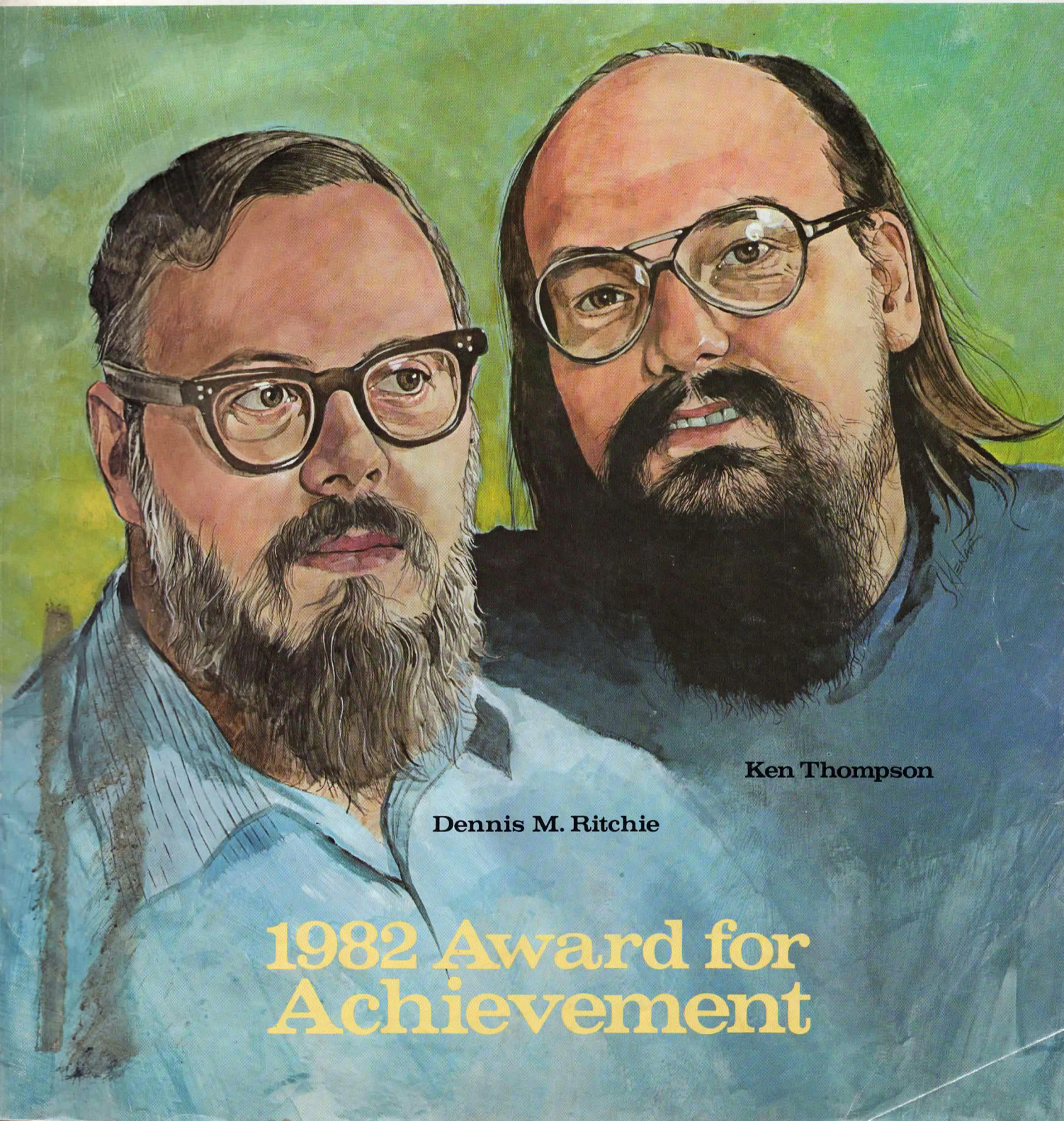
OCTOBER 20, 1982

ANNUAL TECHNOLOGY UPDATE ISSUE

2-μm processes, second-generation optical-projection and wafer-stepping systems, and new computer-aided design tools usher in the 256-K dynamic RAM era / 116

