946.
12. Watson, J. D. The Double Helix: A Personal Account of the Discovery of the Structure of DNA. *Atheneum Publishers.* New York (1968).



UNIX UNLEASHED

The university role of research in maintaining system vitality

by Marshall Kirk McKusick

Since the AT&T divestiture, UNIX has become the focus of a massive marketing effort. To succeed, this effort must convince potential customers that the product is supported, that future versions will continue to be developed, and that these versions will be upwardly compatible with all past applications.

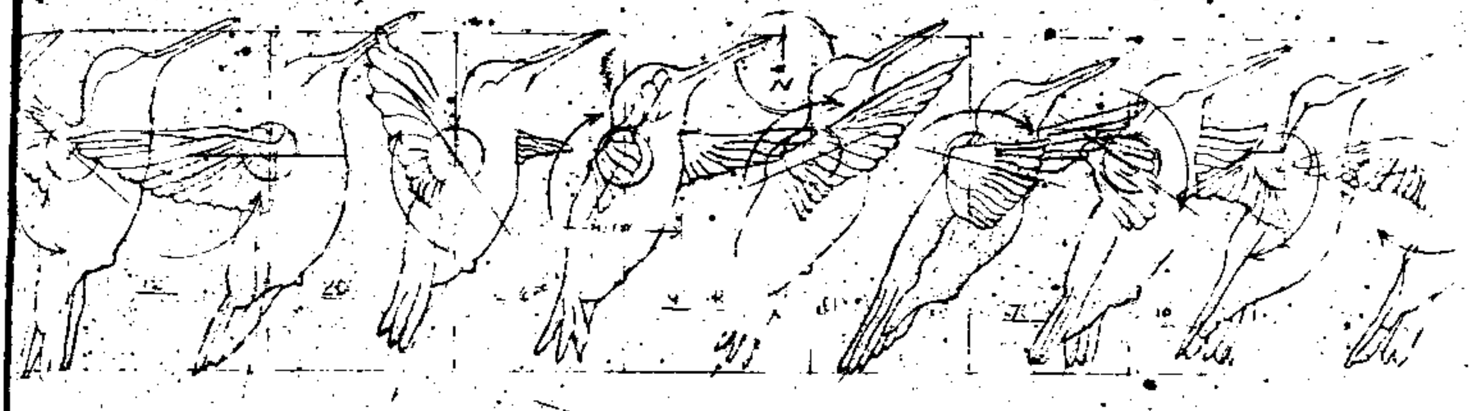
AT&T's size alone ensures that it will be around in years to come. The fact that the company has allocated a growing number of research, development, and support resources to UNIX over the past 10 years provides an assurance of its commitment. Meanwhile, its massive advertising campaign for System V, its presence on the /usr/group UNIX standards committee, and the publication of the *System V Interface Definition* testify to the company's intention to remain compatible with past systems.

Although repeal of the law of entropy is a necessary step along the road to a viable commercial product, this runs counter to orderly system evolution. Be that as it may, AT&T's major UNIX commercialization effort has succeeded in making the system available to a much broader audience than was previously possible.

The freezing of what previously had been an ever-changing UNIX interface represented a major departure from the pattern that the small but highly skilled UNIX community had come to expect. Most early users had accounts at sites that had the source to the programs they ran. Thus, as the system interface evolved to reflect more current technology, software could be changed to keep pace. Users simply updated their programs to account for the new interface, recompiled them, and continued to use them as before. Although this required a large effort, it allowed the system—and the tools that ran on it—to reflect changes in software technology.

At the forefront of the technological wave was AT&T's own Bell

Illustration by M. Kathryn Thompson



Laboratories. It was there that the UNIX system was born and nurtured, and it was there that its evolution was controlled—up through the release of the 7th Edition. Universities also were involved with the system almost from its inception. The University of California at Berkeley was one of the first participants, playing host to several researchers on sabbatical from the Labs. This cooperation typified the harmony that was characteristic of the early UNIX community. Work that was contributed to the Labs by different members of the community helped produce a rapidly expanding set of tools and facilities.

With the release of the 7th Edition, though, the usefulness of UNIX already had been clearly established, and other organizations within AT&T began to handle the public releases of the system. These groups took far less input from the community as they began to freeze the system interface in preparation for entry into the commercial marketplace.

As the research community continued to modify the UNIX system, it found that it needed an organization that could produce releases. Berkeley quickly stepped into the role. Prior to the final public release of UNIX from the Labs, Berkeley's work had been focused on the development of tools designed to be added to existing UNIX systems. After the AT&T freeze, though, a group of researchers at the university found that they could easily expand their role to include the coalescing function previously provided by the Labs. Out of this came the first full Berkeley distribution of UNIX (3.0BSD), complete with virtual memory—a first for UNIX users. The idea was so successful that System V eventually adopted it six years later.

At the same time that AT&T was beginning to put the brakes on further change in UNIX, local area networks and bitmapped workstations were just beginning to emerge from Xerox PARC and other research centers. Users in the academic and research community realized that there were no production-quality operating systems capable of making use of such hardware. They also saw that networking unquestionably would be an indispensable facility in future systems research. Though it was not clear that UNIX was the correct base on which to build a networked system, it was clear that UNIX offered the most expedient means by which to build such a system.

This posed the Berkeley group with an interesting challenge: how to meet the needs of the community of users without adding needless complexity to existing applications. Their efforts were aided by the presence of a large and diverse local group of users who were teaching introductory programming, typesetting documents, developing software systems, and trying to build huge Lisp-based systems capable of solving differential equations.

Although repeal of the law of entropy is a necessary step along the road to a viable commercial product, this runs counter to orderly system evolution.

In addition, they were able to discuss current problems and hash out potential solutions at semi-annual technical conferences run by the Usenix organization.

The assistance of a steering committee composed of academics, commercial vendors, DARPA researchers, and people from the Labs made it possible for the architecture of a networking-based UNIX system to be developed. By keeping with the UNIX tradition of integrating work done by others in preference to writing everything from scratch, 4.2BSD was released less than two years later.

