CS3510: OS-1

Assignment 1:

A brief history of Operating Systems

PLAGIARISM STATEMENT:

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honour violations by other students if I become aware of it.

***Name*: Shreyas Jayant Havaldar**

***Date*: 13-11-2019**

***Signature*: S.J.H.**

Introduction

Need for Operating Systems

“Necessity is the mother of invention.”

This couldn't be truer in case of Operating Systems. In the nascent stages of computing, a very restricted group of people, the majority of them being engineers; designed, assembled, operated and maintained computing machines. The only way to be able to exploit the resources and computing power of the systems back then was to be well versed in assembly language or being capable of manually wiring-up the plug boards. This limited the target audience of the computers to a very small group of people.

Predecessors

Moving away from hand-coded machine language, assemblers and compiler simplified the programming process. As computers got quicker, multiple jobs could be executed which gave rise to the need for **job scheduling**, **file management** and **resource management**. Libraries of support code on punched cards or magnetic tape followed, which would be linked to the user's program to **assist** in operations such as input and output. As client involvement in the operation diminished to a few monitoring people who would simply look after the machines, the concerns of data security, tampering and data loss surfaced which led to the development of **authentication** and **security** measures. All these measures form integral components of Operating Systems even today.

We’ll look at the history of Operating Systems using a generational timeline, listing the primary defining feature of computing machines of that era and then how Operating Systems became what they are today, evolving as new needs arose.

Operating Systems Generation Timeline

- The First Generation (1945 - 1955): **Vacuum Tubes and Plugboards**
- The Second Generation (1955 - 1965): **Transistors and Batch Systems**
- The Third Generation (1965 - 1980): **Integrated Circuits and Multiprogramming**
- The Fourth Generation (1980 - Present): **Personal Computers**
- The Fifth Generation (1990–Present): **Mobile Computers**

Evolution of Operating Systems

The development of Operating Systems could be seen in chronological sequence, each component evolving from a lacking experienced in the preceding version, thus sequentially leading to the modern Operating Systems which we all use today.

Batching

The term OS could be first definitively used for **GM-NAA I/O**, produced in 1956 by General Motors' Research division for its IBM 704. Transistors led to the development of the computer systems that could be manufactured and sold to paying customers. There was a clear distinction between designers, builders, operators, programmers, and maintenance divisions. Each department worked largely independently of each other, thus giving rise to modularity in the skill set and thus diminishing the importance of individuals. A batch system was employed where tray full of jobs were read onto a magnetic tapes in the input room using inexpensive computer, such as the IBM 1401, which was brilliant at I/O, but useless at numerical calculations. Much more expensive machines, such as the IBM 7094, were used for the number crunching, intensive operations.

Machines were largely programmed in FORTRAN and assembly language. Examples include **FMS** (the Fortran Monitor System) and **IBSYS**, IBM's operating system for the 7094.

Uniformity

The lack of uniformity in the Operating Systems distributed to every user was a nagging roadblock, leading to different architecture and I/O systems for each main-frame systems and most applications would have to be manually adjusted, recompiled, and retested for every unique system. Updating to a newer system to satisfy the ever increasing dependence was a headache and exorbitant proposition for consumers. This roadblock was finally overcome with IBM focusing on developing a single, uniform OS, the **OS/360** for every hardware, essentially establishing a family of hardware called the System/360. The IBM 360 was the first major computer line to use ICs, thus providing a major price and performance advantage.

Concurrently, for the UNIVAC 1107, UNIVAC, the first commercial computer manufacturer, produced the **EXEC** I OS, and Computer Sciences Corporation developed the EXEC II OS and delivered it to UNIVAC.

The CPU remaining idle for large chunks of time, primarily due to I/O wait time lead to research concluding in establishment of multiprogramming systems to juice out every computational ounce from the CPU.

Multiprogramming

The solution developed was to partition memory into several chunks, with a different task in each partition. The processor was not idle while a job was completing its I/O operation. Another job was scheduled on the processor so that its time would not be wasted. IBM's **MFT** incorporated this feature.