SIX DOLLARS A Mc GRAW-HILL PUBLICATION

Electronics®

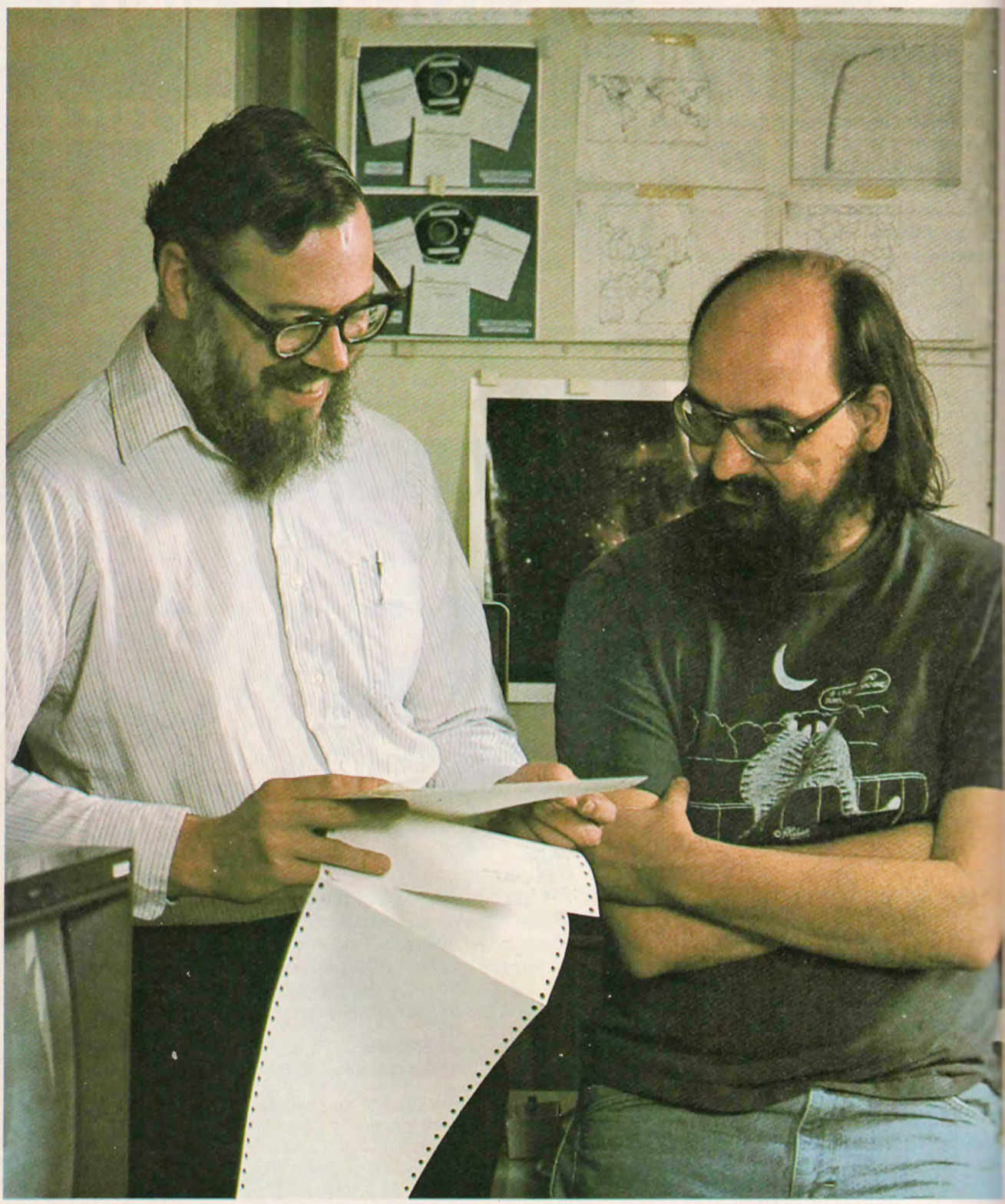


Ken Thompson

Dennis M. Ritchie

1982 Award for Achievement

1982 AWARD FOR ACHIEVEMENT



DENNIS M. RITCHIE KEN THOMPSON

In creating Unix and C, Bell Labs' Thompson and Ritchie have cleared a path to productive programming

by Alfred Rosenblatt, Associate Managing Editor

The winners of this year's *Electronics* Achievement Award are the first in its nine-year history to have made their mark in software, rather than hardware. Ken Thompson and Dennis M. Ritchie, of the Computing Science Research Center of Bell Laboratories, Murray Hill, N. J., have been selected for their efforts in developing the computer operating system Unix and the high-level language C, in which Unix is written—a combination having a major impact across the spectrum of computer usage, from microcomputers to mainframes. In the opinion of the editors of *Electronics*, this stimulant to the growth of the computer industry through the increase in programmer productivity has undoubtedly earned its creators a niche in the annals of the technology.