MECHANISMS FOR PRODUCING ORDERLY EVOLUTION

Software systems have often been compared to biological organisms. They are born and go through a period of innocence akin to childhood. They then go through another burst of growth that takes them into the adult world where they are expected to give up their childish ways. As people come to rely on these systems, crashes and the loss of data cease to be considered acceptable behavior. As the software grows into middle age, it gains a wider exposure that allows it to be used in more critical and demanding applications. During this period, a system reaches the most productive part of its life. As it ages, though, it becomes less able to adapt to changing times. Eventually, it must retire so that younger, more agile systems can move in to take its place.

UNIX was born in a lean and mean era; it was designed for processors that ran at fractional MIPS, with memories smaller than 65 kilobytes, and 10-character-per-second printing terminals that made interaction agonizingly slow. Given such a starting point, it is a tribute to the designers of UNIX that the system can now be found running on multi-MIP processors, with megabytes of memory, and multiwindow bitmapped displays. There are several reasons why UNIX has managed to stretch its biological limits to this degree.

The single most important structural reason is that UNIX was not written in assembly language. Equally important is the fact that it was not written in a complex high-level language that could be compiled only on a large computer system. UNIX has succeeded largely because the C language itself was just high-level enough to allow it to be easily compiled for a wide range of computer hardware, without being so complex or restrictive that systems programmers had to revert to assembly language to get reasonable efficiency or functionality. Although the success of UNIX does not stem solely from the fact that it was written in a high-level language, the use of C was a critical first step.

The second decision essential to the extended evolution of UNIX resulted in the system's early release from Bell Labs to other research environments in source form. By providing source, the system's founders ensured that other organizations would not only be able to use the system, but also tinker with its inner workings. The ease with which new ideas could be adapted into UNIX always has been key to the changes that have been made to it. Whenever a new system would come along that tried to upstage UNIX, someone would dissect the newcomer and clone its central ideas into UNIX. The unique ability to use a small, comprehensible system, written in a high-level language, in an environment swimming in new ideas led to a UNIX system that evolved far beyond its humble beginnings.

Note, though, that the path of evolution is littered with broken carcasses. While gene mutation is critical to the advance of the species, only one in 100 produces a useful feature; the rest result in needless or detrimental changes. The mere existence of an environment for mutation is not enough—some organization must bear responsibility for brutally pruning the weak or useless ideas. Here again UNIX was unique. Unlike other projects beset by competing groups jealously guarding their work from one another, UNIX thrived in an open and cooperative community willing to channel its ideas through a central clearinghouse, in spite of the reputation that clearinghouse had for selective technical scrutiny.

Here one must distinguish between the selection process provided by research and commercial organizations. Research organizations can base pruning considerations strictly on the coherence of a system. They need not concern themselves with how changes might affect past variants of the system. Commercial organizations, though, must ensure that changes will not affect programs built to tie in with an old interface. Thus, paging might be a great idea, but it could cause problems for old software that depends on the execution predic-

WHEN SERIOUS PROGRAMMING IS YOUR BUSINESS...

The Concurrent Euclid language for systems programming provides the best in efficiency, portability, reliability, and maintainability

Compilers running on UNIX/VAX, UNIX/11, VMS/VAX, with code generated for MC68000, MC6809, NS32000, 8086/8088 PDP-11, and soon running on IBM-PC

CONCURRENT EUCLID

Compiler: CSRI Distribution Mgr.
Sandford Fleming Bldg 2002
10 King's College Road
Toronto, Canada M5S 1A4
Tel: (416) 978-6985

Book:
CONCURRENT EUCLID,
THE UNIX SYSTEM AND TUNIS
Available from:
Addison-Wesley Publishing
Company, Reading, MA. 01867
Tel: (617) 944-3700

CONCURRENT
E U C L I D

Circle No. 23 on Inquiry Card

tability of a swap-based system, making it impossible for paging to replace swapping; as a result, the complexity of supporting both schemes must be maintained. As the system becomes more complex, its evolutionary paths will become increasingly restricted.

This is not surprising. No software system can last forever; revolution is as necessary in the software world as storms are in the physical world. The old guard must eventually give way to new blood. The fact that UNIX was provided with a less restrictive growth path in the research environment during its critical adolescent period has probably doubled its life expectancy. Ultimately, the commercial system will have to get beyond its slow-printing terminal orientation and adopt the new technology, lest it be superseded by an onslaught of systems capable of supporting bitmapped displays.

Evolutionary restrictions are such that the facilities of the commercial system lag anywhere from five years to an infinite amount of time behind the research systems. In an effort to provide more modern facilities, many manufacturers have started marketing the 4.2BSD research system in order to sell into more sophisticated technical markets. Over the long term, it is reasonable to expect that the most useful functionality of the research systems will be grafted into the commercial version, while the research version will be extended to provide as much of the commercial version's interface as possible.

THE FUTURE OF UNIX

UNIX is currently in its middle age. The commercial version of UNIX has been widely adopted because it provides better functionality than any other PC operating system on the market. However, users have already discovered that isolated PCs are far less useful than PCs networked together in a way allowing for remote logins, file transfer, distribution of a pipeline across multiple machines, and other distributed computing applications. The research version of UNIX demonstrated in 1982 that UNIX could evolve to accommodate networking. But the commercial version continues to offer little more in the way of network functionality than file transfer and batch-style remote execution. This probably will be remedied, despite the commercial constraints, within the next couple of years.

UNIX thrived in an open and cooperative community willing to channel its ideas through a central clearinghouse.

The current trend in systems research is to provide a variety of environments. One example is the text processing environment offered by Apple's Macintosh. The UNIX user must learn an extensive toolset used in conjunction with an unforgiving shell before diving into document production. The user brave enough to tackle this task needs first to learn how to visualize the output that the system's baroque typesetting language will produce. By contrast, the Macintosh provides a menu-driven toolset interface and a text-processing interface that allow the user to see exactly how the output will appear.

Similarly, the trend in the program development arena is toward object-based environments in which the system can maintain object dependencies, thus keeping the project's binaries, libraries, and documentation up to date as program development proceeds.

