Remarks for Japan Prize award ceremony for Dennis Ritchie, May 19, 2011, Murray Hill, NJ

M. Douglas McIlroy

Lots of people have exclaimed about what a revelation Unix was. As Tony Hoare said of Algol 60, Unix was not only an improvement on its predecessors but also on most of its successors. The successors today are certainly more capable than Unix was in the 1970s, but rarely as elegant. Adoring admirers have fed Linux goodies to a disheartening state of obesity. If only some reincarnation with the genius and sensibility of Ken Thompson and Dennis Ritchie were still guiding the evolution of Unix!

Unix was shaped both by the remarkable players who brought it into being and by the research environment of Bell Labs. Nobody has told that story better than Dennis himself did in the Bell System Technical Journal in 1984, but perhaps I can add some footnotes.

When Bell Labs pulled out of Multics, a visionary but underperforming project to create the computer utility of the future, Ken, Dennis and Joe Ossanna became convinced that a less ambitious system could provide an equally productive environment. In bull sessions Ken, Dennis and Rudd Canaday outlined a basic architecture that has stood the test of time.

Having been burned by Multics, management was disinclined to pony up for more hardware to try their ideas on, but through the good offices of a sister center in Bell Labs research they obtained first a cast-off PDP-7 computer and then a shiny new PDP-11. Early on, I, as Ken's department head, became fascinated by the system and switched my allegiance from the big iron of the comp center to this tiny experimental system. As they would state later in their famous 1974 paper, the system offered ‘‘features seldom found in even much larger operating systems’’. And ... it was downright fun to use.

The astonishingly capable little system attracted visitors like flies. Folks immediately saw how they could use it for their own purposes. Because it was cheap they could convince *their* management to let them try it. Almost overnight Unix flew out of the lab to sites all over the Bell System, from typing pools to the maintenance floor in telephone switching centers.

Thanks to cheap licensing, Unix also flew into academic computer science departments − most notably Berkeley, which molded it into the standard platform for Arpanet, from which in turn sprang the internet.

Much good came from Unix having been forced to fly under Bell Labs budgetary radar. Had it been richly endowed, it could not have percolated so fast and far, and might well have ended up a niche system like Multics. Much good also came from Bell Labs culture. Management always believed that the best research was inner-directed, and did not attempt to dissuade the Multics alumni from further operating-system research. The fact that another center − Visual and Acoustics Research − contributed both equipment and staff to a project in Computing Science Research well illustrates the collegiality of Bell Labs research, where organization charts didn't necessarily reflect project affinities.

Ken Thompson was undoubtedly the original moving spirit for Unix, but Dennis Ritchie was in on it from the start. And it is Dennis we have to thank for the C language. C made Unix easy to modify and eventually easy to install on new hardware.

With hindsight one might view C as a distillation of previous practice. Not so. Dennis discussed at length the puzzle of how to fully exploit byte-addressed machines. He finally came up with a beautiful way to reconcile address arithmetic with indexing − one of those inventions that is so right that once you see it you think you always knew it. The rightness of C is further attested by the fact that while Unix spread to all kinds of computer, C and its descendents spread even further. C became the language of choice for implementing all kinds of system software both in and outside of Unix shops. C even influenced hardware architecture: proposed instruction sets came to be evaluated partly on the basis of how well they could be exploited by a C compiler.

The spread of Unix was enabled by a major effort of Dennis and Steve Johnson to make both Unix and C portable across machines. The value of portability of applications had long been recognized, but operating systems, which brokered the interaction between applications and hardware, had been seen as inherently tied to the hardware. Yet from a more general perspective an operating system is just a program that happens to run forever. Only small corners of it deal with the idiosyncrasies of a particular machine. Unix proved that the reasons for standardizing Fortran or Cobol also weighed in favor of standard operating systems. New computers get off to a flying start with a corpus of tried and true software and the learning curve is minimized. I well remember how easy it was to get aboard the Bell Labs Cray machine, which came with Unix. There was almost no potential barrier to flipping back and forth between the supercomputer downstairs and microcomputers in the labs.

Another important, but largely overlooked, contribution of Dennis's was the page template for the Unix manual. Devised for the very first Unix manual, that template and the concise writing style that went with it, has stood the test of time. Latter-day deviations from the pattern seem flabby and obscure by comparison.

A distinctive feature of the man-page template was the BUGS section. Here real troubles were disclosed and infelicitous design decisions were noted as challenges for improvement. At one point, the gnomes of AT&T bowdlerized it: in the official Unix product BUGS became APPLICATION USAGE! But in general this beacon of honesty has persisted as part of the Unix ethos.

Dennis was a fixture at meetings of the Usenix users group. Crowds networking in the corridors would break to pack his talks about current developments. Of course every newcomer wanted to see and hear the man behind the system. Old hands came to listen to the master perhaps even more eagerly. If you read one of his papers, you'll see why. Dennis combines perfect control of the technical matter with a polished, but easy writing style, and an unerring sense of how much to say. That felicity is also on display on his home page, which offers engaging pieces about many things he's worked on. One tells of a foray into cryptography wherein he implemented a remarkable code-breaking technique due to Jim Reeds. This resulted in a visit by spooks from NSA, who gently discouraged conspicuous publication.

On Dennis’s home page you can also read about Labscam, a wonderful practical joke that I won’t spoil by summarizing today.

Ritchie and Thompson made an amazing team; and they played Unix and C like a fine instrument. They sometimes divided up work almost on a subroutine-by-subroutine basis with such rapport that it almost seemed like the work of a single person. In fact, as Dennis has recounted, they once got their signals crossed and both wrote the same subroutine. The two versions did not merely compute the same result, they did it with identical source code!

Their output was prodigious. Once I counted how much production code they had written in the preceding year − 100,000 lines! Prodigious didn’t mean slapdash. Ken and Dennis have unerring design sense. They write code that works, code that can be read, code that can evolve.

A mathematician’s distance from the center of his universe is often measured by Erdös number − how many degrees of coauthorship separate him and the legendary Paul Erdos. It has been my good fortune to snag a Ritchie-Thompson number of one. But I am only one among thousands for whom Unix and C have been both an enabler and an inspiration. I’m sure all will join me in applauding the wisdom of the Japan-Prize judges who recognized the singularity and pervasive influence of those inventions.

Dennis, it is an honor to have this opportunity to both congratulate and thank you for the achievement.

Reformatted, with proofreading corrections, October 13, 2011