

What Can We Learn From Dennis Ritchie?

As we noted earlier this week, one of the founding fathers of UNIX and the creator of C, Dennis Ritchie, passed away last weekend. While I feel that many in computer science and related fields knew of Ritchie's importance to the growth and development of, well, everything to do with computing, I think it's valuable to look back at his accomplishments and place him high in the CS pantheon already populated by Lovelace, Turing, and (although this crowing will be controversial, at least until history has its say) the recently-departed Steve Jobs.

UNIX was one of the first multi-user operating systems, allowing scientists and researchers to share computer time on what were traditionally batch-based machines. The concept of multi-user and multitasking were of great interest to researchers simply because of the time required to write, run, and receive the output of batch programs. Computer time, in batch mode, was expensive, as [this anecdote](#) illustrates:

While mulling over the problems of operating systems in 1969, [Ken] Thompson [the co-creator of Unix] in his spare time developed a computer game called "Space Travel." The game simulated the motion of the planets in the solar system. A player could cruise between the planets, enjoy the scenery, and even land the ship on the planets and moons.

The game, first written on Multics and then transliterated into Fortran for the GECOS operating system, ran on a GE 635 computer. The game's display was jerky and hard to control because the player had to type commands to control the ship. Also, it cost about \$75 in CPU time on the big GE 635, a cost that hardly endeared it to management.

At \$75 a game, especially in 1960s dollars, it was hard for a hacker to have any fun. Dennis Ritchie and Thompson worked together to build UNIX as a hacker's paradise, a place to test small programs and share the results. He was a physicist and mathematician by training but entered the nascent world of mainframe and micro-computing at just the right time. The 1960s and 1970s were a time of great change in the way computing interacted with the world. Whereas the the common view was that "These darn computers are going to mess up my phone bill," in reality computers were messing up the status quo. In a few short years paper records were slowly eroded by computation, telephone switches were changing from wild, steampunk octopi into a quasi-mechanical

system of routers and terminals. Bell Labs was at the forefront of it all, tasked with connecting the world through copper wire. Most important, what he was doing was *difficult*, something we forget in the days of drag-and-drop, autocompleting IDEs.

The key to UNIX was the concept of sharing. The OS was begun in 1969 as a reaction to Bell Labs shutting down Thompson and Ritchie's favorite operating system, Multics. With the cooperation of multiple organizations including MIT, a group of four New Jersey Bell Labs programmers began working on a neglected PDP-7 machine where they ported the Space Travel game and began to build out a file system in order to save games. Slowly, a command structure that anyone familiar with modern Linux would understand accreted around this file system. Slowly word of UNIX trickled out of the small cabal of original users and in 1971 the Bell Labs patent filing office began using it to format documents for printing using **nroff**.

It is also important to note that Linus Torvalds was born in 1969, making him a prime candidate to reap the benefits of what you could term the UNIX Age. To come of age in the tumult of a new industry is important and Gates, Torvalds, and Ritchie all were excellent examples of this.

Ritchie went on to create a number of other improvements and, in the development of the C operating system, gave the world its first multi-machine, cross-compatible coding standard that anyone, from a grizzled machine language veteran to a young student in Helsinki, could use and understand. The UNIX source code was passed from programmer to programmer like holy writ even after **AT&T** refused to make it available to education institutions. It was written in C with some of its core components written in machine language in order to shave off time, cycles, and most important, to retain an elegance that Ritchie and Thompson inculcated through cross-pollination of ideas. No one man, not even Ritchie, understood the complexity of the beast that became UNIX and that was by design. The goal was simplicity up front and complexity in the back, a model that everyone in computing would do well to emulate.

Also important was the desire to reach a golden ideal in clarity and elegance. "Peer pressure and simple pride in workmanship caused gobs of code to be rewritten or discarded as better or more basic ideas emerged," wrote Doug McIlroy, a member of the UNIX team. "Professional rivalry and protection of turf were practically unknown: so many good things were happening that nobody needed to be proprietary about innovations".

The question is, then what can we learn about building our own products from this giant of computing? **First, Ritchie and Thompson wanted to have fun.** There was no initial push to make money and, in fact, their goal was to save money or at least hide their gaming by moving it to a less costly machine.

The second is the necessity to work outside your comfort zone. Ritchie was a physicist and a mathematician. However, he became a programmer. While it's clear that his background helped him immensely in building UNIX and C, as Bjarne Stroustrup **noted**, Ritchie was not afraid to attempt to work in new and unfamiliar territory. "If Dennis had decided to spend that decade on esoteric math, Unix would have been stillborn," he writes.

Third is the importance of a hands-off approach to innovation. Ritchie was lucky in that Bell Labs had the money and staff to allow him to hide in the shadows with his friends, creating what they wanted on their own timeline. Google seems to have captured that same sense of internal experimentation obviously with their 20% projects as well as their Labs products that slowly metamorphose into mainstream tools. That the Google founders allowed these 20% projects almost immediately after inception of the company is a testament to Thompson and Ritchie's methodology. People build mean tools when the foreman is watching and masterpieces when left to their own devices.

Finally, we have the importance of sharing. It amuses me to no end to see a small startup cloak their product behind NDAs and secrecy or to watch entrepreneurs mistake glad-handing with networking. When this happens, it's clear that their idea is not novel nor will it be particularly successful nor is their attitude particularly conducive to growth. I would argue that many current, successful entrepreneurs aren't successful because they talk a good game but because they play one.

Arguably the most important software project in the world today, Linux, is important because it gloriously available and open. There are those who will crow that open is not synonymous with profitable, but those people are at best pessimists and at worst fools.

In the end Dennis Ritchie taught us that computing wasn't a secret society, one that required long years of service and special incantations to join. His intellectual largesse is writ large over everything we do online and his still as an explainer – although notoriously shy – shone in his voluminous commentary and **online notes**. Although none of us can attain what he and the Bell/AT&T team attained, especially considering their milieu and the relative nascence of the information age, we're reminded that this doesn't

matter. After all, as we learned from the UNIX source code all those years ago:

** You are not expected to understand this.*

You simply have to build on it.

Img via Wikipedia