Another major research topic is how to build systems using several tightly coupled processors. Ideally, processors can be used together to provide the illusion of a single larger machine. True multiprocessor support requires that the UNIX kernel run simultaneously on all processors. The system depends on only one kernel being in operation, synchronizing through a large global memory and the selective blocking of interrupts. This synchronization structure does not lend itself to running the kernel simultaneously on more than one processor. A message-based synchronization structure lends itself much more readily to a kernel running in a distributed environment. Though several groups have modified the existing kernel to run in a multiprocessor environment, the added complexity makes further evolution difficult.

The challenge facing UNIX researchers today is the need to add functionality capable of supporting environments while they produce kernel modifications that will allow UNIX to run in a multiprocessor environment. Since system complexity increases much faster than the size of its code, the current structure requiring a monolithic kernel may become untenable. However, growing familiarity with the UNIX interface argues strongly in favor of maintaining it.

Some researchers now believe that the revolution will come from below without end users being any the wiser. They think that a new system will be developed that consists of a small kernel-kernel providing only the lowest level of message passing, hardware scheduling, and virtual memory management. The current UNIX kernel will be broken into several server processes on top of the kernel-kernel. Sample services from existing UNIX systems include a file system server, a TCP/IP network server, and a terminal line server. New services might include an Ada development server or a document preparation environment server. Users familiar with existing UNIX systems could continue to run on top of the UNIX file system server, while users wishing to work within an Ada environment could work with an object-based file system through an object-oriented Ada database server. Synchronization between the servers could be offered by the message interface provided by the kernel-kernel. By restructuring synchronization through messages rather than by using a global-shared memory, servers could be decoupled from the particular processor they run on. The servers would be able to interact equally well whether they were on the same processor, two processors running together, or on two processors separated by miles of network cable.

This scheme would reduce the complexity of the system to a manageable level. Rather than having to deal with a single mountain of interrelated code, each server could be treated as a smaller independent module. This will mean that users will no longer be required to run a huge kernel providing many features they neither need nor want—meaning that the system will consume fewer of the resources they *do* want.

Systems such as this will not come without a cost. Message-based synchronization requires more CPU cycles than the use of semaphores in a shared memory. As a result, such systems do not compete well in shops where timesharing computers are used by so many people that nearly all the system's services must be active on the same machine. However, users are quickly moving toward having personal networked workstations in which CPU cycles are cheap but disk and memory are expensive. This will provide a setting in which the performance problems of a message-based system become unnoticeable and the benefits of rapid prototyping and new integrated facilities usher in a new era of growth.

Dr. McKusick is involved in the development of Berkeley UNIX as a Research Computer Scientist for the Computer Systems Research Group at the University of California. While a graduate student, he implemented the fast file system distributed on 4.2BSD and worked on the Berkeley Pascal system. ■

The views of the future are gleaned from discussions with Bill Joy and Rick Rashid. Acknowledgements to Charles Darwin and Thomas Jefferson are also in order. I thank those people whose accomplishments, though unmentioned, contributed much to the system's vitality. For the last six years, I have been supported by the National Science Foundation under grant MCS80-05144, and the Defense Advance Research Projects Agency (DoD) under ARPA Order No. 4031 monitored by Naval Electronic System Command under Contract No. N00039-82-C-0235.

MORNING STAR TECHNOLOGIES

UNIX* COMMUNICATIONS

X.25 • HASP • SNA3270 • SNA3770

Drop-in communication systems for MULTIBUS* based computers. Offload the CPU intensive process of communication with the HORIZON™ Series of boards from MORNING STAR. Complete systems include your choice of hardware and software combinations to custom fit your data communication needs. Available for: **Sun Microsystems, Masscomp, Pyramid, Heurikon, Plexus, NCR Tower, Sperry 5000, Celerity and more.**

Call today for more information

800 - 558 - 7827

Morning Star Technologies, Inc.

1760 Zollinger Road, Columbus, Ohio 43221

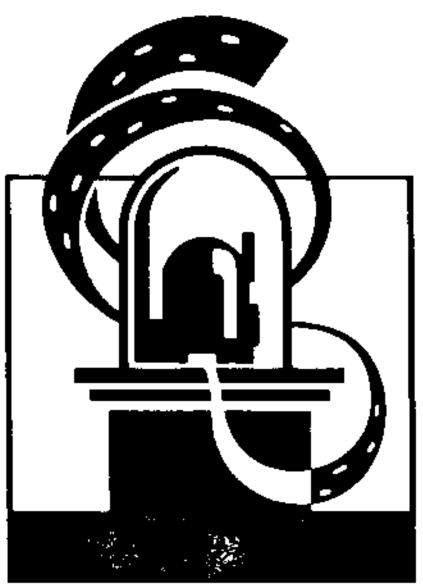
In Ohio call (614) 451-1883 TWX - 510 - 600 - 3272

*UNIX is a Trademark of AT&T Bell Labs • MULTIBUS is a Trademark of Intel Corp

Circle No. 22 on Inquiry Card

BSD'S BIG ADVENTURE: THE BAD BERKELEY-TO-BOSTON CONNECTION

BY DAVID CHANDLER



The cover of last April's *UNIX REVIEW* depicted a garden labyrinth of tall green hedges labeled "The Networking Maze". Our intention was to illustrate the confusion many UNIX users have experienced when grappling with networking. A similar device would serve well in conjunction with the tale of bureaucratic intrigue that follows. This, too, is a story of networking, but the implications are much broader—affecting the very existence of the Berkeley Software Distribution of UNIX.

In this particular labyrinth, the principal players are the Defense Advanced Research Projects Agency (DARPA) and two of the major beneficiaries of its budgetary bounty—Bolt Baranek and Newman (BBN, the Cambridge, MA-based communications software house) and the Computer Systems Research Group at the University of California at Berkeley. The three have generated a fair amount of drama over the course of an uneasy five-year relationship that has spawned the development of 4.2 and 4.3BSD. During recent weeks, the relationship nearly foundered as tensions between Berkeley and BBN staffers built up to the snapping point.

