Module I
OPERATING SYSTEM

An operating system (OS) is the program that, after being initially loaded into the computer by a boot program, manages all the other programs in a computer. The other programs are called applications or application programs. The application programs make use of the operating system by making requests for services through a defined application program interface (API). In addition, users can interact directly with the operating system through a user interface such as a command line or a graphical user interface (GUI).

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. Some popular Operating Systems include Linux, Windows, UNIX, MS-DOS, MS-Windows - 98/XP/Vista, Windows-NT/2000, OS/2 and Mac OS, OS X, VMS, OS/400, AIX, z/OS, etc.

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. All computer programs, excluding firmware, require an operating system to function.

The operating system (OS) is the most important program that runs on a computer. Every general-purpose computer must have an operating system to run other programs and applications. Computer operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the storage drives, and controlling peripheral devices, such as printers.

For large systems, the operating system has even greater responsibilities and powers. It is like a traffic cop, it makes sure that different programs and users running at the same time do not interfere with each other. The operating system is also responsible for security, ensuring that unauthorized users do not access the system.

An operating system, or "OS," is software that communicates with the hardware and allows other programs to run. It is comprised of system software, or the fundamental files our computer needs to boot up and function. Every desktop computer, tablet, and Smartphone includes an operating system that provides basic functionality for the device.

Operating System is software that works as an interface between a user and the computer hardware. The primary objective of an operating system is to make computer system convenient to use and to utilize computer hardware in an efficient manner. The operating system performs the basic tasks such as receiving input from the keyboard, processing instructions and sending output to the screen. Operating system is software that is required in order to run application programs and utilities. It works as a bridge to perform better interaction between application programs and hardware of the computer.

Objectives of Operating Systems

- **Convenience**: An OS makes a computer more convenient to use.
- **Efficiency**: An OS allows the computer system resources to be used in an efficient manner.
- **Ability to evolve**: An OS should be constructed in such a way as to permit the effective development, testing, and introduction of new system functions without interfering with service.

The other objectives are:

- To make the computer system convenient to use in an efficient manner.
- To hide the details of the hardware resources from the users.
- To provide users a convenient interface to use the computer system.
- To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
- To manage the resources of a computer system.
- To keep track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.

- To provide efficient and fair sharing of resources among users and programs.

OS as resource manager – Efficiency

It is not the OS itself but the hardware that makes all kinds of services possible and available to application programs. An OS merely exploits the hardware to provide easily accessible interfaces. Exploitation means management upon the hardware resources, and thus also imposes control upon or manages the entities that use the services so that the resources are used efficiently. In the classes later on, we will discuss this aspect, including process scheduling, memory management, I/O device management, etc. One thing worth mentioning here is that, different from other control systems where the controlling facility, the controller, is distinct and external to the controlled parts, the OS has to depend on the hardware resources it manages to work.

As we know, an OS is in nature a program, consisting instructions, thus it also needs CPU to execute instructions so as to function as a controller, and main memory to hold instructions for CPU to fetch. At the same time, the OS has to be able to relinquish and regain later the control of CPU so that other programs can get chance to run but still under the control of the OS. By utilizing the facilities provided by hardware, the OS may schedule different processes to run at different moments and exchange the instructions and data of programs between external storage devices, like hard disks, and main memory. These topics will be covered as the course proceeds.

Functions of Operating Systems

Following are some of important functions of an operating System.

1. Memory Management
2. Processor Management
3. Device Management
4. File Management
5. Security
6. Control over system performance
7. Job accounting
8. Error detecting aids
9. Coordination between other software and users

Memory Management

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management.

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part is not in use?
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor management.

- Keeps tracks of processor and status of process. The program responsible for this task is known as traffic controller.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management

An Operating System manages device communication via their respective drivers. It does the following activities for device management

- Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

File Management

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

An Operating System does the following activities for file management.

- Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Activities

- **Security** − By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- **Control over system performance** − Recording delays between request for a service and response from the system.
- **Job accounting** − Keeping track of time and resources used by various jobs and users.
- **Error detecting aids** − Production of dumps, traces, error messages, and other debugging and error detecting aids.
- **Coordination between other softwares and users** − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

The Evolution of Operating Systems

Point out that the evolution of operating system software parallels the evolution of the computer hardware they were designed to control.