Unix, a set of programs that controls the hardware devices and software that make up a computer system, is regarded as one of the most powerful operating systems available, particularly for micro- and mini-computers. Since it was not developed for a specific computer, it is fairly machine-independent and can be applied to different computers with minimal effort. It manages multiple users, who can also have access interactively to more than one program at a time, and it automatically manages the system files. Although not considered the most efficient system, it more than makes up for that with ease of application programming.

As of June 1, there were 1,654 Unix licenses from American Telephone & Telegraph Co. and almost 8,000 actual installations—astounding figures for a product that is scarcely being marketed by AT&T but sold on a request basis only, with no support and no warrantees. Fees for commercial licenses range from $20,000 to $43,000 depending on the version, and though customers almost have to demand the operating system from AT&T, they are practically clamoring for it.

Moreover, annual shipments of Unix-based computer systems are expected to exceed the 100,000-unit mark by 1985, according to the Unix information service of market researcher Gnostic Concepts. The Menlo Park, Calif., firm also puts 1985 U. S. end-user expenditures on Unix-related systems and associated software, support, and supplies—largely for mini- and micro-computers—at $5.1 billion.

Systems as successful as Unix often result from a well-conceived and -executed plan to meet a clearly defined need. But "it certainly did not happen that way," says Ritchie. Indeed, the development of an operating system for multiple users and programs was probably a low-priority item at Bell Labs when Thompson began his work in 1969. As Ritchie made clear in the July-August 1978 issue of The Bell Systems Technical Journal, if Unix was designed to meet any specific need, it was the one felt "by its major author, Ken Thompson, and soon after its origin by [Ritchie], for a pleasant environment in which to write and use programs."

Freedom. Unix came about because unconventional but brilliant people committed to software research were permitted to follow their interests and their instincts. A first glance at Thompson and Ritchie in their work place on the top floor of Building 2 at Murray Hill shows that they and their group march to a drum that beats differently from the usual corporate rhythm. Jeans, tee shirts, and sneakers are the rule, as are unconventional working hours. The work often shifts from office to home, where both men have computer terminals.

In fact, Ritchie describes himself as "definitely a night person," accomplishing more at home "because there are too many distractions in the office—a lot of which I find myself, of course. I spend late morning or afternoon talking to people about problems: I hear about them, think about them, gather evidence, and then think about how to solve them at night." Thompson's schedule can be even more unconventional—he's been known to work 30 hours at a stretch.

Thompson is 39, married, and the father of a 13-year-old boy. He was born in New Orleans but moved around a lot as a child because his father was in the Navy. In Italy during high school, he rattled off the *patois* of Naples.

Ritchie is 41, single, and a native of Bronxville, N. Y., just north of New York City. He has lived in the Northeast his entire life.

Arrival. Both joined the labs about the same time, Thompson in 1966 with a master's in electrical engineering from the University of California at Berkeley, and Ritchie in 1967 after eight years in which he "almost" attained a doctorate from Harvard.

Both claim long-time interests in computers. Indeed, Thompson looks upon his MSEE as "an excuse. In those days, computer science was trying to emerge from other departments where it was a sub-branch. At Berkeley, it was double E. I was interested enough to take electrical courses, but software was always the key." Ritchie likewise moved over to computers after doing his thesis in applied mathematics, more specifically in recursive function theory.

At Bell Labs, the two found themselves caught up in a project called Multics, for multiplexed information and computing service, a multiuser time-sharing sys-

tem with a colorful history. Developed in the late 1960s by General Electric Co. and the Massachusetts Institute of Technology to run on Honeywell Inc. mainframes (it is now being sold by Honeywell), Multics finally proved too expensive for Bell and, as its management saw it, a failure. But the researchers disagreed.