Christmas came early, though, as DARPA announced a truce on December 18 that should maintain the relationship for a while longer—long enough, at least, to produce the much-anticipated release of 4.3BSD sometime in February.

The details of how things went sour would do justice by Franz Kafka, but to sort them out requires that we go back five years to the time before the Berkeley-to-Boston bureaucratic labyrinth had been erected—back to the time when DARPA had just decided to bring both Berkeley and BBN under contract to develop a networking scheme for a single operating system adaptable to the needs of all DARPA contractors.

Under the terms of the contract, BBN was to write the Transmission Control Protocol/Internet Protocol (TCP/IP) for the DoD's ARPANET standard, and Berkeley was to develop sockets and network interfaces. This delineation of duties was in accordance with the International Standards Organization Reference Model for Open Systems Interconnection (ISO-OSI), the seven-layer specification for network design. One could apply the analogy of a sandwich: BBN's work on TCP/IP was comparable to the filling—the network and transport layers (OSI layers three and four); and Berkeley's work on sockets and interfaces was the bread—the surrounding physical, data link, session, and presentation layers.

In keeping with DARPA's bidding, BBN developed the original implementation of TCP/IP, and provided it to Berkeley. From that time forward, the two organizations were supposed to work together to integrate the protocol into the whole of BSD's networking code. Upon reviewing the BBN code, though, researchers at Berkeley took issue with certain of its elements. In fact, they thought so little of BBN's effort that they took the liberty of making significant modifications (much as they already had done with AT&T's UNIX code). Thus was born a second version of TCP/IP.

It was at this point, as Holmes would say, that the plot thickened. BBN was not, if you will, *pleased* that its version had been altered for purposes other than integration. Berkeley researchers, of course, believed that their efforts had yielded a superior version. Meanwhile, DARPA, the distressed matchmaker/general contractor, attempted to mediate.

Despite heavy lobbying from BBN, the Berkeley version of TCP/IP became a *de facto* standard when it was included as part of the official 4.2BSD release in September, 1982. Prior to this watershed event, the differences between the two versions had been significant but not fundamental. This soon was to change. Though DARPA continued to encourage Berkeley to use the BBN code, splits between the two implementations became increasingly evident.

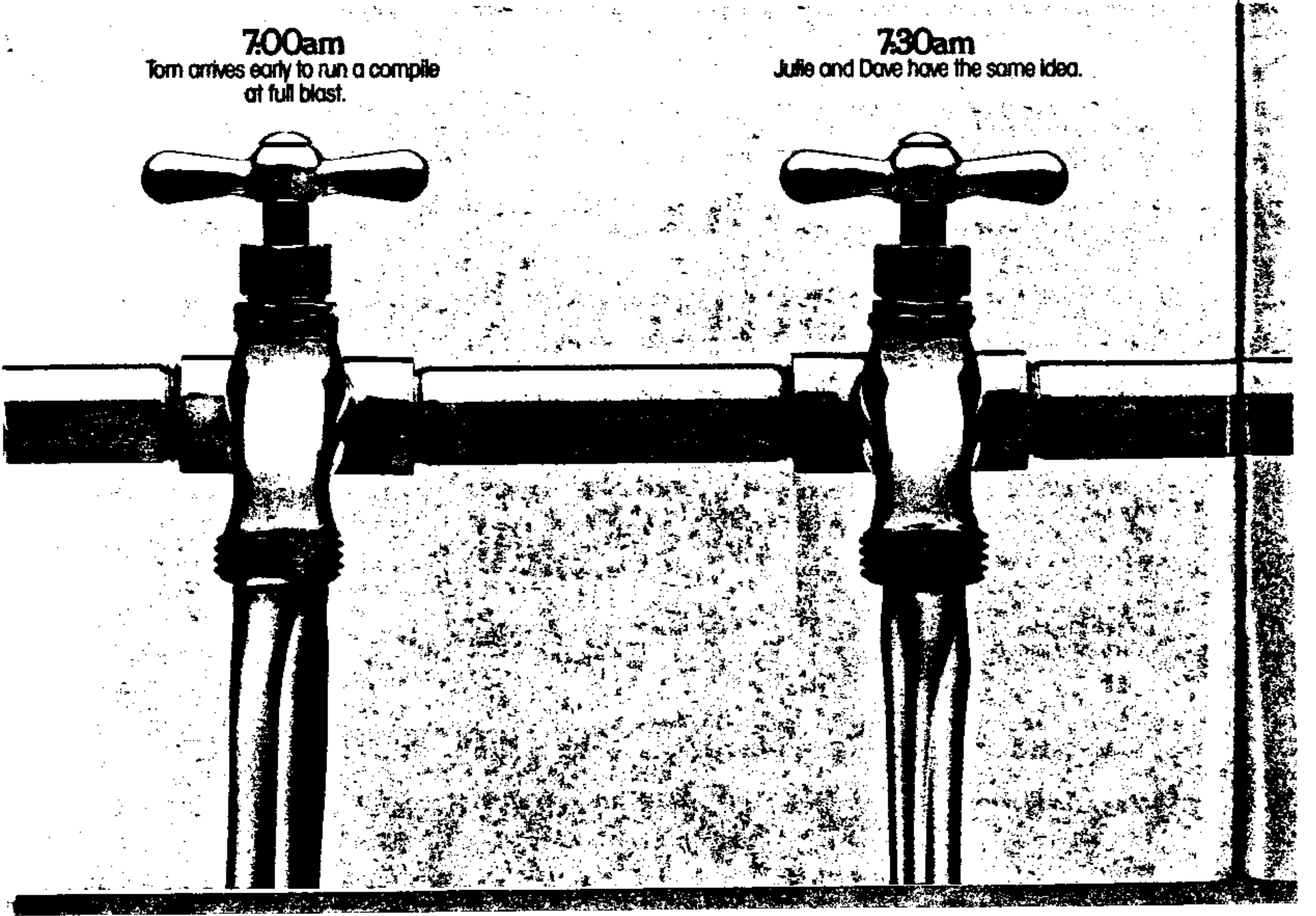
Colonel Robert Baker, R & D Program Manager for DARPA's Distributed Computing Program—which includes the Berkeley UNIX networking project—recalled what happened next: "There were some things oriented toward local network performance that were put into the Berkeley version, and some oriented toward internetwork performance that were put into the BBN version. BBN integrated its version of the code with 4.2 such that it was possible for someone who had 4.2 to replace the Berkeley networking code in their kernel with the code from BBN. . . ."