1940s

1. Provide students with an overview of first-generation computers, which were based on vacuum tube technology. Point out that there was no standard operating system software at that time.

2. Mention that a typical program included every instruction needed by the computer to perform the tasks requested, and the machines were poorly utilized, i.e., the CPU processed data and made calculations for only a fraction of the available time. Basically, early programs were designed to

use the resources conservatively at the expense of understand ability.

1945: ENIAC, Moore School of Engineering, University of Pennsylvania.
1949: EDSAC and EDVAC
1949 BINAC - a successor to the ENIAC

1950s

1. Provide students with an overview of second-generation computers (1955-1965). Point out that business environments placed importance on cost effectiveness; however, computers were still very expensive.

2. Outline two major improvements that were widely adopted: computer operators were hired to facilitate each machine's operation, and job scheduling was instituted.

3. Use examples to show how job scheduling helps improve productivity. Point out that job scheduling introduced the need for control cards, which defined the exact nature of each program and its requirements. Discuss job control language (JCL).

4. Discuss various factors that helped improve the performance of the CPU, such as an increase in the speed of I/O devices, the development of access methods, the introduction of buffers and timer interrupts, etc.

5. Point out that during this time, techniques were developed to manage program libraries, create and maintain data files and indexes, randomize direct access addresses, and create and check file labels. However, programs were still run in serial batch mode, one at a time.

1951: UNIVAC by Remington
1952: IBM 701
1956: The interrupt
1954-1957: FORTRAN was developed

1960s

1. Provide students with an overview of third-generation computers dated from the mid-1960s. Point out that they were designed with faster CPUs, but their speed caused problems when they interacted with the relatively slow I/O devices.

2. Explain how the concept of multiprogramming helped solve this problem. Discuss the mechanism of its implementation.

3. Use examples to explain the concepts of passive multiprogramming and active multiprogramming. Point out the disadvantages of passive multiprogramming and how these were overcome by active multiprogramming.

4. Point out that in this generation, few advances were made in data management, and the total operating system was customized to suit users' needs.

1960s: Disks become mainstream
1961: The dawn of minicomputers
1962 Compatible Time-Sharing System (CTSS) from MIT
1963 Burroughs Master Control Program (MCP) for the B5000 system
1964: IBM System/360
1966: Minicomputers get cheaper, more powerful, and really useful
1967-1968: The mouse
1964 and onward: Multics
1969: The UNIX Time-Sharing System from Bell Telephone Laboratories

1970s

1. Point out that during the late 1970s, computers had faster CPUs, thus creating an even greater disparity between their rapid processing speed and slower I/O access time. Multiprogramming schemes to increase CPU use were limited by physical capacity of main memory.
2. Discuss how the concept of virtual memory solved the physical limitation issue.

3. Point out some other important developments during this time: database management software became a popular tool; a number of query systems were introduced; and programs started using English-like words, modular structures, and standard operations.

4. Discuss the implications this supercomputer had on computer systems and operating system design.

- Multi User and Multi tasking was introduced.
- Dynamic address translation hardware and Virtual machines came into picture.
- Modular architectures came into existence.
- Personal, interactive systems came into existence.

1971: Intel announces the microprocessor
1972: IBM comes out with VM: the Virtual Machine Operating System
1973: UNIX 4th Edition is published
1973: Ethernet
1974 The Personal Computer Age begins
1974: Gates and Allen wrote BASIC for the Altair
1976: Apple II

1980s

1. Discuss the various developments in the 1980s, such as improved cost/performance ratio of computer components, greater flexibility of hardware, and the introduction of the concept of firmware, etc.

2. Provide students with an overview of multiprocessing, which was introduced during this time and allowed the parallel execution of programs.

3. Point out that the evolution of personal computers and high-speed communications sparked the move to distributed processing and networked systems, enabling users in remote locations to share hardware and software resources.

4. Provide students with an overview of network operating systems and distributed operating systems.

- August 12, 1981: IBM introduces the IBM PC
- 1983 Microsoft begins work on MS-Windows
- 1984 Apple Macintosh comes out

1990s

1. Point out that the demand for Internet capability in the mid-1990s sparked the proliferation of networking capability. The World Wide Web, conceived by Tim Berners-Lee, made the Internet accessible by computer users worldwide.

