History of Operating Systems

Ayman Moumina

History of Computing
Prof. Tim Bergin
5/3/01

Operating systems are the software that makes the hardware usable. Hardware provides "raw computing power." Operating system makes the computing power conveniently available to users, by managing the hardware carefully to achieve good performance.

Operating systems can also be considered to be managers of the resources. An operating system determines which computer resources will be utilized for solving which problem and the order in which they will be used. In general, an operating system has three principal types of functions:

i. **Allocation and Assignment of System Resources:** Allocation and assignment of system resources such as input/output devices, software, central processing unit, etc.
ii. **Scheduling:** This function coordinates resources and jobs and follows certain given priority.
iii. **Monitoring:** This function monitors and keeps track of the activities in the computer system. It maintains logs of job operation, notifies end-users or computer operators of any abnormal terminations or error conditions. This function also contains security monitoring features such as any authorized attempt to access the system as well as ensures that all the security safeguards are in place (Laudon and Laudon, 1997).

Throughout the history of computers, the operating system has continually evolved as the needs of the users and the capabilities of the computer systems have changed.

As Weizer (1981) has noted, operating systems have evolved since the 1940s through a number of distinct generations, which roughly correspond to the decades. Although this observation was made in 1981, this is still roughly valid after two decades. In this paper, we shall also follow the similar approach and discuss the history of operating systems roughly along the decades.

Early History: The 1940s and the 1950s:

In the 1940s, the earliest electronic digital systems had no operating systems.
Computers of this time were so primitive compared to those of today that programs were often entered into the computer one bit at a time on rows of mechanical switches. Eventually, machine languages (consisting of strings of the binary digits 0 and 1) were introduced that sped up the programming process (Stern, 1981). The systems of the 1950s generally ran only one job at a time. It allowed only a single person at a time to use the machine. All of the machine's resources were at the user's disposal. Billing for the use of the computer was straightforward - because the user had the entire machine, the user was charged for all of the resources whether or not the job used these resources. In fact, usual billing mechanisms were based upon wall clock time. A user was given the machine for some time interval and was charged a flat rate.

Originally, each user wrote all of the code necessary to implement a particular application, including the highly detailed machine level input/output instructions. Very quickly, the input/output coding needed to implement basic functions was consolidated into an input/output control system (IOCS). Users wishing to perform input/output operations no longer had to code the instructions directly. Instead, they used IOCS routines to do the real work. This greatly simplified and sped up the coding process. The implementation of input/output control system may have been the beginning of today's concept of operating system. Under this system, the user has complete control over all of main storage memory and as a result, this system has been known as single user contiguous storage allocation system. Storage is divided into a portion holding input/output control system (IOCS) routiner, a portion holding the user's program and an unused portion (Milenkovic, 1987).

Early single-user real storage systems were dedicated to one job for more than the job's execution time. Jobs generally required considerable setup time during which the operating system loaded, tapes and disk packs were mounted, appropriate forms were placed in the printer, time cards were "punched in," etc. When jobs completed, they required considerable "teardown" time as tapes and disk packs were removed, time cards were "punched out" etc. During job setup and job teardown, the computer sat idle.

Users soon realized that they could cut down the amount of time wasted between the jobs, if they could automate the job-to-job transition. First major such system, considered by many to be the first operating system, was designed by the General Motors Research Laboratories for their IBM 701 mainframe beginning in early 1956 (Grosch, 1977). Its success helped establish batch computing – the groupings of the jobs into a single deck of cards, separated by control cards that instructed computers about the various specification of the job. The programming language that the control cards used was called job control language (JCL). These job control cards set up the job by telling the computer whether the cards following it contain data or programs, what programming language is used, the approximate execution time, etc. When the current job terminated, the job stream reader automatically reads in the control language statements for the next job and performs appropriate housekeeping chores to facilitate the transition to the next job. Batch processing system greatly improved the use of computer systems and helped demonstrate the real value of operating systems by managing resources intensely. This type of processing called single stream batch processing systems became the state-of-theart in the early 1960s (Orchard-Hays, 1961).

The 1960s: The Era of Timesharing and Multiprogramming:

The systems of the 1960s were also batch processing systems but they were able to take better advantage of the computer resources by running several jobs at once. They contained many peripheral devices such as card readers, card punches, printers, tape drives and disk drives. Any one job rarely utilized all of a computer's resources effectively. It was observed by operating system designers that when one job was waiting for an input-output operation to complete before the job could continue using the processor, some other could use the idle processor. Similarly, when one job was using the processor, other jobs could be using the various I/O devices. The operating system designers realized that running a mixture of diverse jobs appeared to be the best way to optimize computer utilization. The process by which they do so is called multiprogramming in which several users simultaneously compete for system resources.

The job currently waiting for I/O will yield the CPU to another job ready to do calculations if another job is waiting. Thus, both input/output and CPU processes can occur simultaneously. This greatly increased CPU utilization and system throughput. To take maximum advantage of multiprogramming, it is necessary for several jobs to reside in the computer's main storage at once. Thus, when one job requests input/output, the CPU maybe immediately switched to another, and may do calculations without delay. As a result, multiprogramming required more storage than a single system. The operating systems of the 1960s, while being capable of doing multiprogramming, were limited by the memory capacity. This led to the various designs of multiprogramming such as variable position multiprogramming that helped to utilize the storage capacity much more efficiently (Smith, 1980).

In the late 1950 and 1960, under the batch processing mode, users were not normally present in the computing facility when their jobs were run. Jobs were generally submitted on punched cards and magnetic tapes. The jobs would remain in the input tables for hours or even days until they could be loaded into the computer for execution.

The slightest error in a program, even a missing period or comma, would "dump" the job, at which point the user would correct the error, resubmit the job, and once again wait hours or days before the next execution of the job could be attempted. Software development in such an environment was particularly a slow process (Weizer, 1981). University environments provided a fertile ground for dealing with such limitations. Student programs tended not to be uniform from week to week, or from one student to another, and it was important that students received clear messages about what kinds of errors they made.