As Berkeley UNIX has evolved over the past two or so years toward the 4.3 release, both versions of the TCP/IP code have remained available. DARPA, though, reached the decision some time ago that it would allow only one version to be integrated into 4.3. As Baker recalled, "We decided that we had dealt with two versions for long enough. You know, the situation in the past has been that a number of people have gotten the distribution from Berkeley and then have replaced the networking code with the code from BBN because it offered more functionality in the area of internetwork robustness. Still, we decided that we didn't want to continue supporting both versions."

This is a story of networking, but the implications affect the very existence of the Berkeley Software Distribution of UNIX.

At last June's Usenix meeting in Portland, Berkeley researchers announced that 4.3 would be released officially within two months. But that was before the networking controversy came to a head. One standard TCP/IP implementation had to be chosen, and the decision clearly was not going to be easy; while on the one hand DARPA had paid several million dollars for the BBN code and had supported it right from the start, the Berkeley code had been run in academic and commercial use under 4.2 for almost three years, and—despite DARPA's assertions—was considered by many to be superior. More crucial yet was the fact that patience on all sides was wearing thin, and the Berkeley researchers responsible for BSD itself had intimated that they would not tolerate the bureaucratic sojourn much longer.

A compromise of sorts was reached. "What we did", Baker recalled, "was send out for beta testing [last August] with the two versions of the networking code. . . . We also had some people [in the UNIX community] make an independent evaluation of the two versions, looking both at functionality and the software itself. The conclusion was that the BBN code offers some functionality that the Berkeley code doesn't, and that the reverse is also true. Either


7:00am
Tom arrives early to run a compile at full blast.
7:30am
Julie and Dave have the same idea.

version could be used as the base for 4.3 and still provide a performance advantage over what's now in 4.2."

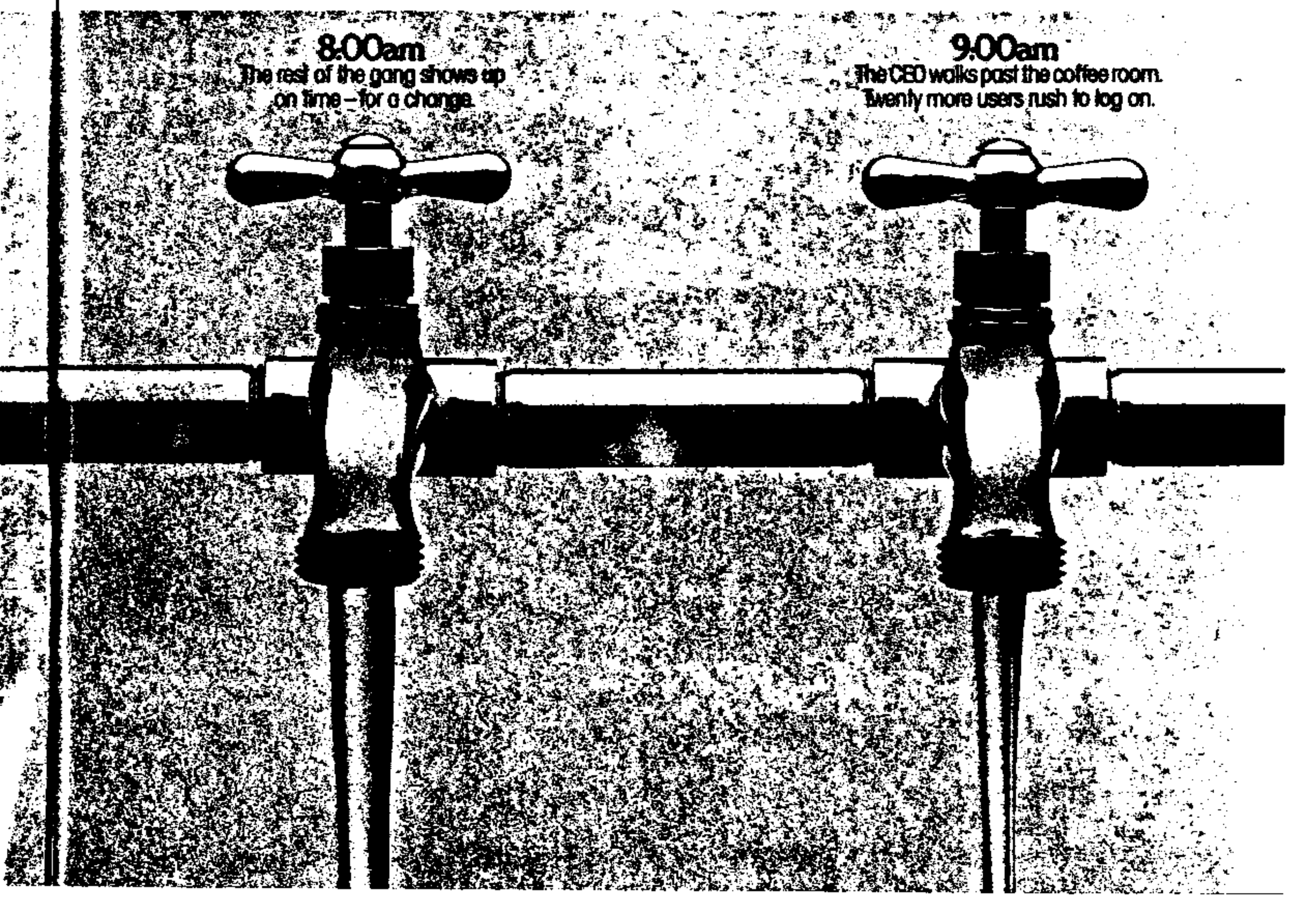
Kirk McKusick, a research computer scientist at Berkeley and one of the chief developers of 4.3, lobbied strongly along with his colleagues for the Berkeley version of TCP/IP. According to McKusick, of 35 beta sites testing 4.3BSD, 33 chose to use Berkeley TCP/IP—and one of the other two was planning to switch because of repeated system crashes with the BBN code. Included among the 4.3 beta test sites running Berkeley's TCP/IP were a number of major commercial vendors who already had begun to base products on it, and who had expressed an interest in releasing those packages.