2. Be sure to note that increased networking also created increased demand for tighter security to protect hardware and software.

3. Point out that the decade also introduced a proliferation of multimedia applications demanding additional power, flexibility, and device compatibility for most operating systems.

4. Tim Berners-Lee, the person who is credited with designing the first World Wide Web server. Discuss the implications of this design on today's computing environments.

1990 Microsoft Windows 3.0 comes out
1991 GNU/Linux
1992 The first Windows virus comes out
1993 Windows NT

2000s

1. Point out that primary design features of current operating systems are based on providing support for various services such as multimedia, Internet and Web access, and client/server computing, etc. Outline various requirements of computer systems in order to meet these demands, such as increased CPU speed, high-speed network attachments, and increased number and variety of storage devices.
2. Explain the process of virtualization.
3. Discuss recent advances in processing speeds.

2007: iOS
2008: Android OS

Serial Processing:
- Early computer from late 1940 to the mid 1950.
- The programmer interacted directly with the computer hardware.
- These machine are called bare machine as they don't have OS.
- Every computer system is programmed in its machine language.
- Uses Punch Card, paper tapes and language translator

These system presented two major problems.
1. Scheduling
2. Set up time

Scheduling: Used signup sheet to reserve machine time. A user may sign up for an hour but finishes his job in 45 minutes. This would result in wasted computer idle time, also the user might run into the problem not finish his job in allotted time.

Set up time: A single program involves:
- Loading compiler and source program in memory
- Saving the compiled program (object code)
- Loading and linking together object program and common function

Each of these steps involves the mounting or dismounting tapes on setting up punch cards. If an error occur user had to go the beginning of the set up sequence. Thus, a considerable amount of time is spent in setting up the program to run. This mode of operation is turned as serial processing ,reflecting the fact that users access the computer in series. Simple

Simple batch Systems

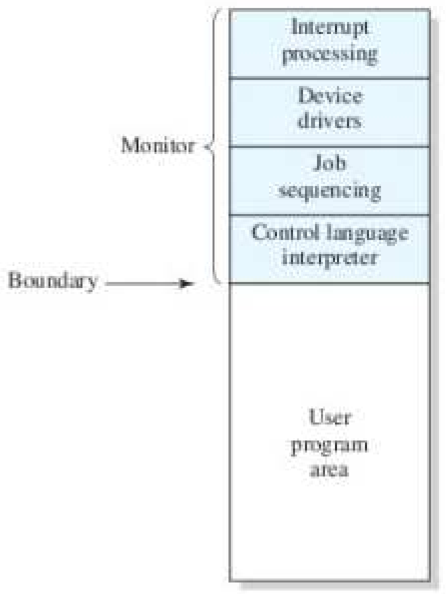
The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems with Batch Systems are as follows

- Lack of interaction between the user and the job.
- CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
- Difficult to provide the desired priority.
- Early computers were very expensive, and therefore it was important to maximize processor utilization.
- The wasted time due to scheduling and setup time in Serial Processing was unacceptable.
- To improve utilization, the concept of a batch operating system was developed.

Batch is defined as a group of jobs with similar needs. The operating system allows users to form batches. Computer executes each batch sequentially, processing all jobs of a batch considering them as a single process called batch processing.

The central idea behind the simple batch-processing scheme is the use of a piece of software known as the monitor. With this type of OS, the user no longer has direct access to the processor. Instead, the user submits the job on cards or tape to a computer operator, who batches the jobs together sequentially and places the entire batch on an input device, for use by the monitor. Each program is constructed to branch back to the monitor when it completes processing, at which point the monitor automatically begins loading the next program.



Memory Layout for resident memory

With a batch operating system, processor time alternates between execution of user programs and execution of the monitor. There have been two sacrifices: Some main memory is now given over to the monitor and some processor time is consumed by the monitor. Both of these are forms of overhead.

Multi Programmed batch Systems

A single program cannot keep either CPU or I/O devices busy at all times. Multiprogramming increases CPU utilization by organizing jobs in such a manner that CPU has always one job to execute. If computer is required to run several programs at the same time, the processor could be kept busy for the most of the time by switching its attention from one program to the next. Additionally I/O transfer could overlap the processor activity i.e, while one program is waiting for an I/O transfer; another program can use the processor. So CPU never sits idle or if comes in idle state then after a very small time it is again busy.