Trouble. "Multics succeeded quite well as a research project in operating systems," Ritchie says. "But it turned into an expensive morass for the labs because it was sold as an answer to real computing needs, not as just a research project."

Suddenly, after about four years, Bell Labs in 1969 cut loose from Multics, setting the interested pair of researchers adrift. Thompson, 27 at the time, was bereft. "There was Fortran batch and hardly anything else," he says, for after the Multics experience the labs decided to buy commercial computing as a service or buy operating systems that were supported by their manufacturers.

"I wanted to build an operating system for myself," Thompson says. "I make software, and I needed an environment that would make it easy to build. It was all vague, and we [in computer research] made some proposals for machinery for this purpose, but we were turned down."

Not to be denied, Thompson did some scrounging around on his own and found an obsolete and discarded machine—a Digital Equipment Corp. PDP-7—that had all the ingredients needed for an operating system. Its very fast disk and two terminals—a graphics display and a Teletype machine—were reincarnated by Thompson as a two-user time-sharing system to test a file-system design he and a colleague had worked out on paper. Thus, the progeny of a restless intellect, was born Unix, a single-user multiprogramming system that Thompson idiosyncratically christened after Multics, but with an "ix" instead of "ics."

The creation drew Thompson to the attention of his peers in the research department. "That's when Dennis Ritchie came in, along with others," he recounts. Another two years, and Thompson was ready to put together a pitch for a larger machine. He had his eyes on a PDP-11, but the time was still not ripe at the labs to mention "operating system" and "time sharing" in a research context. Instead, Thompson and his friends, getting wind that the labs' patent department was shopping around for a word-processing system, put together a proposal for what today would be called an office-automation system, "although that description had not yet been invented," Thompson says. "We called it an editing system for office tasks, and in that sort of tainted view of what we really wanted to do, it was approved and we got the -11."

It turns out that they got hardware and little else. "The machine had no software—some diagnostics, but no systems software," he recalls. "Later [DEC] produced RSTS, which is a Basic-oriented, multiple-user interpreter. It was totally inadequate for building languages and doing the kind of computing we were into—writing programs instead of multiplying numbers."

Eventually, the patent department took over Thompson's PDP-11–based system, and he was able to buy a larger PDP-11/20 and a PDP-10 memory management unit. During this time, many diverse computer applications, typically for record keeping or trouble reports, began turning up at Bell System operating companies. Usually, they required multiprogramming or time sharing of some sort, and the PDP-11 was independently chosen for many jobs because, as Thompson says, "it had lots of good peripherals and was reliable, small, and cheap—a typical hardware kind of decision. When they couldn't make it go with the DEC software, they'd look around, hear about Unix, and invariably adopt it."

Hence the proliferation of Unix applications throughout Bell, followed by what Thompson perceived as the need for a "total rewrite." Far more terminals than

The 1982 Achievement Award

For their work in developing the transportable operating system Unix and the high-level language C, Ken Thompson and Dennis M. Ritchie have been designated by the editors of *Electronics* as the recipients of the magazine's ninth annual Achievement Award. The efforts of Thompson and Ritchie, members of the technical staff in the computer research department of Bell Laboratories, Murray Hill, N. J., have had a major impact across the spectrum of computer usage, from microcomputers to mainframes. The elegant interactive features and versatility of the multiuser and multitasking environment the combination creates are gaining it tremendous popularity and a growing constituency among system integrators.

Previous winners have been: in 1974, Gordon E. Moore, president of Intel Corp., for his overall accomplishments; in 1975, the four developers of integrated injection logic, Horst Berger and Siegfried Wiedmann of International Business Machines Corp. and Arie Slob and Cornelius Hart of Philips of the Netherlands; in 1976, Robert C. Dobkin of National Semiconductor Corp. for linear-circuit development; in 1977, Charles H. House of Hewlett-Packard Co. and B. J. Moore, president of Biomation Corp., for major instrumentation innovations; in 1978, Paul Richman, president of Standard Microsystems Corp., for advanced developments in MOS technology; in 1979, Andrew H. Bobeck of Bell Laboratories for his role in the invention of magnetic-bubble memories; in 1980, Abe Offner, Jere D. Buckley, and David A. Markle for their development of the projection mask aligner at Perkin-Elmer Corp.; and in 1981, Carver Mead, professor of computer science and electrical engineering at the California Institute of Technology at Pasadena, and Lynn Conway, research fellow and manager of the VLSI system design area at Xerox Corp.'s Palo Alto Research Center in California, for their work in structuring the design of very large-scale integrated circuits.

had originally been considered needed to be added "without going in and writing a bunch of code for each configuration. This was just sanity on our part; we were spending too much time consulting on the phone."