In 1959-1960, a system called MAD (Michigan Algorithmic Decoder) was developed at the University of Michigan. MAD was based on ALGOL, but unlike ALGOL, is took care of details of running a job in ways that few other languages could do. MAD offered fast compilation, essential for a teaching environment and it had good diagnostics to help students find and correct errors. These qualities made the system not only attractive to the student programmer but also to various researchers at the University of Michigan Campus (Rosin, 1969).

While there were attempts to provide more diagnostics and error-correcting mechanisms by the groups such as those in the University of Michigan, another group tried to develop systems that would allow greater access to the computing systems and reduce the waiting time for jobs to execute. One of the major developments in this direction was timesharing system which enabled many users to share computer resources simultaneously. In the timesharing mode, the computer spends a fixed amount of time on one program before proceeding to another. Each user is allocated a tiny slice of time (say, two milliseconds). The computer performs whatever operations in can for that user in the allocated time and then utilizes the next allocated time for the other users. What made such a concept possible was the difference between the few milliseconds (at least) between a user's keystrokes and the ability of a computer to fetch and execute dozens, perhaps hundreds of simple instructions. The few seconds a user might pause to ponder the next command to type in was time enough for a computer, even in those days, to let another user's job to execute, while giving the illusion to each user that the complete machine (including I/O devices) and its software were at his or her disposal. Although this concept seems similar to multiprogramming, in multiprogramming, the computer works on one program until it reaches a logical stopping point, such as an input/output event, while for timesharing system, every job is allocated a specific small time period (Laudon & Laudon, 1997).

MIT's Department of Electrical Engineering was one of the pioneers of the timesharing system under the guidance of John McCarthy, Robert Fano and Fernando Corbato. Since 1957, it had been running a computer IBM 704 in a batch-processing mode. However, the instructions of programming and the development of software were very difficult given the long turnaround time, the time between the submission of a job and the return of results, of hours and even days. This motivated them to develop a system that would reduce the turnaround time substantially. This led MIT to implement the first timesharing system in November 1961, called CTSS – Compatible Time-Sharing System. The demonstration version allowed just three users to share the computer at a particular time. It reduced the turnaround time to minutes and later to seconds. It demonstrated the value of interactive computing as the timesharing system was also called (Crisman, 1964).

Timesharing systems helped facilitate the software development process significantly. With turnaround time reduced to minutes, no longer a person writing a new program had to wait hours or days to correct errors. With timesharing, a programmer could enter a program, compile it, receive a list of syntax errors, correct them immediately and re-execute this cycle until the program is free of syntax errors thereby reducing development time significantly (Crisman, 1964).

Within a year of MIT's successful demonstration, several other universities, research organizations and manufacturers, noting the advantages of timesharing system, had begun to develop their own systems. Many of these systems were further evolved into next generation of operating systems. For example, MIT developed Multics operating system as the successor of CTSS. Multics, although was not successful, gave rise to perhaps the most versatile operating system existing even today – the UNIX system. In 1964, IBM also developed CP/CMS system at its Cambridge Scientific Center, a timesharing system for its new System/360 mainframe which eventually became the major operating system – VM operating system – for its System/360 and System/370 computers (Weizer, 1981).

The 1960s – Disappointing Efforts of IBM to Develop OS/360 Operating System:

In April 1964, IBM introduced its new generation of mainframe computers, System/360. It was so named because it was aimed at full circle of customers, from business to science – customers who did a lot of mathematical calculations as well as those who did simpler arithmetic on large sets of data. System/360 was not just one model but a whole line of computers targeted to different customers. The major selling point was the promise that programs written for one model would also work in larger models, thus saving a customer's investment in software as business grew. This system and its successor System/370 dominated the mainframe market in the 1960s and 1970s and its basic architecture served as the anchor for IBM's product line into the 1990s (Pugh et al., 1991).

For this computer system, IBM also planned a very ambitious operating system, called OS/360. It was immediately recognized that the developmental effort would be huge with an initial estimated budget of $ 25 million. IBM chose Frederick J. Brooks, Jr., one of the most able students of computer pioneer Howard Aiken. Brooks eventually would become a leading advocate in the 1970s for developing an engineering discipline for software construction, and the author of one of the most famous books regarding software engineering, The Mythical Man-Month.

OS/360 was perhaps the biggest and the most complex programs that have ever been attempted. According to the initial plan, it would consist of hundreds of program components, totaling more than a million lines of code, all of which had to work in a perfectly coordinated manner. OS/360 was to utilize the technology of "multiprogramming" as well. Although multiprogramming had been successfully implemented at that time, so far, it was not implemented in such a large scale as it was in OS/360. While realizing that incorporation of multiprogramming was a marketing necessity, the design team also realized that it could delay the delivery of the OS/360 in time for System/360 introduction and thus decided to delay the delivery of a full multiprogramming system until mid-1966s (Pugh, 1991).

The development of the OS/360 control program – the heart of the operating system – was based at the IBM Program Development Laboratories in Poughkeepsie, New York. There, it had to compete with other System/360 software projects that were all asking for the company's best programmers which were already in short supply. The development task got underway in the spring of 1964 and was methodically organized from the start – with a team of a dozen program designers leading a team of sixty programmers trying to implement some forty functional segments of code. Soon, the schedules began to slip not for any specific reason but for numerous small causes. More people were added to the development team and by October 1965, there were some 150 programmers who were at work on the control program. Nevertheless, at that time, the development was estimated to be running at about six months late. A test trial was conducted and found that the system to be very sluggish and the software needed extensive rewriting to make it usable. Moreover, by the end of 1965, fundamental design flaws emerged for which there appeared to be no easy remedy (Pugh, 1991).

In April 1966, IBM publicly announced the rescheduling of the multiprogramming version of OS/360 for delivery in the second quarter of 1967 – nine months later than it was originally planned. IBM's problems with OS/360 development were now a public knowledge. Users were anxious and so were the shareholders. Inside IBM, there was a growing sense of desperation. The only possible response it had was to add more and more programmers to the task. This was later recognized by Brooks as being precisely the wrong thing to do. First of all, the quality of programming staff go down as more and more people are added. Second, difficulty of coordinating between their work which became more and more fragmented, is considerable. This was more pronounced in the ways when structured programming was not in existence and one programmer's work was much more difficult to match with another. In general, writing a major piece of software was a subtle task and it did not help to keep adding more and more programmers. As Fred Brooks had noted, "The bearing of a child takes nine months, no matter how many women are assigned" (Brooks, 1974, p. 17).

