The Impact of The Introduction of UNIX and its Features on the Development of Modern Operating Systems

Logan Murphy

University of North Texas

Abstract

The UNIX Operating system has been around since the 1970s and has had a big influence on the development of the operating systems of today. UNIX introduced several features that many users take for granted today in modern operating systems, such as an organized hierarchical file system, concurrent multitasking and a decent user interface. Without UNIX, it is likely that such features would have been introduced much later and would have set operating system and application design back several several years to a decade.

Keywords: UNIX, modern operating systems, UNIX-like, BSD, Windows, Mac, Linux

UNIX[1] is a multi-user time-sharing operating system that was created originally to play a game called Space War in 1970. But as time went on UNIX was eventually crafted into a full blown operating system that had many features that today are mostly taken for granted because of their presence in almost every modern operating system. One of these features was the type of hardware that UNIX ran on. In a time when most business or university class machines were huge mainframes that took up half a room, such as the IBM 7070 series of mainframes. UNIX was being run on the so called "micro-computers" of the time which merely took up the space of a moderately sized cabinet. UNIX's success can be attributed to its heritage, portability, file system, time-sharing capabilities, user interface, and eventual adoption by academic institutions.

While UNIX was quite innovative, several important features that UNIX introduced where in fact the "full exploitation of carefully selected set of fertile ideas" (Ritchie and Thompson 1974). The two operating systems mentioned here simply possessed the most notable concepts that the developers of UNIX borrowed and gave some context as to what existed before UNIX. The Compatible Time-Sharing System or CTSS for short, was an operating system that was developed at MIT in 1961-62 and was one of the first time sharing operating systems. It is notable for its use in the development during the MultICS project and is notable for introducing several features that UNIX borrowed such as the concept of mail (Vleck 1995), the line editor QED was also reversed engineered for use in UNIX in the form of Ed the line editor. MultICS is a very interesting operating system and is the operating system that UNIX borrowed the most concepts from. Initially started in 1965 as a joint development between Bell Labs, MIT, and

[1] UNIX is not an acronym, see http://catb.org/jargon/html/U/Unix.html for why UNIX is all-caps.

General Electric. In 1969 Bell Labs eventually pulled out of the project because of the financial investment with little immediate return on the company's part. Two of the developers of MultICS that worked at Bell Labs, Ken Thompson and Dennis Ritchie wanted to create an operating system that could keep have the same feel that MultICS was capable of providing but without the massive resource cost that it required. A notable feature that UNIX uses from MultICS is the concept of the hierarchical file system, with MultICS being the first to implement such a file system. Despite this however, aspects of MultICS' design made it somewhat insecure in terms of file access which was one problem that UNIX's file system addresses.

One of the key factors that distinguished UNIX from its competition and lead to its success later on was and still is the portability of the major portions of the UNIX kernel and the various userland utilities. The Programming language C, created by Dennis Ritchie and Brian Kernaghan in 1970, was the reason for the portability of the UNIX operating system. C was created as a language for lower level system programming which at the time was only done with machine dependent assembly languages. Because of this most of the UNIX kernel could be and was written in C without a severe performance penalty and remain portable to other platforms as long as a C compiler existed for that platform. And with the growing popularity of UNIX and the C language because of their utility every relevant platform had a C compiler, even the IBM line of personal computers of a decade later would have UNIX ported to them.

The UNIX file system possessed several innovative features that allowed UNIX to especially stand out from other operating systems at the time, even MultICS which possessed a file system with a similar hierarchical structure. UNIX's file system was organised into a hierarchy of files that resemble a tree structure with a "root" directory containing several other files or sub-directories that branch out from the central directory. While the concept was already implemented in MultICS, it was UNIX that brought the feature into the mainstream because of UNIX's popularity later on. Almost every other operating system with the exception of CTSS which had separate directories for different users, just stored all of the data that was produced either in punch cards, tape, or on the hard drive with no real organisation. A side effect of this organisation was the lack of access control on files, as any user that could access the machine with any user account could glimpse potentially sensitive files. To remedy this each file within the file system contains meta information that determine other user's access to the file and how they can access them. For example a script that Bob writes couldn't be accessed by Joe unless Bob sets the file to read write for other users, and the script couldn't be executed until the meta information were changed to allow the file to be execute. In addition to file permissions the UNIX file system kept track of information such as file size, date created, date edited, and a bit that indicates whether or not the file was a directory. UNIX also had special device files that would otherwise behave as normal files, but were in reality interfaces to various physical devices such as tape drives or printers, which data could be written to and then sent to the appropriate device that the file was associated with. As an example if you wanted to write a tape archive file to a tape drive you would list the special file that the tape drive was along with the files you wanted to tar and it would be written to the tape archive. This method of device interface was

