BRAIN SCAN

Unix's founding fathers

Dennis Ritchie invented C and was one of the key members of the team behind Unix—two developments that underpin much modern software

Directly west from the southernmost tip of Manhattan, a bit more than 15 miles away, lies the sleepy-looking suburban town of Murray Hill. Just south of the town's centre lies a huge complex of buildings which, despite its size, looks fairly unprepossessing, boring as only business parks in the suburbs can be. But a surprising portion of what passes for modern technology can be traced back to this site, the home of Bell Laboratories, now the research arm of Lucent, but previously that of AT&T, a big American telecoms firm. It was at the Labs, as they are known colloquially, that the transistor was invented in 1947, making possible solid-state computing and paving the way for the microchip.

But the Labs were not only the birthplace, in this sense, of modern computer hardware. Much of modern software—computer programs and the special programming languages in which they are written—originated there too. Two instances in particular stand out: the programming language called C, which from the early 1970s has been perhaps the most popular programming language; and the Unix operating system, first booted up in 1971, and still going strong in everything from laptops to airline-reservation systems. Dennis Ritchie, who has worked at the Labs since 1967, was central to both projects. He is revered as the inventor of C, and, with Ken Thompson, as the co-inventor of Unix.

However, both projects were, in fact, intensely collaborative. Dr Thompson had written C's immediate predecessor, a language known (logically enough) as B. And though Dr Thompson was the first person to work on Unix, Dr Ritchie and others, including Brian Kernighan, Rob Pike and Doug McIlroy (who headed the research group), were fundamental to its development. Dr Ritchie is the last of this group to remain at the Labs—at 62 he retains an aura of youthful enthusiasm. While others have departed for academia or newer companies, he is now the head of systems software research at Bell Labs, and is continuing his research into operating systems and languages.

Dr Ritchie likes to emphasise that he was just one member of a group. With characteristic modesty, he suggests that many of the improvements he introduced when developing C simply "looked like a good thing to do". Anyone else in the same place at the same time, he implies, would have done the same thing. But Bjarne Stroustrup, who came to the Labs later and designed C++, a further improved version of C, disagrees. "If Dennis had decided to spend that decade on esoteric math, Unix would have been stillborn," he says.

All the key participants recall the genesis of Unix and C, and the environment at Bell Labs, as something of an idyll. As Dr Kernighan says, "it was a remarkable collection of really outstanding people who were pretty well paid to do whatever they wanted, and most of them had really good taste about what to work on." Dr McIlroy later wrote that "so many good things were happening that nobody needed to be proprietary about innovations." Unix was not even given a name for more than a year after it was first invented. So much of what they did was done, initially, for themselves alone, sometimes for sheer amusement, and yet it has had a lasting legacy in the world outside. How did this happen, and what lessons follow for today's programmers?

There we were, all in one place

To answer this question, it is necessary, though difficult, to recall just how comparatively primitive the state of computing was 30 years ago. The first version of Unix was written

by Dr Thompson for the PDP-7, a computer made by the Digital Equipment Corporation, which cost a mere $72,000, and came with eight kilobytes of memory, and a hard disk a bit smaller than a megabyte. By contrast, a desktop computer today typically costs a hundred times less, has roughly 64,000 times as much memory and a hard disk 40,000 times as big.

That any software, albeit with many revisions and modifications, could have survived such changes and still be a core technology today is nothing short of astonishing. Amusingly, Dr Ritchie recalls that one of the factors that made Dr Thompson's programming of Unix possible was the fluency he had gained with the PDP-7 while writing an early computer game called "Space Travel", which also happened to be one of the first programs to run under Unix.