At the peak, more than 1,000 people at Poughkeepsie were working on OS/360. These included programmers, technical writers, analysts, secretaries and assistants – and all together some 5,000 staff-years went into design, construction and documentation of OS/360 between 1963 and 1966 (Pugh, 1991).

OS/360 was finally introduced into the market in mid-1967, a full year late. By that time, IBM had spent half a billion dollars on it – four times the original estimate of $ 125 million. According to IBM's chairman Tom Watson, Jr., this was "the single largest cost in the System/360 program and the single largest expenditure in company history" (Watson, 1990, p. 353).

When OS/360 came out, it was not just late but full of bugs as well, that took years to eradicate. IBM had to offer several other operating systems to the users of System/360 including its CP/CMS system which eventually developed into its VM operating system. The experience of OS/360 also provided IBM with enough experience to develop another operating system in the early 1970s called MVS. These two operating systems continued to serve the IBM mainframes until present.

The 1960s – Garmisch Conference: The Concept of Software Engineering:

The experience of OS/360 made the computer community aware that the software had not been catching up with hardware and because of that, the potential of rapidly advancing hardware technology was not being realized fully. Many of the software projects were becoming unmanageable and were going utterly wrong. The OS/360 project was illustrative of these problems this era faced in developing operating systems and other software. These systems were huge conglomerates of software written by people who really did not understand that software, as well as hardware had to be engineered to be reliable, understandable and maintainable. Endless hours and enormous amount of money were spent detecting and removing bugs that should never have been in the systems in the first place. Errors in the earliest phase of the projects were not located until long after the projects were delivered to customers where they were much more difficult and expensive to correct. People turnover often resulted in large numbers of software modules being scrapped and then rewritten by new people because the existing modules could not be understood (Brooks, 1975).

At that time, the US Department of Defense and NATO were developing defense systems which utilized state-of-the-art software. However, they were worried about the software quality since even a small bug in the software of military systems, would have disastrous consequences. One of the actions taken by NATO was to take initiative in sponsoring a world-wide working conference of academic and industrial software developers. The conference was held in Garmisch, Germany in October 1968 under the term "Software Engineering." The term was deliberately chosen by the organizers in order to emphasize "the need for software manufacturers to be based on the types of the theoretical foundations and practical disciplines that are traditional in the established branches of engineering" (Naur and Randell, 1968, p. 13).

The Garmisch conference brought about a major cultural shift in the perception of programming at that time. Prior to that, software development as a discipline was fragmented with no theoretical foundations. The Garmisch conference was the catalyst in providing a framework for developing better software. Some of these ideas included structured design, formal methods and developmental models, all of which were designed to manage the inherent complexity of writing large programs. Structured design methodology took the view that the best way to manage complexity was to limit the software writer's field of view and keep him/her in focus. Formal methods were expected to simplify and mathematize the design process by which programs were created. The development model viewed the software writing process not as a once-and-for-all construction project, like the way IBM approached OS/360 project, but as a more organic process, like the building of a city. Thus, software would be conceived, specified, developed and implemented – then it can be improved over from time to time, with added "bells and whistles" (Weizer, 1981).

These frameworks, especially the structured design methodology and development model, are still being used today. They helped build the future software including the operating systems much more efficiently. IBM's other mainframe operating systems such as MVS and VM system have been much more effective than the OS/360 mainly because of the use of these methodologies. The emergence of the field of software engineering and the recognition of the importance of developing a disciplined and structured approach to the construction of reliable, understandable and maintainable software were truly fostered by the devastating experiences with many of the operating system development efforts of the 1960s.

The 1970s – General Development:

The 1970s saw several significant development that vastly broadened the scope and importance of operating systems. The experimental timesharing systems of the 1960s evolved into solid commercial products in the 1970s. This was vastly facilitated by the improvement in data communications between computers. The TCP/IP (Transmission Control Protocol/Internet Protocol) started to become widely used especially in military and university computing environments. Communications in local area networks were made practical and economical by the Ethernet standard developed at Xerox's Palo Alto Research Center (Quarterman & Hoskins, 1986).

As more and more data started to be transmitted through the communication lines, they became more and more vulnerable to interception and unauthorized access. Operating systems of these days not only needed to deal with the interconnectivity of the networks but also the security. Encryption received much attention – it became necessary to encode proprietary or private data so that even if the data were compromised, it was not of any value to anyone other than the intended receivers. Other aspects of computer and network security such as viruses and hacking had increasingly challenged the operating systems. As a result, the design of a secure operating system received top priority at that time (McCauley, 1979).

Several major operating systems were developed during these periods, some of which such as IBM's MVS and VM operating systems for its mainframe computers and Bell Labs' UNIX operating system are still in operation. UNIX operating system is particularly noteworthy because this is the only system that has been successfully implemented in every kind of computer – from microcomputers to supercomputer. In the next section, the development of UNIX is described in somewhat detail.

The 1970s – Development of UNIX:

From 1965-1969, Bell Labs participated with General Electric and Project MAC at MIT in the development of Multics system. Originally designed for the mainframe, Multics was a large and complex system. The Multics designers envisioned a general purpose computer utility that could essentially be "all things to all people" (Organick, 1972, p. 3).

As the effort progressed, it became clear that although Multics was likely to deliver the variety of services required, it would be a huge, expensive system and very difficult to develop. For these reasons, Bell Laboratories withdrew from the effort in 1969. This, however, did not dissuade some members of the Bell Labs' research staff to work on a far less ambitious system. The group, led by Ken Thompson, sought to create a simple computing environment for programming research and development, which later they named "UNIX" – "a somewhat treacherous pun on 'Multics,'" (Ritchie, 1984, p. 1580) according to the words of one of the co-developers, Dennis Ritchie. Given the limited budget, as the Labs was no longer funding it, and the high cost of mainframe computer time, they had to scrounge around and found a discarded obsolete computer – a PDP-7 which was manufactured by DEC (Digital Equipment Corporation). It was a minicomputer designed for dedicated laboratory application and provided only a fraction of the power of the conventional mainframe. The design of UNIX evolved over a period of few months in 1969 based on a small set of primitive concepts.