The DARPA evaluation went on for a number of months. On December 18, though—some four months after the anticipated release date of 4.3—DARPA sent a formal letter to Berkeley indicating it had decided that the 4.3 release should include Berkeley's version of TCP/IP. The decision, DARPA said, had been based both on technical and political evaluations.

"There's been a much greater degree of testing and experience with the Berkeley code", DARPA's Baker explained. "Of course, it's the code that's been included in the past distributions. So, on the basis of the evaluations we have done, we decided to use the Berkeley code as the base for the networking code distributed in 4.3, and to incorporate into it whatever additional functionality we find we need from the BBN code. . . .

"Actually the Berkeley code in its current form already has incorporated a number of things from the BBN code. That's happened as part of the process of putting together 4.3 and running it through beta testing."

A member of the BBN development team chose not to comment on what he called "a long and involved story". Berkeley researchers, though, understandably were pleased—not only because the Berkeley version of TCP/IP had been selected, but because the matter finally had been resolved. McKusick said the major remaining hurdle for 4.3 relates to documentation. In the aftermath of DARPA's decision, it will be necessary to comb through some 10,000 pages to ensure that all TCP/IP references conform. Barring


8:00am
The rest of the gang shows up on time—for a change.
9:00am
The CEO walks past the coffee room. Twenty more users rush to log on.

any unforeseen complications, 4.3 is expected to be released officially in February.

New Features

Considering what is meant by the term "new", let's distinguish between those things that have been revised from those that have been created from scratch. The forthcoming 4.3BSD fits under the heading "revised". *4.4BSD*, however, likely will contain things that never before have been released.

In commenting on the general practice of generating new releases, McKusick explained that, "We basically tend to alternate between making major functional changes and tune-ups."

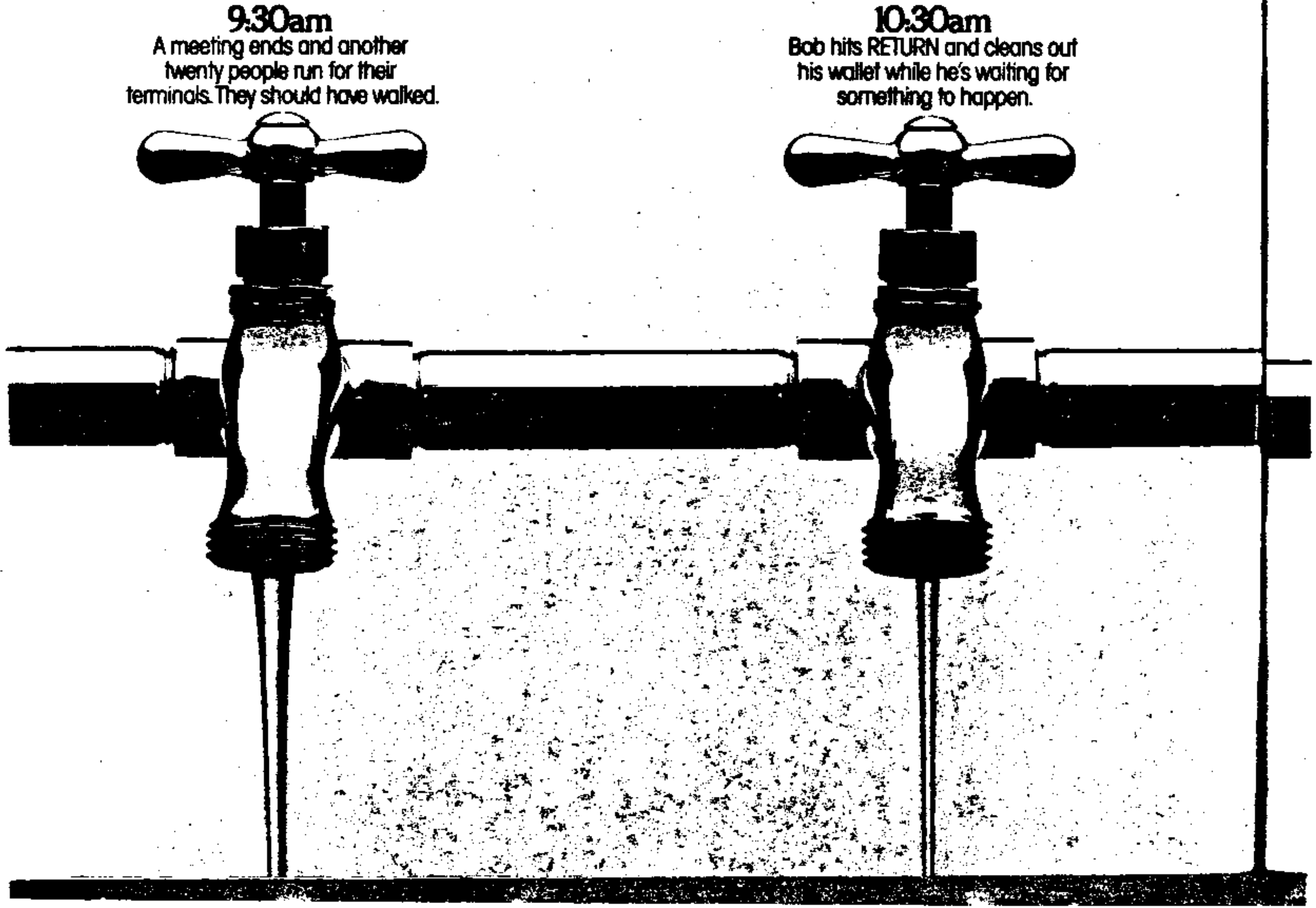
"4.0 came out with several enhancements—job control, auto-configuration, and all that—but it didn't run that fast. It ended up spending about 20 percent of its time doing context switching, whereas VMS, for example, was spending about 4 percent. . . . But then 4.1 came out and offered snappier performance.