The severe limitations of the computers of the day forced Dr Thompson and Dr Ritchie to be ruthlessly efficient in their designs. Of course, this was true of other operating systems and languages of the time, which have long since faded from use. What distinguished the Bell Labs team is that, from the beginning, they focused on doing things that had not been done before, and doing them with clarity. In the words of Arnold Ross, an American mathematics educator, they "thought deeply of simple things". One key and ambitious innovation was the idea of portability. At the time, different kinds of computer hardware ran different operating systems. Both Unix and C were, from the beginning, meant to move beyond the PDP-7; indeed, soon after its invention, Dr Ritchie and Dr Thompson "ported" it to the PDP-11, a more powerful model. Indeed, Dr Ritchie says that the PDP-7 was already obsolete at the time. It was the team's success in designing a rudimentary word-processing system using Unix on the PDP-7 that allowed them to get the funds for a PDP-11. The word processor proved enormously useful to Bell Lab's patent division.

In moving both pieces of software to the PDP-11, they accomplished another remarkable feat. Unix was re-written almost entirely in C. Until then, operating systems, which handle all the day-to-day workings of a computer, had been written in "machine language" which was different for different computers, and nearly opaque to humans. C, on the other hand, is a "high-level language" in that it has a greater degree of abstraction. This step meant that Unix could easily be moved to just about any computer of the time that had adequate memory; all one had to do was write a compiler (a program that translates C into machine language) for each computer. And because Dr Ritchie had been careful to keep the core of C very compact, this was relatively easy to do. Dr Ritchie counts the portability of Unix as one of the two major factors for its success, and says that C "tagged along with that". However, as he admits, the fact is that C was instrumental in making Unix portable.

The second factor Dr Ritchie cites is the "software tool approach" of Unix—breaking up complicated tasks into discrete software tools made the system easier to work with. For example, "paging"—the need, in the old days of character-based displays, to show only one screenful of information at a time—was broken into a separate small program. Dr McIlroy was the one who devised the system of "pipes" that allowed different programs to pass data to one another. This was unusual for the time. Dr Ritchie points out the strange irony that the Unix group was, in fact, outside the main computing centre at Bell Labs, and hence was more willing to experiment.

Another factor helped the duo of C and Unix to spread much faster than they otherwise would have. AT&T was required under the terms of a 1958 court order in an antitrust case to license its non-telephone-related technology to anyone who asked. And so Unix and C were distributed, mostly to universities, for only a nominal fee. When one considers the ineptness of AT&T's later attempts to commercialise Unix—after the court order ceased to be applicable because of another antitrust case which broke up AT&T in 1984—this restriction, an accidental boost to what would later become known as the open-source movement, becomes even more crucial.

The later history of Unix is convoluted, and indeed has again become mired in court battles. Following its origins at Bell Labs, a competing version sprang up at the University of California, Berkeley, which first released its version of Unix in 1977, under the leadership of a graduate student named Bill Joy, who later went on to found Sun Microsystems. Ideological battles raged between adherents of the two versions of Unix through much of the 1980s.

To an extent, this rivalry was stripped of relevance by an unexpected entrant. In 1991, an obscure university student in Finland, Linus Torvalds, announced a project to write a new, open-source clone of Unix from scratch—what has come to be known as Linux. That someone would seek to do this is a testament to the high regard in which programmers hold the achievement of the Bell Labs group. Dr Ritchie, in return, expresses a high regard for Linux, attributing its success to the fact that it was a unified effort, at a time when other competing versions of Unix were mired in legal battles.

Linux is also the true heir of the Unix tradition in the sense that its development process is collaborative. Dr Pike says that the thing he misses most from the 1970s at Bell Labs was the terminal room. Because computers were rare at the time, people did not have them on their desks, but rather went to the room, one side of which was covered with whiteboards, and sat down at a random computer to work. The technical hub of the system became the social hub.

It is that interplay between the technical and the social that gives both C and Unix their legendary status. Programmers love them because they are powerful, and they are powerful because programmers love them. David Gelernter, a computer scientist at Yale, perhaps put it best when he said, "Beauty is more important in computing than anywhere else in technology because software is so complicated. Beauty is the ultimate defence against complexity." Dr Ritchie's creations are indeed beautiful examples of that most modern of art forms.