By the early 1970s, UNIX was working well to the satisfaction of the designers, providing remarkably powerful facilities for a single user on the PDP-7. However, the designers still had difficulty convincing the computer community of its merits. However, in 1973, Dennis Ritchie, a former Multics teammate joined the UNIX team which made a considerable difference. First, like any other operating system before it, the first version of UNIX was written in Assembly language which made it machine-dependent. Ritchie designed a new language called "C," especially for the UNIX to be written on. This was a "systems implementation language, designed for writing programming systems in much the same way that higher level languages FORTRAN and COBOL were designed for scientific and commercial purposes respectively. The use of C made UNIX "portable," that is, machine-independent so that it could be implemented on any computer system. In fact, this was the first time an operating system was written on a higher-level language than Assembly language, and thus became the first operating system with portability (Milenkovic, 1987). The designers also enlarged the capabilities of the original design of UNIX such as expanding the capability of the text-processing features. They also convinced Bell Labs' patent department to use the system for preparing patent applications. This was the first time they found a real prospective user for UNIX. Bell Labs made funding for a larger computer available to them and the newly launched minicomputer Digital Equipment PDP-11/45 was selected for this purpose (Ritchie and Thompson, 1978).

AT&T, the parent company of Bell Labs before telephone deregulation of 1983, was not allowed to compete in the computer industry, so it made the UNIX systems available to universities at a nominal fee. More importantly, AT&T also distributed its source code. The minimal design of UNIX and its simplicity compared to complex operating systems of the mainframe allowed it to develop immediate rapport with the academic world and research laboratories. By 1975, UNIX systems had become extremely popular in the universities and a users' organization developed that evolved into the group called USENIX and within a couple of years, graduates of the universities began to import UNIX culture into the computer industry, making UNIX the standard operating system among the computer professionals in the 1980s (Salus, 1994).

By 1977, UNIX began to grow organically as more and more software were added to the basic system originally developed by Ritchie and Thompson. The clean, functional design of UNIX made this organic growth possible without affecting the inherent reliability of the system. One of the very powerful versions of UNIX was developed by the University of California at Berkley. It was Berkley UNIX with TCP/IP communication standards that helped transform the restricted ARPANET to the wideopen Internet (Laudon and Laudon, 1997). Sun Microsystems is one of the many firms which took full advantage of UNIX. Its SunOS operating system is UNIX-based. Sun wanted a system for supporting a network of workstations. In the 1980s, it enhanced
Berkeley's version to include facilities for a graphic, windowing and mouse-oriented interface. It also included facilities for diskless workstations to use the network for storing and sharing (Courington, 1985).

In 1983, Thompson and Ritchie received the ACM's Turing Award, the most prestigious award in the computing community. In its citation, ACM noted "the genius of the UNIX system is its framework, which enables programmers to stand on the work of others" (Salus, 1994, p. 81).

It is not that UNIX does not have limitations. It was a complicated set of commands. While it is a delight to the professional programmers, it is not user-friendly enough for the novice users and thus, has not been truly become an operating system of choice for the personal computers of the 1990s where user-friendliness is the important criterion for user acceptability. Its security features are generally weak because it allows multiple users and multiple computer jobs to access same files simultaneously, although some versions of UNIX have been modified to be more secure. It requires relatively large amount of RAM and disk storage capacity (Laudon and Laudon, 1997).

Despite its limitations, UNIX system appealed to the users because of simplicity in design while being flexible and open. More importantly, its popularity reflected a cultural shift that was occurring in the 1970s in the computing community, as the independent-minded users were beginning to reject the centralized mainframe with its rigidity and relative lack of access and immediacy in favor of a decentralized small minicomputers that were already being introduced in the market. Many computer lobbyists and programmers preferred smaller decentralized systems because of the accessibility and flexibility. The cost of mainframe time, even with timesharing might have been too high for many programmers with limited funds, who needed to test and debug their programs and rewrite and run them again. At that time, there were a few good operating systems available to satisfy the needs and UNIX filled this void. As Ritchie noted, "Because they were starting afresh and because manufacturers' software was, at best unimaginative and often horrible, some adventuresome people were willing to take a chance on a new and intriguing, even though unsupported, operating system" (Ritchie, 1984, p. 758).

This cultural shift of the computing community was also facilitated by another development – this time in the hardware – which totally revolutionized the computer industry, in the form of the microprocessor. The development of microprocessor (popularly known as microchip) in the 1970s eventually changed the nature of computer by being the enabling technology for personal computers. In this personal computing environment, operating systems became elevated to a higher level of importance and by the 1990s, became the dominating factor in the software industry. In the following section, the development of microprocessor and the personal computer in the 1970s is discussed.

The 1970s – The Beginning of Microprocessor and Personal Computer Era:

The enabling technology for the personal computer is the microprocessor (popularly known as microchip). These processors were actually integrated circuits, which printed thousands of transistors onto small silicon chips. The first integrated circuits were produced in 1962 for the military and they cost about $ 50 and contained an average of half a dozen active components per chip. After that, the number of components on a chip doubled every year. By 1970, it was possible to make LSI (Large Scale Integration) chips that contained thousands of active components in the chip (Freiberger and Swaine, 1984).

In 1968, a firm called Intel was established to commercially produce these integrated circuits. Initially, it marketed its integrated circuits to calculators, watches and games which in fact revolutionized these industries. In 1969, while designing an integrated circuit for a new scientific calculator for a Japanese manufacturer, Intel engineer Ted Hoff came up with the idea to design a general purpose chip in which specific calculator functions can be performed. Eventually, Intel started to market them in November 1971, under the brand name Intel 4004, a 4-bit microprocessor, selling for $ 1,000 (Slater, 1987).

In 1973, Intel replaced the 4004 with an 8-bit version under the brand name Intel 8008. By this time, several manufacturers also had begun to produce their own microprocessors – such as the Motorola 68000, the Zilog Z80 and the Mostek 6502. With this competition, the price of the microprocessor fell to around $ 100 (Veit, 1993).