But the wastage of time in instances of jobs failing, and long waiting-times paved the way for timesharing, a variant of multiprogramming, in which each user has an online terminal.

Time-Sharing

Each user has access to the computing machines, with fast interaction being provided to small job users and big batches being run when the CPU becomes idle. IBM developed **TSS/360**, over-emphasising on the time-sharing aspect of the Operating Systems, which resulted in a over-complicated structure. This concept's importance wasn't lost though, **CTSS** (Compatible Time Sharing System), was developed at M.I.T. on a specially modified 7094. Also the Dartmouth Time-Sharing System (**DTSS**), an OS first developed at Dartmouth College between 1963 and 1964, was the first successful large-scale time-sharing system to be implemented, and was also the system for which the BASIC language was developed. The lack of necessary protection hardware lead to the development of security measures.

Virtual Memory

Burroughs Corporation introduced the B5000 in 1961 with the **MCP** (Master Control Program) operating system, which was the first to introduce the concept of virtual memory to the commercial environment. It also was fully written in ESPOL, a high level programming language.

Security

Project MAC at MIT, working with GE and Bell Labs, developed **MULTICS**. It was a pioneering step to establish a large scale virtual machines with the concept of ringed security privilege levels. It was designed to support hundreds of users on a machine only slightly more powerful than a PC, although it had much more I/O capacity. This revolutionary concept couldn't find commercial success but lead to establishment of multiple core concepts of Operating Systems.

Portability

Minicomputers took the world by storm in the late 1960's and emerged **UNIX**, a stripped-down, one-user version of MULTICS developed at AT&T Bell Laboratories originally for the PDP-7, and later for the PDP-11. Being written in C language, it's portability across all machine architectures enabled it to become the first-choice of mini-computers and work-stations. The influence of UNIX was so phenomenal, that it spurred a plethora of new free and open-source Operating Systems projects like **BSD**, **Linux**, **GNU**, and **System V**. The advent of personal computers also led to the growth of networks, creating network operating systems and distributed operating systems.

Graphical User Interface

With the development of LSI (Large Scale Integration) circuits, personal computers (microcomputers) became the norm. Disk based Operating Systems like **CP/M, MS-DOS.** These all Operating Systems were based on users typing in commands from the keyboard. This all changed when Steve Jobs saw the potential of GUI in Xerox's **PARC** and eventually built a UNIX based **Mac OS X** based on **MACH** microkernel. XEROX failed to recognise the potential of GUI systems and this blunder, till date is perhaps the biggest technological mis-steps ever encountered. Windows followed suit with its graphical interface based **Windows 95**. Microsoft Windows, MacOS and the open source Linux are the 3 most popular Operating Systems today and keep evolving everyday. Windows tried to evolve into a more Tablet friendly Operating System with **Windows 8,** but facing wide-spread backlash stuck to the more intuitive PC interface for **Windows 10**. Linux still remains the top choice of developers all across the world, primarily due to its open source nature.

Mobile Computing

In the early 1990s, Psion released the Psion Series 3 PDA, a small mobile computing device. It supported user-written applications running on an operating system called EPOC which eventually evolved into **Symbian,** which dominated the mobile

Operating Systems scenario in the early 2000’s only to be brutally killed by the much more user-friendly **iOS** based on the Unix-like Darwin**,** released in 2007 by Apple and **Android,** based on Linux released in 2008 by Google. Android being open-source and employing development in the globally popular JAVA language, became the global leader in Mobile Operating Systems.

Conclusion

Operating Systems have come a long way from being a privilege of few able programmers to an inextricable component of daily life. Operating Systems often define our computing choices, with fierce competition in the domains of PC’s as well as Mobiles, with even fiercer users whose swear by “their” OS and act as advocates for its use. Thus the history of Operating Systems, is an important story we should be aware of, as to realise how easy any computational activity is today; again the courtesy of this magnificent journey of Operating Systems, and to ensure we do not repeat blunders that giants like XEROX committed by not realising potentials and keep striving to develop even better Operating Systems as each day progresses.