1983 Turing Award Lecture

Reflections on Software Research

DENNIS M. RITCHIE
AT&T Bell Laboratories

The ACM A. M. Turing Award for 1983 was presented to Dennis M. Ritchie and Ken L. Thompson of AT&T Bell Laboratories at the Association's Annual Conference in October for their development and implementation of the UNIX operating system.

The UNIX time-sharing system was conceived by Thompson and developed jointly with Ritchie in the late 1960's. A key contribution to the portability of the UNIX system was the development by Ritchie of the C Programming Language.

Their seminal paper, "The UNIX Time-Sharing System," was originally presented at the Fourth ACM Symposium on Operating Systems Principles in 1973 and a revised version subsequently appeared in the July 1974 issue of Communications. *This paper received the ACM award for best paper in programming languages and systems in 1974.*

According to the Turing Award selection committee, "The success of the UNIX system stems from its tasteful selection of a few key ideas and their elegant implementation. The model of the UNIX system has led a generation of software designers to new ways of thinking about programming. The genius of the UNIX system is its framework, which enables programmers to stand on the work of others."

Author's present address: AT&T Bell Laboratories, Room 2C-517600 Mountain Avenue, Murray Hill, NJ 07974.

The award is the Association's highest recognition of technical contributions to the computing community. It honors Alan M. Turing, the English mathematician who made major contributions to the computing sciences.

Ritchie and Thompson gave separate lectures at the conference. Ritchie focused on the nature of the environment at Bell Labs that made development of UNIX possible. Thompson reflected on the question of how much one can trust software in contradistinction to people who write it.

Thompson's paper begins on page 171.

Can the circumstances that existed in Bell Labs that nurtured the UNIX project be produced again?

The UNIX[1] 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 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

[1] UNIX is a trademark of AT&T Bell Laboratories.

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 computation 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]; as 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 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, for example 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, 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 systems, 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 a 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 twelve 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 to to exacerbate any current squabbles, but also because it is timely: The organization that publishes the company telephone directory recently asked 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 to support of basic research is deep and will continue [1].

Fundamental research at Bell Labs in physics and 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 machines 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 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 interests 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 pressures 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, usual 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*, 12 (1981).
5. Organick, E. I. *The Multics System.* MIT Press, Cambridge, MA, 1972.
6. Ritchie, D. M. UNIX time-sharing system: A retrospective. *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).

Categories and Subject Descriptors:

C.5.2 [**Computer System Implementation**]: Minicomputers; D.4.0 [**Software**]: Operating Systems—*general*; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management—*systems analysis and design*

General Terms:

Design

Additional Key Words and Phrases:

Directory-assistance system, PDP-11