In January 1975, the first microprocessor-based computer, the Altair 8800 was introduced. It was essentially a kit for hobbyist, sold by mail order as a kit for about $ 400 and a few-hundred more already assembled. It contained an Intel 8080 microprocessor produced by Micro Instrumentation Telemetry System (MITS) in Albuquerque, New Mexico. It had no display, no keyboard and not enough memory to do anything useful. The only way the Altair could be programmed was to enter programs in pure binary code by flicking the small hand switches on the front, a situation reminiscent of the early computers of the 1940s. When loaded, the program would run; but the only sign of the execution of the programs was the shifting patterns of the neon bulbs on the front. It had very little that could be considered truly useful for a user. But, it was a dream come true for some computer hobbyists if they are so dedicated to keep flickering the switches (Ferguson and Morris, 1995).

The limitation of the Altair actually came as a boom to many small-time entrepreneurs and computer buffs, as it gave an opportunity to develop add-on features and boards so that extra memory, teletypes and audiocassette recorders (for permanent data storage) could be added to the basic machine. Another group of people thought that they could make Altair more usable by developing software for it.

The news of the introduction of Altair in the market made these computer buffs and entrepreneurs immediately jump into the opportunities of adding-on hardware and writing software. In 1975, two of these computer buffs Bill Gates and his childhood friend Paul Allen decided to write BASIC, in order to take advantage of these opportunities. They decided to develop BASIC language for Altair. After obtaining the permission from Ed Roberts, the owner of MITS, Bill Gates and Paul Allen formed a partnership which they named Micro-Soft (the hyphen was dropped later). After six weeks of intense programming effort, they delivered a BASIC programming system to MITS in February 1975. They refused to sell it to MITS though, rather they licensed it in return for a royalty. The Altair 8800 and the add-on boards and BASIC software transformed electronics as the computer hobbyists showed strong enthusiasm. During the first quarter of 1975, MITS received $ 1 million in orders for Altair. The royalty from this provided a substantial cash flow for Microsoft, then only a tiny company (Veit, 1993). The royalty concept as providing regular cash flow reinforced Bill Gates' mind regarding its advantage and in future negotiations with others he would stick to this position, as he would be in the 1980s with his negotiations with IBM. Since 1975, the personal computer industry saw rapid expansion. The peripherals such as keyboards, disk drives and monitors were added to the bare-bone Altair models. There were also several Altair clones that began to appear in the market.

In 1976, the first operating system for these Intel-based personal computers was written by a system programmer named Gary Kildall. As a consultant for Intel, he developed an operating system called CP/M (Contro Program for Micros) for Intel 8080. While doing that he recognized that the floppy disk would make a good mass storage device for small programs that managed the flow of information to and from a floppy disk. He realized that a disk had several advantages over magnetic or paper tape. First, it was faster. Second, the user could both read and write data on it. Its primary advantage was that a disk had "random" access. Users did not have to run through the entire spool of tape to get at a specific piece of data. To accomplish this, however, required some special programming, something which IBM did in the 1960s for its mainframe computers called Disk Operating System (DOS). However, a personal computer disk operating system had little to do with mainframe operating system. There was no need to schedule and coordinate the jobs of many users. There was no need to "spool" or otherwise direct data to several printers, card punches and tape drives; a personal computer had only a couple of ports to worry about. What was needed was rapid and accurate storage and retrieval of files from a floppy disk. A typical file would, in fact, be stored in a set of fragments, inserted at whatever free space that were available on the disk. The operating systems for personal computer needed to be designed in such a way that it can find these free spaces, put data there, retrieve them later on and then reassemble the fragments (Kildall, 1981).

With this concept, he extended the CP/M operating system for Intel 8080 that can also deal with disk drives. He called this a specialized code, the BIOS – Basic Input/Output System. In 1977, a manufacturer of Altair-clone, IMSAL approached Gary Kildall to use CP/M for its products. Kildall rewrote CP/M in order to incorporate the BIOS for the disk drives. This change standardized the operating system for the Intelbased system for a while. This system allowed the floppy disk as a storage medium and expanded the capabilities of the personal computer in terms of storage of data and programs. By 1977, many microcomputer manufacturers, including MITS, IMSAL and others, were offering 8-inch floppy disk drives, mainly manufactured by Shugart Associates with CP/M as the operating system (Veit, 1993).

Despite the increasing popularity of the microcomputers among the computer hobbyists and professionals, they did not appeal to the non-experts and the households. They were still intimidating for most of the non-professionals and novices and there was not much use for the individuals and households. In 1975, one company, Apple Computer was established that changed the microcomputer from being a challenging tool for the computer professionals and hobbyists to a useful personal computer for households and non-expert individuals.

The Rise of Apple Computer:

In 1975, two computer enthusiasts, Steve Jobs and Steve Wozniak, founded a company called Apple Computer. This was nothing extraordinary by itself, as there were numerous small companies assembling computers were popping up all over. But, what distinguished Apple from others was its vision and determination to make microcomputer a consumer product for a much greater market of households and non-expert individuals. For this purpose, they packaged the product as a self-contained unit in a plastic case, able to be plugged into a standard household outlet like just any other appliance; it was to incorporate a keyboard to enter data, a screen to view the output and some form of storage to hold data and programs. Jobs and Wozniak also realized that the machine would need software to appeal to anyone other than a computer enthusiastic. With this vision, Apple I came out in 1975, which could plug into a television set display (Young, 1988).

In 1977, a much improved version of Apple called Apple II came out. It used MOS Technologies' (a spin-off of Motorola) 6502 chips rather than Intel 8080, the standard chips at that time. It used fewer chips than the comparable Altair machines, yet it outperformed them due to superior circuit design. It had excellent color graphics capabilities, which made it suitable for interactive games. Although Apple I's BASIC was written by Steve Wozniak, for Apple II, Microsoft was contracted out for a better version. The fee from this helped Microsoft to overcome the threat of bankruptcy at that time given it had only few contracts for writing software at that time (Manes and Andrews, 1993).

Initially, Apple II used a cassette tape but by the end of 1977, Wozniak designed a disk controller that simplified floppy disk drives that were much more simpler than the ones that were used by Altair and others at that time. Apple's floppy-disks were 5.25 inches and could hold 113 KB of data. The disk drive sold for $ 495 which included an operating system software and a controller that plugged into one of Apple II's internal slots. The operating system was written by Jobs and Wozniak. It was written in UNIX which enabled it to be portable. In 1980, Apple used an attachable card called Soft Card from Microsoft which allowed Apple II to run CP/M. For Microsoft, this piece of hardware was one of the best selling products at that time (Williams and Moore, 1985).