implemented and made possible by the way the PDP-11 addresses such devices, with specific addresses within memory where data could be written and then would be sent through the Unibus to the device in question.

The most pivotal aspect of UNIX's success from a usability standpoint was UNIX's advanced time-sharing capabilities, allowing UNIX to effectively handle multiple users and process running on the same machine. To give some context, before Concurrent Multitasking as we know it in operating systems existed there was a process known as time-sharing. The concept was fairly simple on paper, while the mainframe was not being used by someone else, other processes from other users could be run thus getting more use out of the large and expensive to run mainframes. Unfortunately time-sharing is limited by the complexity of the hardware that it's implemented on, because to achieve true "concurrent" processing the operating system would have to save all the information of the process onto some form of storage, and then load the data of the next process back into memory for every running process on the machine. Unfortunately the hard drives and memory at the time had slow read and write speeds which made concurrent processing largely impractical. As technology got more robust however, time-sharing evolved from a queue of batch processes to a more concurrent form of operation. An example of this from 1962 is the IBM 7094, the memory of the 7090 series of mainframes consisted of 2 32 kilobyte banks that could store the state information of two different processes at once and allowed for more robust task switching on the operating system and a smother experience on the users' part. The Unibus present in the 1970 Digital Equipment Corporation (DEC) PDP-11, the

system that UNIX was ran and developed on initially, allowed for asynchronous access to peripherals such as memory and the hard drive. Because of the Unibus, reads and writes to and from memory and hard drive controllers could be done simultaneously allowing for more rapid and seamless switching of process on a UNIX system.

UNIX's user interface contributed to its success because of its robust yet simple and powerful design. The user interface of UNIX came to be known as the "shell" which is yet another concept that UNIX borrowed from the MultICS project. The term shell originated because it was the layer of interaction above the operating system, thus its shell. The idea for the shell came from a developer of CTSS Louis Pouzin. He felt that smaller commands should be able to be used as components for larger programs or jobs (Pouzin 2000). Eventually the shell concept was adapted for use in the MultICS operating system. When the development of UNIX came along the Thompson Shell was born into the first version of UNIX in 1971. The Thompson Shell was created to solve some of the problems that was observed in MultICS, and to provide a more cohesive interface for UNIX. One thing that was unique in UNIX was that almost every command were all separate programs. While this may seem like a drawback at first the Thompson Shell had the ability to redirect the input or output of a particular program into that of another. This allowed for the combinations of these smaller programs into one much more complex program. This construct was possible because of the concept of a pipe, which would let the output of one program flow into that of another. With this a user could create something like a spell checker (AT&T tech channel) out of just the utilities available on a stock system.

Eventually however the Thompson Shell proved inadequate for more complex tasks. As such a man named Stephen Bourne created a new shell that was christened, the Bourne Shell and was released in version 7 UNIX in 1979. The Bourne Shell was more robust than the Thompson shell, things such as "if" and "goto" statements important for conditional execution were present only as separate commands on the Thompson Shell, were built in to the Bourne shell, giving the user more expressive power and better performance when writing complex shell scripts.

