The Biography of a Software Engineer – Dennis Ritchie
Raivo Ruikis – 17330501

When speaking of the developments in the industry of software development, the Linux operating system is seen as very developer oriented environment which is not particularly geared towards the average user just browsing Facebook. Immediately the name Linus Torvalds comes to mind, the man who created the Linux kernel and during the development thereof, was unsatisfied with the proprietary version control systems available to him and subsequently created Git.
Linus Torvalds however, is not the focus here. The Linux kernel is based upon the Unix operating system, a precursor to many operating systems in use today, particularly server based systems.

Dennis Ritchie, one of (if not the) the most important people in the history of software engineering was born in Bronxville, New York in 1941. In 1967, Ritchie began his long line of work at Bell Labs where he worked with Ken Thompson on the Multics operating system, which was eventually built into Unix. Up until this point, programming was all done on the assembly level with no easier alternatives available. In order to supplement the assembly with a system level language, Thompson created B, which Ritchie later replaced with his own language C, which is still in use to this day and is the foundation for many high level programming languages.

Ritchie died in 2011, one week after Steve Jobs, who regrettably overshadowed him in terms of media coverage. This did not do justice to the immense effect Ritchie had on all of software engineering as without him we would all still be writing in assembly. Nowadays C is used on every major operating system not only Linux and in particular for system drivers for devices such as graphics cards. C itself forms the backbone of mid level programming allowing for high levels of control over memory and other resource usage while still being simple enough for large projects with many developers working together to be able to understand the bigger picture of what the code is doing.

It can be argued that the higher the level of the programming language, the less documentation is required to explain it as the code is easier to interpret and essentially documents itself. In terms of software engineering, this means that many ambitious projects a company might undertake require significantly less turnover time than they would if we didn’t have access to these high level languages.

Personally I do most of my programming in python for smaller scripts and C++ for larger, more intricate projects. Both of these languages are descendants of C and to that end owe their existence to Dennis Ritchie. Arch Linux is also my operating system of choice. Given its very much DIY nature, I get to play around with many aspects of

Ritchie’s original Unix design; the file system structure and in general the modularity of the operating system.

Unix based operating systems such as various Linux distributions and MacOS are by far the most used in software engineering, mostly attributed to their built in C compiler abilities. While Windows can be seen as the most popular OS amongst the general populace but is ill suited to serious software development which is where Unix shines. I for one switched over to Arch Linux due to a driver issue and being frustrated by Windows’ registry system, leading to there being silly issues while doing even something as simple as ensuring that Java is in your path, issues which are basically non exitent or easily resolvable in a Unix environment. These personal details may at times seem irrelevant but at the end of the day we are all software engineers in this course and the problems experienced by one are often experienced by many in this field. As such, when looking at the bigger picture of software engineering, a tool that is easier to use for doing the same job is nothing but a boon and if your only goal is to make better software faster then what more do I need to say?

In conclusion it would be impossible to do Dennis Ritchie justice for all that he has contributed to the world of software engineering, as any modern system in place most likely has some component that can be traced back to the C language or the Unix operating system.