In 1979, Apple Computer added the first spreadsheet software for microcomputers called VisiCalc. It also added a word processing software. With these application software, coupled with flexibility and relative ease of use, Apple Computer demonstrated the potential of a personal computer at the desktop. Apple's success convinced many others of the feasibility of such a computer. One of the firms that decided to enter in the personal computer industry was none other than IBM, the most dominating firm in the computer industry at that time and this changed the computer industry dramatically since 1980.

The 1980s – IBM's Entry into the Personal Computer Industry and its Effect on Operating Systems:

While the 1980s saw such development related to the operating systems as distributed processing and client-server processing, it is the personal computer segment that had the major impact on the computer industry. In this decade, personal computer and its operating system played a significant role and became the dominating segment in the computer industry.

The watershed event in this decade starts with IBM's entry into the personal computer market. So far, IBM and most of the other industry heavyweights have shied away from entering into the personal computer market doubting its potential for being a consumer product and leaving it mainly to the upstart small firms. But, Apple's success demonstrated that the microcomputer had the potential for being attractive to the households and individuals as well as to the businesses with spreadsheet and word processing software. After careful consideration, in 1980, IBM decided to enter the personal computer (PC) market.

As soon as the decision had been made, IBM moved with remarkable speed. Traditionally, IBM's bureaucratic development structure had been taking about three years to market a product. But, IBM decided that in order to speed the process of bringing its PC to market, it would outsource all the components that it did not already have in production. Although IBM was the world's largest software developer at that time, paradoxically, it did not have the skills to develop software for personal computers. Its bureaucratic software development procedures were slow and geared to large software projects but it did not have the flexibility, agility and other critical skills to develop the kind of software needed for personal computers (Chopsky and Leonsis, 1988).

IBM decided to use the fastest microprocessor available at that time – Intel 16-bit 8088 which gave it a significant advantage over the other personal computer brands which used 8-bit Intel 8080. For operating systems, Gary Kildall's CP/M system was the logical choice as it had already established itself as the standard for Intel-based systems. Digital Research, the firm established by Kildall, was at that time developing a 16-bit version of CP/M and IBM decided to approach Kildall for this new version. IBM also decided to include a version of BASIC as the standard option for its PC. Microsoft's BASIC was at that time the standard in the Altair and other Intel-based microcomputers (Chopsky and Leonsis, 1988).

However, for some reason, Kildall lost the opportunity. There are several versions of the story of how he lost it. One version of the story was that when IBM team arrived, he was doing some recreational flying and without his presence, his wife (or other executives of the company, according to another version) refused to sign the nondisclosure agreement that IBM wanted Digital Research to sign (Manes and Andrews, 1993).

When IBM negotiating team visited Microsoft to close the deal on BASIC, it sought Bill Gates' help in recommending what to do about the operating system. Bill Gates was highly eager to accommodate IBM's needs and offered to provide one to IBM, which without seeing the actual product, entered into an agreement. Bill Gates, with his experience of the advantages of royalty rather than outright selling of BASIC for Altair, insisted a royalty for each copy it sells rather than selling outright. IBM agreed with royalty fee set to be between $ 10 and $ 50 for each copy sold.

Microsoft, however, did not have an actual operating system ready, neither did it have the resources to develop one to beat IBM's deadline. However, Gates knew that Tim Paterson, the owner of Seattle Computer Products had developed an operating system for Intel 8086 chip, known internally by QDOS for "Quick and Dirty Operating System." Microsoft initially paid $ 15,000 for the rights to use the product and later paid a larger sum of money for the complete rights. Microsoft, after slight modification named it MSDOS (MS standing for Microsoft) (Ichbiah and Kneeper, 1991).

During the summer of 1991, the first personal computers by IBM began to come off from the IBM assembly plant and by early August, initial shipments totaling 1,700 machines were delivered to Sears Business Centers and ComputerLand stores, the two retail outlets that IBM had chosen. A fully equipped IBM personal computer, with 64 KB of memory and a floppy disk, cost $ 2,880 (Ichbiah and Kneeper, 1991).

Within the next few weeks, the IBM personal computer became a runaway success exceeding almost everybody's expectations. IBM's brand name and IBM's extraordinary marketing effort contributed to this popularity. While many business users had hesitated over buying an Apple or another relatively unknown brand at that time, the presence of IBM logo – most venerated brand name in computer industry at that time – convinced them that the personal computer technology was for real. In this manner, IBM did legitimate the personal computer (Ichbiah and Kneeper, 1991).

During 1982-1983, the IBM personal computer became an industry standard. IBM's decision to allow it to have an open architecture meant the other firms can copy its design. This encouraged other manufacturers to produce computers with the same architecture which came to be known as clones, in order to take advantage of the huge demand that market was experiencing. The clones were usually less expensive but run on the same software. Among the most successful of the clone manufacturers was Houstonbased Compaq. Several of the leading manufacturers of other brands such as Tandy, Commodore, Victor and Zenith also switched into making IBM clones. As the demand for IBM and its clones increased, so did the software. In response, new application software started to come to the market at an increasing rate. Lotus Development Corp's Lotus 1-2-3 spreadsheet software, WordStar as word processing software, dBase as the database software were the market leaders at that time in their respective product categories. Alongside these hardware and software, a huge sub-industry of peripherals also developed that manufactured printers, memory boards and various add-ons. By 1983, the personal computer impacted society so much that the Time magazine awarded as their Man of the Year, not to a person but a machine: the PC.