The success of UNIX is also attributable to the institutions that it was distributed to during the development of UNIX and the variations that arose as a result. At first UNIX was sold only to universities such as the University of California at Berkeley for use in computer science research, plus it was relatively affordable at the time given the comparative price of other operating systems. Eventually as talented programmers started writing their own utilities for UNIX a full fledged release of a new variant of UNIX called Berkeley Software Distribution (BSD). One of BSD's most important contributions was sockets, the foundation of all modern day networking, which includes the Internet. As time went on AT&T saw the opportunity to commercialise UNIX, more so especially after the people that used it in college were moving in to positions of authority in major businesses. So as a result of this UNIX was licensed by AT&T (as they weren't allowed to sell it because of an antitrust lawsuit that was resolved in 1982) to several different manufactures to create there own diverse but largely incompatible UNIX Variants. Eventually an ideological and business conflict emerged between BSD, and AT&T System V revision 4 of UNIX on which version would be the standard for UNIX as the different

variants of UNIX were usually based on one or the other. Eventually an organisation called the Open group and the IEEE created the single UNIX specification and the POSIX standard respectively, which most UNIX variants and UNIX likes attempt to conform to to this day.

UNIX's influences are most notable through the applications of itself and its decedent's in addition to the user interface innovations that UNIX implemented. An operating system classified as a UNIX like known as Linux is notable for its wide applications from servers to personal computers to embedded performance critical applications. To give an example almost every web page that a user might read is likely served up by a system running Linux, and if not it's served up by a server running a BSD or rarely Oracle Solaris or Windows Server. In terms of embedded systems Linux is almost always the operating system that is being used in those applications. A non-exhaustive list of applications include drone hardware, self driving cars, manufacturing robots, "smart" appliances, and most significantly cellular phones, with 74 percent (statcounter 2019) of cellular phones running android, a variation of Linux. The reasoning behind Linux and UNIX based operating systems is because of how user customisable they are as operating systems allowing them to be altered to fit a variety of roles without losing out on performance. UNIX and UNIX-likes are usually pretty developer friendly because of the utilities included to ease the task of monotonous jobs making UNIX-likes like Linux epically popular with developers. While not as popular as Windows, UNIX has directly and indirectly

influenced the Windows operating system as well, Windows was at one point POSIX compliant. On a final note, UNIX is still utilised by various universities developing cutting edge technology such as the MIT use UNIX to develop and deploy their projects such as advance robotics software and AI research. From a usability perspective, without the spread of an operating system with a decent implementation of time-sharing running multiple programs such as watching a video and editing multiple files at once would not be possible, nor would complex user interfaces and graphical user interfaces would be very hard to implement. Overall UNIX has grown into a flexible and capable operating system that has had clear influence on other operating systems throughout its 50 year life spawn through its various distributions.

Sources Cited:

N.d Multics retrieved from https://stuff.mit.edu/afs/athena/reference/multics-history/

n.d. The UNIX System -- History and Timeline retrieved from

http://www.unix.org/what_is_unix/history_timeline.html

Johnson S. Portability of C Programs and the UNIX System Retrieved from

https://www.bell-labs.com/usr/dmr/www/portpap.html

N.d Definition - what does Time-sharing mean? Retrieved from

https://www.techopedia.com/definition/9731/time-sharing

Ritchie M. D. (1979) The Evolution of the Unix Time-sharing System retrieved from

https://www.bell-labs.com/usr/dmr/www/hist.html

Vleck V. T. (1995, June 22) The IBM 7094 and CTSS retrieved from

https://multicians.org/thvv/7094.html

AT&T tech channel (2012, Feb 22) AT&T Archives: The UNIX Operating System

https://www.youtube.com/watch?v=tc4ROCJYbm0

Pouzin L. (2000, Nov 25) The Origins of the Shell retrieved from https://www.multicians.org/shell.html

Mascheck S. (n.d) The Traditional Bourne Shell Family retrieved from https://www.in-ulm.de/~mascheck/bourne/index.html#origins

Mobile Operating System Market Share Worldwide (n.d) retrieved from http://gs.statcounter.com/os-market-share/mobile/worldwide

PDP-11 Busses n.d retrieved from http://www.psych.usyd.edu.au/pdp-11/

Ritchie, D., Thompson, K., (1974) The UNIX Time-Sharing System retrieved from https://people.eecs.berkeley.edu/~brewer/cs262/unix.pdf

Vleck V. T. (1993) Multics Security Retrieved from https://www.multicians.org/security.html