"Next was 4.2. We were trying to put in many new things and we wanted to do it with the minimal amount of code. Admittedly, 4.2 ran about 20 percent slower than 4.1. . . Many of the internal structures of the kernel were used much more heavily in 4.2 than in 4.1, so a facility that was adequate in 4.1 became less than adequate in 4.2."

With 4.3, "we started to fix the things that had turned up. We spent most of our time analyzing the system and figuring out where and why it was slow. That told us what we needed to tweak."

So what is it that we can expect of commercial systems bearing a "4.3" label? The best source of publicly-available details is a paper written by McKusick and Mike Karels of UC Berkeley, and Sam Leffler of Lucasfilm (late of UCB) for the Summer '85 Usenix Conference. A copy is contained in the conference's proceedings. Those who seek technical details are hereby referred to that paper, since we have space enough here only to discuss the general distinctions, of which there are three.

Performance Improvements. This actually can be broken down into the two sub-classes of performance optimization for the general timesharing environment—changes to the kernel and changes to the


9:30am
A meeting ends and another twenty people run for their terminals. They should have walked.
10:30am
Bob hits RETURN and cleans out his wallet while he's waiting for something to happen.

system libraries and utilities. The changes to the kernel involve—among other things—name caching, intelligent auto-siloing, process table management, scheduling, clock handling, file system management, networking, the exec command, context switching, *setjmp* and *longjmp* calls, and various compensations for the lagging compiler technology of the system. Improvements to libraries and utilities, meanwhile, have resulted from work done on hashed databases, buffered I/O, mail system software, network servers, the C runtime library, and the C shell.

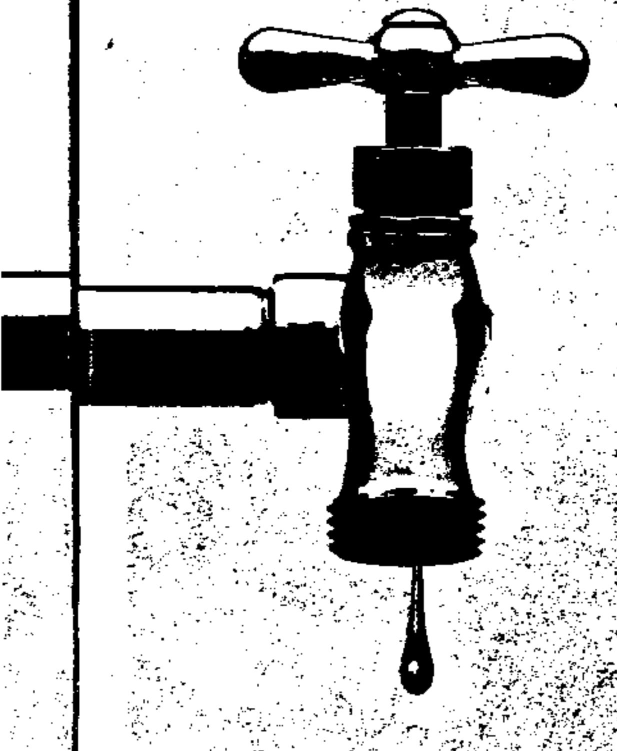
Functional Extensions. As with the performance improvements made for the system, functional extensions have been divided into extensions to a) the kernel and b) libraries and utilities. On the kernel level, as McKusick et al. stated in the Usenix paper, "A significant effort went into improving the networking part of the kernel. The work consisted of fixing bugs, tuning the algorithms, and revamping the lowest levels of the system to better handle heterogeneous network topologies." With regard to extensions to libraries and utilities, the developers wrote, "Most of the changes to the utilities and libraries have been made to allow them to handle a more general set of problems, or to handle the same set of problems more quickly."

Tightened Security. Certain changes also were made to both the kernel and the system's utilities in order to enhance security. These changes have been discussed in modest detail elsewhere, but as the Usenix paper noted, "Since we do not wish to encourage rampant system cracking", we prefer not to make the details public here.

Insights with Foresight

Carter George mentions in this month's feature article on communications futuristics that one does well when discussing the future to avoid sounding like Jeanne Dixon. There are, however, certain trends in computing that lend themselves to extrapolation, and Berkeley UNIX not only relates to many of these trends, but often helps to shape them. BSD UNIX developer McKusick holds a seasoned perspective on what may lie on the other side of the horizon, and is in a position to influence decisions at Berkeley. Hence, we asked him to

11:00am
With response time this slow, everyone may as well go to lunch.



TOO MANY USERS CAN SLOW THINGS DOWN TO A TRICKLE.

Most UNIX™-based computers can really pour it on. When you're the only one logged in.

But as soon as the rest of the gang shows up, all that performance goes right down the drain.

That's when you need The Practical Parallel™ from Sequent.

Because it's a true parallel computer, The Practical Parallel gives you more consistent response than any single-processor system. At 8 MIPs, it has enough performance to keep over a hundred users running full blast.

But unlike other 8-MIP computers, you can get ours without getting soaked. Our systems start at well under a hundred grand.

Our price/performance is four times better than a VAX 8600 and two times better than a Pyramid 98X. And it's a fraction of the cost of getting everybody a workstation.

Naturally, it supports UNIX 4.2, System V, PC networking, and a flood of other options. Best of all, you can add processors quickly, inexpensively, and easily as your needs grow.

If you don't want bad response time to hang you out to dry, call us at 800/854-0428. Or write to Sequent, 15450 S.W. Koll Parkway, Beaverton, Oregon 97006-6063 and ask about The Practical Parallel.

It's the system that never runs hot and cold.

SEQUENT
The Practical Parallel™

See Us at UniForum booth #1562.