One company that benefited the most out of this was Microsoft. Almost every model of IBM PC and its clones were supplied with its MS-DOS operating system. As hundreds of thousands and eventually millions of machines were sold, money poured into Microsoft. By the end of 1993, half a million copies of MS-DOS had been sold, netting $ 10 million (Ferguson and Morris, 1995). This revenue stream allowed Microsoft to diversify into computer application software without having to rely on external venture capital. It also allowed Microsoft to cross-subsidize some of the software that initially did not succeed. For example, in mid-1983, Microsoft began to develop a word processing software package called Word. That product was released in November 1983 with a publicity splash which included distribution of some 450,000 diskettes demonstrating the program in the PC World magazine. Even so, Word was initially not a successful product and had a negligible impact on the market leader at that time, WordStar. But, the cash flow from the MS-DOS allowed Microsoft to continue to market the product at a loss until the opportunity came later to bundle it properly with its new generation of operating systems, Windows (Edstrom and Eller, 1998).

One major deficiency of MS-DOS was that it was not very easy to use. The user interacted with the operating system through a command line interface in which instructions to the operating system had to be typed explicitly by the user in an exact manner. If there was even a single letter out of place or a character is missing or mistyped, the user had to type the line again. While many technical people were delighted in the intricacies of MS-DOS, ordinary users found it highly perplexing and sometimes intimidating. This problem, what is called lack of user-friendliness, prevented the PCs being truly acceptable as a consumer product. In 1984, Apple solved this problem when it introduced its Macintosh model.

1980s – Introduction of Macintosh in 1984:

Apple was the only major microcomputer manufacturer that did not switch to producing IBM-compatibles but chose the path of its own. In 1984, it unveiled its Macintosh model which was far superior to any of the IBM PCs or its clones in terms of user-friendliness. It used a technology called graphical user interface (GUI) and a pointing device called a mouse. The movement of the mouse moves the cursor on the screen. By moving the cursor to the appropriate words or pictures (called icons) and then clicking them allowed user to give appropriate commands to the computer. In this manner, the user need not memorize a lengthy list of commands that must be typed into the computer.

The graphical user interface or GUI (which is sometimes called WIMP for Windows, Icons, Mouse and Pull-down menus), had been in the process of developing since 1960s by various groups and by 1981, Xerox had used it for its Xerox Star computer. But, Xerox priced it too high and failed to provide critical hardware and support. Xerox never took the personal computer seriously and made very little marketing effort. As a result, Xerox perhaps missed one great opportunity (Smith and Alexander, 1988).

In December 1979, Steve Jobs was invited to visit Xerox Palo Alto Research Center (PARC) in Silicon Valley where Xerox was developing the technology for "the office of the future" where applications to graphical user interface was displayed. Since then, Jobs had in mind that the company's next computer had to look like the machine he had seen at Xerox PARC. At first, he started Lisa project with this concept in mind. But, Lisa project was a failure and Apple put all its effort into the Macintosh project that had started in 1979 (Lammers, 1986).

In January 1984, Apple introduced the Macintosh with huge promotional efforts that included a legendary Super Bowl commercial. Priced at $ 2,500, it received high praise for its design, aesthetic quality and user-friendliness. Its elegant operating system so far was a great achievement. It displayed a combination of aesthetic beauty and practical engineering that was extremely rare to find (Guterl, 1984).

But, after the initial enthusiasm, the sales were disappointing. The problem was the lack of sufficient number of software and other add-ons. This is because of Apple's policy to keep Macintosh's architecture closed. This closed architecture meant that hardware and software developers would find it difficult to create their own Macintosh add-ons and software without the close cooperation with Apple. A lack of third-party support created a problem for Macintosh, and sales never peaked up (Wallace and Erickson, 1992).

In order to reposition itself, Apple invited several of the leading software firms to develop software. But, a lack of sufficient level of demand for Mac software (which had then 10 percent market share of the personal computer market) caused this software developers to be discouraged. The only major firm which did accept to write software for Mac at least for a while was Microsoft. Microsoft, since 1981, had been somewhat involved in the Macintosh project, developing some minor parts of the operating system. By taking the offer from Apple to write programs for them, Microsoft found an environment much insulated from the highly competitive IBM-compatible market where it was facing intense competition for its application software against such strong competitors as Lotus in spreadsheet applications and Micro Pro in word processing. Later, it would be able to convert the same applications, so that they would run on the IBM-compatible PC. By 1987, Microsoft, in fact, was deriving half of its revenue from its Macintosh software (Veit, 1993). More importantly, working on the Macintosh gave Microsoft firsthand knowledge of the technology of graphical user interface on which it based its new Windows operating system for the IBM PC, to which we turn next.

1980s – Launching of Microsoft's Windows:

Microsoft started its own graphical user interface (GUI) project in September 1981, shortly after Bill Gates had visited Steve Jobs at Apple and seen the prototype of Macintosh computer under development. Initially, it was estimated that it would take six programmer years to develop the system. But, when version 1.0 of Windows was released in October 1985 – it was estimated that the program containing 10,000 instructions had taken eighty programmer years to complete (Wallace and Erickson, 1992).

The Microsoft Windows was heavily based on the Macintosh user interface. On 22 November 1985, shortly after Windows was launched, Microsoft signed a licensing agreement to copy the visual characteristics of the Macintosh, thereby avoiding legal trouble for version 1.

However, although competitively priced at $ 99, sales of Windows 1.0 were sluggish at first because it was unbearably slow. Although a million copies were sold, most users found the system to be little more than a gimmick and the vast majority of users stayed with MS-DOS. Part of the reason was that the microprocessor at that time –Intel 80286 – was not fast enough to support GUI technology. In the late 1980s, when the next generation of microprocessors – the Intel 386 and 486 became available- the GUI became much more supportable. At that time, Microsoft introduced its Windows 2.0. Windows 2.0 popularity also provided Microsoft with the opportunity to bundle its Excel spreadsheet software and its world processing software, Word. With it allowed their market share to increase considerably and eventually become the market leader in their respective applications.

In April 1987, IBM and Microsoft announced their joint intention to develop a new operating system called OS/2. On 17 March 1988, Apple filed a lawsuit alleging that Microsoft's Windows 2.0 infringed Apple's registered audio visual copyrights protecting the Macintosh interface. Apple argued that

Microsoft's original 1985 agreement with Apple had covered only version 1 of Windows but not version 2 (Ichbiah and Kneeper, 1991).

The lawsuit was eventually dismissed after three years. Meanwhile, Microsoft was achieving one of the most dramatic growths of any business in the 20th century. Most of the growth was achieved in the applications software. However, significant revenues were also derived from its Windows 2.0 operating system (Ichbiah and Kneeper, 1991).