Before C. As for the language in which Unix was written, Thompson started it in Fortran but, as he puts it, "I lack self-discipline. When it came to the restrictions of software, I relaxed a little and it finally became clear that this was not Fortran. I named it B. It was an interpreter."

Ritchie soon took the intermediate language of the interpreter and produced machine language for a Honeywell mainframe—the group had been considering portability from the very early days, Thompson says. Ritchie wrote a compiler on the PDP-11, then a compiler on the Honeywell machine. Next he rewrote it, introduced structures, and it became C. "From then on, Dennis owned the language," Thompson says, and in 1973 Unix's operating-system kernel was rewritten in C, along with its utility programs.

C is a flexible, high-level programming language that combines powerful logical instructions with the ability to manipulate individual bits and characters. It is easier for a programmer to grasp and maintain than the assembly language in which operating systems were usually written. "It was radical to write an operating system in a high-level language for a small computer," Thompson says. "It was thought that you needed the efficiency of machine language. It just seemed clear to us that that was not true."

In their next rewrite of Unix, Thompson brought in a machine that was very different from the PDP-11—the 32-bit Interdata 8/32. (Interdata was acquired some years ago by Perkin-Elmer Corp. of Norwalk, Conn.) "We realized that because the system was written in a higher-level language, it should be possible to move it to other machines," Ritchie says.

This was done by going into the Unix kernel—the part of the operating system that controls the computers and its peripherals—looking for places where it was dependent on the PDP-11, and rewriting them to make them machine-independent. A full 90% of an operating system can be expressed independently of the machine, Thompson says: "the whole file system, the queuing in the disk drivers, memory allocation, everything but the actual connection to the input/output."

To iron out the remaining 10% of code, the researchers invented new types and concepts for parameters—such as memory addresses or the expression of time of day—that varied with machine. From there, changing the type specification is fairly easy.

They were quite successful. Ritchie points out that Unix is the most widely transported operating system. Machines that use it include the DEC VAX minicomputer, the 68000 and 8086 microprocessor, and two versions of the IBM 370 mainframe. Instead of person years to transport a system to a new computer, Ritchie estimates a system could be readied for demonstration in as little as three person months.

In fact, programmers seem to like to use Unix precisely because of C. "Unix and C were being used at the same time they were being designed," Ritchie points out. "It was not a matter of sitting down and saying, 'here are the specs, I'll go ahead and do it and fix up the results.' When things were added to either one, it was because a need was seen. In particular, the human interface in both cases was always chosen with ourselves in mind.

"One of the fortunate things about doing things this way is that Ken had enough taste and strength to keep the thing coherent. Unix is simple enough so that if you're reasonably clever, you can grasp in full reality the way things are. The same is true of C. It's a fairly pragmatically designed language."

Many systems still lack some of the Unix concepts, Thompson adds. He points to the hierarchical file system—the basic interface with files and devices is the same. Utilities can work on either, which means they are much simpler. And there is a formatless interface with the file system. "Our files are strings of bytes, period," Thompson says. "They don't have lots of irrelevant stuff you must specify. The formatless files and everything else means that the output of any program can be used as the input of any other program." This last concept is referred to as piping.

However, Thompson and Ritchie both fear that Unix's success may also be the source of its undoing. There are now several sources of Unix, and they are starting to become more and more different and incompatible. The Bell System now has several available, the University of California at Berkeley has its BSD4, and there are more. "Since one of the things we were most interested in was portability, the existence of different versions could well defeat the purpose," Ritchie observes.

As for the future, Ritchie is as persistent as a terrier, tweaking Unix, modifying it, redoing various aspects to make it deal with terminals and networks in a cleaner way. He is also interested in distributed computing and in making distributed processing more feasible with Unix.

Thompson, too, is working on networking, and computer chess is another passion—along with colleague Joseph Condon, he designed and built Belle, a machine that plays other machines and is three-time U. S. computer-chess champion.