The Practical Parallel is a trademark of Sequent Computer Systems, Inc. UNIX is a trademark of AT&T Bell Laboratories, Inc.

Circle No. 8 on Inquiry Card

expound on what he sees in BSD's future:

Remote File Systems. "There are three major directions in which we are going", McKusick noted. "The first is remote file systems—something akin to Sun's NFS. Many of these systems have been done, and there is this continuum: the developer either can get them to run really fast with terrible semantics, or can get them semantically correct—which, for example, AT&T's Remote File System does—but suffer with abysmal throughput (like 20 or 30 KB per second) . . . . We're not willing to take that kind of throughput just because it's remote, so we might be willing to whittle away at some of the UNIX semantics. It's not exactly clear yet what that's going to mean, but we'll try to restrict our changes to things on which we don't think a lot of programs depend."

"Most of the changes have been made to allow utilities and libraries to handle a more general set of problems."

Virtual memory. "The second thing we're looking at is virtual memory", McKusick explained. "We basically feel that the next iteration of virtual memory needs to address a whole different set of concerns, which can be summarized by observing that memory is getting cheap and large very quickly. Therefore, what we want to do with physical memory is significantly different than what we've done in the past.

"Our question basically becomes: how do you design a 'virtual memory' system, given that you have lots of memory? We'll try to come up with a design that accomplishes that. At the same time we're going to throw in various things that come along for free—things like *copy-on-write, lightweight processes,* and *shared memory.*"

When asked if Berkeley would include a shared memory design similar to the one included in System V, McKusick responded, "It depends: if we choose to implement it in the way AT&T has done in System V, then we can make it easy for people to pick up. If we don't, it will be less easy. Whenever AT&T has defined a facility before we do,

RELIANT™

Solves information retrieval problems, enhances UNIX* system productivity!

A powerful, terminal-independent application development system, Reliant helps you help yourself. You develop and install new applications easily and in a fraction of the time that would be needed without Reliant. Reliant also makes you less dependent on expensive application programmers.

Reliant consists of four distinct but interrelated parts: a Data Definition System to rapidly define data to be entered; a General Data Entry Layer to enter, revise and delete data without error; a Report Generator to quickly specify the data to be retrieved and in what form; and a Document Generator to create the form of data presentation.

Best of all, because it's software from Daystrom, you get UPDATES FREE, for LIFE!

To try Reliant in your own environment, send us a $100.00 handling fee. We'll rush you the latest manuals and software to try for 30 days. If you agree it really is the answer to your information retrieval needs, we'll credit the $100.00 to the purchase price.

Of course, the best of software is worthless if you don't have the proper hardware. Daystrom solves that problem, too, with our "M-Family" of computers — the best power/performance ratio at the least cost in the industry. Choose from a variety of components to custom configure your system.

To find out how much help we can really be, contact Daystrom and we'll send you our latest catalog.

Making computers work the way you always thought they should.℠

DAYSTROM
11206 Euclid Avenue
Cleveland, Ohio 44106
216-421-8737

*UNIX is a registered trademark of AT&T Bell Laboratories
Reliant is currently available for Dec Vax, PDP-11, Integrated Solutions and Sun Microsystems.

Circle No. 9 on Inquiry Card

we have tried to implement a compatible interface or a broader set of primitives from which the interface could be built. We deviate from this policy only when we feel that there is a significant technical advantage to an incompatible design."

Stackable Line Disciplines. This is a third likely area for future Berkeley UNIX development. "System V calls it *streams*", McKusick explained. Does this mean that Berkeley is planning to adopt AT&T's *streams* technology? Conceptually, perhaps, but literally. . .no. McKusick elaborated: "There are two parts to it—the mechanism and the content. The mechanism is the ability to plug these things together; the content is what one puts inside the things that are plugged together. *Streams*—or stackable line disciplines—provides this connecting ability but says nothing about what's inside. AT&T—as far as I can tell— [has it structured so that] the pieces will go together just like railroad cars—you can connect them together, but what you load inside can differ from boxcar to boxcar. The things that we will provide inside the boxcars will be based on what we currently have in BSD, while AT&T will load in things based on its view of how networking should be. AT&T's base primitives for networking are virtual circuits, whereas ours are datagrams."

Would there then be compatibility between the Berkeley and AT&T disciplines? "One should be able to pick boxes up out of System V and with a very small amount of work put them into a Berkeley system, and vice versa. Whether one would *want* to do that is another matter."

The next BSD release, then, may well develop just in the way that its 4.2 forebear did. It probably will offer major functional enhancements over its immediate predecessor, but only at the expense of performance. McKusick acknowledged that, "In 4.3, we strove to provide binary compatibility with 4.2, so that one would be able to take a 4.2 binary and run it easily on 4.3. But in 4.4, one will—at a minimum—need to recompile certain programs. It may even be necessary to make textual changes in a small number of them. The system will offer vast new functionality, but it undoubtedly will have a performance drop. That always happens." ●

David Chandler is the Associate Editor of UNIX REVIEW.

UNIX SYSTEMS UTILITY SOFTWARE —SO YOU CAN GET ON WITH YOUR JOB.

For more information, call or write. (703) 734-9844

UBACKUP
BACKUP, RESTORE, AND MEDIA MANAGEMENT

USECURE
SYSTEM SECURITY MANAGEMENT

SPR
PRINT SPOOLING AND BATCH JOB SCHEDULING

SSL
FULL-SCREEN APPLICATION DEVELOPMENT

S-TELEX
TELEX COMMUNICATIONS MANAGEMENT

SSE
FULL-SCREEN TEXT EDITOR

These products are available for most UNIX or UNIX-derivative operating systems, including System V, 4.2 BSD, 4.1 BSD, Xenix, Version 7, System III, Uniplus, and others.

UNIX is a trademark of AT&T Bell Laboratories.

UNITECH SOFTWARE INC. **Visit us at UniForum Booth No. 1151**

8330 OLD COURTHOUSE RD. SUITE 800 VIENNA, VIRGINIA 22180

Circle No. 10 on Inquiry Card