Success of Windows 2.0 made Microsoft to lose interest in OS/2. When OS/2 was finally launched in early 1988, Microsoft failed to provide adequate software support for it. Because of this, the first version of OS/2 never took off. To the annoyance of IBM, Microsoft continued its Windows project intensely while ignoring OS/2 project. In fact, in 1990, Microsoft introduced a new Windows version – version 3.0.

The 1990s and Beyond – The Dominance of Microsoft in the Operating System Market and the Challenges It Faces:

On 22 May 1990, Microsoft introduced Windows 3.0 all around the world with an extravagant publicity and events that cost about $ 10 million. Windows 3.0 was wellreceived. Microsoft continued to take advantage of its dominance in operating systems by bundling the application software with operating systems and by taking advantage of its intimate knowledge of the source code of the operating system.

In April 1991, IBM announced a new version of OS/2 – release 2.0. The new operating system was said to have cost $ 1 billion to develop and was designed to replace all previous operating systems for IBM-compatible computers, including Microsoft's own MS-DOS and Windows. However, despite the sound technical merits of OS/2, IBM continued to lose ground against Microsoft – partly because of a lack of appealing software and partly because of failure to market it effectively (Ichbiah and Kneeper, 1991).

Failure of OS/2 resulted in further dominating position for Microsoft Windows. This position was further reinforced in 1995 with its release of Windows 95 which was an immediate success. Since then, it has introduced several other versions of Windows including Windows 2000 and Windows XP.

Despite its successes and the dominating position of the Windows operating system, Microsoft faces several challenges. One is the US Department of Justice's lawsuit against Microsoft charging that it had used its dominating position illegally. While it had lost in District Court, the case is currently pending in the Appeals Court.

The other challenges have to do with the future of the operating system as such. The advent of the Internet has opened up new possibilities and challenges. First, there are open-source systems like Linux, which is available freely online to anybody who wants to view, download or adapt it. This clearly threatens Microsoft's dominating position. Second, the Internet may provide a platform in which operating system may become much less important. In this rapidly changing environment of computing and information technology, it is extremely difficult to say which direction the operating systems will take.

References:

Brooks, Jr., F.P. The Mythical Man-Month: Essays in Software Engineering. Reading, Mass: Addison Wesley, 1975.

Chopsky, J. and T. Leonsis. Blue Magic: The People Power and Politics Behind the IBM Personal Computer. New York: Facts on File, 1988.

Courington, B. The UNIX System: A Sun Technical Report. Mountain View, Sun Microsystems, 1985.

Crisman, P.A. et al. The Compatible Timesharing System. Cambridge, Mass: MIT Press, 1964.

Edstrom and Eller. Barbarians Led by Bill Gates. New York: Henry Holt, 1998.

Ferguson, C.H. and C.R. Morris. Computer Wars: How the West Can Win in a Post-IBM World. New York: Random House, 1993.

Freiberger, P. and M. Swaine. Fire in the Valley: The Making of Personal Computer.
Berkeley, CA: Osborne/McGraw-Hill, 1984.

Grosch, H.R.J. "The Way It Was in 1957." Datamation, September 1977, pp. 121-132.

Guterl, Fred. "Design Case History: Apple's Macintosh." IEEE Spectrum, December 1984, pp. 34-43.

Ichbiah and S.L. Kneeper. The Making of Microsoft. Rocklin, CA: Prima Publishing, 1991.

Kildall, Gary. "CP/M: A Family of 8-and 16-bit Operating Systems." Byte, June 1981, pp. 219-229.

Lammers, ed. Programmers at Work. Redmond, WA: Microsoft Press, 1986.

Laudon, K. and J. Laudon. Information Systems: A Problem-Solving Approach. Fort Worth, TX: The Dryden Press, 1997.

Manes and Andrews. Gates: How Microsoft's Mogul Reinvented an Industry and Made Himself the Richest Man in America. New York: Doubleday, 1993.

Milenkovic. Operating Systems: Concept and Design. New York: McGraw-Hill, 1987.

Naur, P. and B. Randell (eds.). Software Engineering. NATO Scientific Affairs Division, Brussels. Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 October, 1998, 1969.

Orchard-Hays. "The Evolution of Programming Systems." IRE Proceedings, January 1961, pp. 285-295.

Organick, E.I. The Multics System: An Examination of the Structure, Cambridge, MA: MIT Press, 1972.

Pugh, E. , R.J. Lyle and H. Palmer. IBM's 360 and Early 370 Systems. Cambridge: MIT Press, 1991.

Ritchie, Dennis. "The Evolution of the UNIX Timesharing System." AT& T Bell Laboratories Technical Journal, August 1984, pp. 1577-1593.

Ritchie, D.M. and K. Thompson. "UNIX Timesharing System: The UNIX Timesharing System." Bell System Technical Journal, August 1978, pp. 1991-2019.

Rosin, R.F. "Supervisory and Monitor System." ACM Computing Surveys, March 1969.

Salus, P. A Quarter Century of UNIX. Reading, MA: Addison-Wesley, 1994.

Slater, R. Portraits in Silicon. Cambridge, MA: MIT Press, 1987.

Smith, A.J. "Multiprogramming and Memory Contention." Software – Practice and Experience, July 1980, pp. 531-552.

Smith, D.K. and R.C. Alexander. Fumbling the Future: How Xerox Invented, Then Ignored, the First Personal Computer. New York: Morrow, 1988.

Stern, N. From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computer. Bedford, MA: Digital Press, 1981.

Veit, S. History of Personal Computer. Asheville, NC: WorldComn, 1993.

Wallace, J. and J. Erickson. Hard Drive: Bill Gates and the Making of the Microsoft Empire. New York: John Wiley, 1992.

Watson, Jr., T. & P. Petre. Father and Son & Co. London: Bantam Press, 1990.

Weizer, N. "A History of Operating Systems." Datamation, January 1981, pp. 119-126.

Williams, G. and R. Moore. "The Apple Story, Part 2." Byte, January 1985, pp. 167-180.

Young, Jeffrey. Steve Jobs: The Journey Is the Reward. Glenview, IL: Scott, Foresman, 1988.