CS3012 Biography of an Influential Software Engineer:

Dennis Ritchie

Dennis Ritchie was born on the 9th of September, 1941 in Bronxville, New York and died on the 12th of October, 2011. Ritchie got bachelor degrees from the Harvard University in the fields of Physics and Applied Mathematics. During his lifetime, he created one of the most widely used programming languages of all time, C. He also co-developed the UNIX operating system with Ken Thompson for which they both received the Turing Award in 1983, as well as many other awards later in his life. Ritchie is also known for his work on BCPL, ALTRAN, B, Limbo, Plan 9 and Inferno during his time in the Bell Telephone Laboratories.

In the late 60s, Dennis Ritchie was working on the Multics operating system, the development of which was canceled after it was considered too expensive to develop it further. Ritchie himself said that the project was “sort of over-engineered in a sense”. It was at that time that he started his collaboration with Ken Thompson, and after Multics was abandoned, he and Ritchie decided to work on a successor called UNIX. The computer hardware in the 70s had many implementations which had a bad effect on the software engineers’ life as they had to spend more time on adapting their software for each new platform.

Assembly language was used in the original UNIX kernel but Ritchie and Thompson decided that in order to have more control over the data covered by the operating system, they needed a higher level language. Hence, they used BCPL to write UNIX. They squeezed the BCPL into 8 kilobytes and they have renamed it as ‘B’. Storage limitations on the BCPL compiler required a technique on which the output was generated as soon as possible. This forced the creation of B and its successor; C. BCPL’s technological problems were avoided on the design of B. For example, B evaded BCPL’s global vector mechanism for communicating between separately compiled programs by forcing the entire program to be presented all at once to the compiler. This mechanism was used for separately compiled programs, where the software engineer has to specifically associate the name of each externally visibly procedure and data object with a

numeric offset in the global vector. The later versions of B and all of C used a linker to solve this problem by resolving external names occurring in files compiled separately, instead of leaving the problem of assigning offsets on the programmer.

B like BCPL also had major problems. For example, according to Ritchie, its character handling mechanisms were clumsy, a trait inherited from BCPL. Also the B and the BCPL models implied overhead with pointers, by defining a pointer as an index in an array of words, which forced pointers to be represented as word indices. Each pointer reference generated a run time scale conversion from the pointer to the byte address expected by the hardware. Eventually, Ritchie realised that a typing scheme was necessary to deal with these problems and to prepare for newly developed hardware. The threaded code generated by the B compiler was also much slower than assembly language, so it was decided that the UNIX operating system would not be written in B. Ritchie began extending B with adding a character type and rewriting its compiler to generate machine instructions for the appropriate hardware instead of the faulty threaded code. The extended language was called NB (New B) but it didn't exist for very long.

NB supplied the types *int*, *char*, arrays of *int*s and chars, and pointers to *int*s and *char*s. NB retained the semantics of arrays from B and BCPL. A first major invention occurred in NB's development, with a rule that still survives in today's C. The values of array type are converted when they appear in expressions, into pointers to the first of the objects making up the array. This rule was introduced after Ritchie tried extending the type notation in NB. There was no good place to stash the pointer containing the base of the array, for a structure containing an array, and there also wasn't any way to arrange it to be initialised. Another crucial development came in the form of a generalization of types, which was the biggest difference between C and its predecessors. Given an object of any type, it should be possible for the software engineer to describe a new object that gathers other basic types into an array, yields it from a function, or is a pointer to it. Ritchie states that Algol 68 was a major contributor to the scheme of type composition in C. After several improvements, Ritchie believed that NB deserved a different name, and so he followed the single letter style and called it C.

After renaming the language, the operators && and || were introduced which made the evaluation of expressions mechanism more explicit. Many other changes were also added, notably the introduction of the pre-processor, in recognition of the file inclusion in BCPL and PL/I. The original had separate files with string replacements, namely #include and #define of parameter less macros. Moreover, this was extended even further by adding macros with arguments and conditional compilation. Finally in early 1973 the C language and compiler were ready to allow Ritchie and Thompson to rewrite the UNIX kernel for the PDP-11 machine. As well as that, the C language was adapted to other machines available machines such as Honeywell 635 and IBM 360/370.

In 1978, Dennis Ritchie wrote and published a book called “The C Programming Language” with Brian Kernighan, which was regarded to be the authoritative reference on C. In the latter stages of the 1970s, the C language grew more with several new type structures added. Ritchie and Thompson became so confident with C’s usefulness and efficiency that they decided to recode the UNIX system’s utilities and tools.

UNIX and C inspired many programming languages and operating systems in the almost four decades that followed, that are commonly in use today. These include C++, JavaScript, Linux, Mac OS, iOS and Android to name a few. It is hard to dispute that Dennis Ritchie had a major influence with his contributions to the computer science industry and has severely changed the history of computer science.

By Damian Debny

Student Number: 15315535