In multi-programmed batched operating systems, the operating system reads jobs from disk drives where a list of jobs is already being stored through card readers. The operating system then pulls and stores as much job as it can in the memory. Then from the memory, operating system start working on a job. Now, whenever a job reaches a situation where is has to be waiting for one or more tasks to be completed like use of any IO devices, the operating system pulls another job from the memory and starts working on it. Whenever this job also starts waiting, for example it need to use the same IO which is already in use by its previous job, the operating systems pulls another job. This is how, a multi-programmed batched systems harness the power of disk drives and memory.

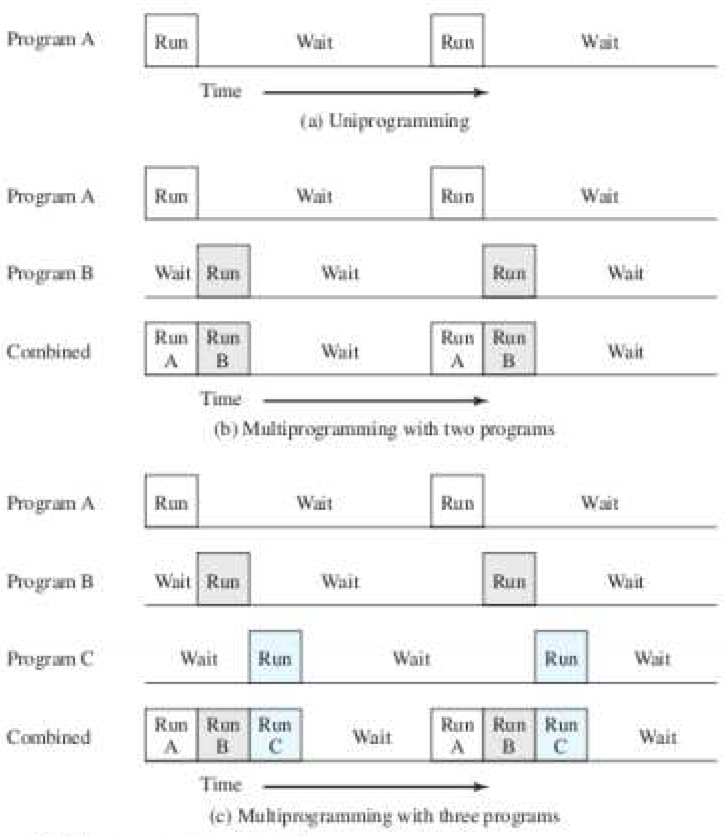
In multiprogramming, multiple programs (or jobs) of different users can be executed simultaneously (i.e. at the same time). The multiple jobs that have to be run simultaneously must be kept in main memory and the operating system must manage them properly. If these jobs are ready to run, the processor must decide which one to run.

In multi-programmed batch system, the operating system keeps multiple jobs in main memory at a time. There may be many jobs that enter the system. Since in general, the main memory is too small to accommodate all jobs. So the jobs that enter the system to be executed are kept initially on the disk in the job pool. In other words, we can say that a job pool consists of all jobs residing on the disk awaiting allocation of main memory. When the operating system selects a job from a job pool, it loads that job into memory for execution.

Normally, the jobs in main memory are smaller than the jobs in job pool. The jobs in job pool are awaiting allocation of main memory. If several jobs are ready to be brought into memory, and if there is not enough room for all of them, then the system must require memory management. Similarly, if many jobs are ready to run at the same time, the system must schedule these jobs.

The processor picks and begins to execute one of the jobs in main memory. Some jobs may have to wait for certain tasks (such as I/O operation), to complete. In a simple batch system or non-multi-programmed system, the processor would sit idle. In multi-programmed system, the CPU switches to second job and begins to execute it. Similarly, when second job needs to wait, the processor is switched to third job, and so on. The processor also checks the status of previous jobs, whether they are completed or not.

The multi-programmed system takes less time to complete the same jobs than the simple batch system. The multi-programmed systems do not allow interaction between the processes (or jobs) when they are running on the computer. Multiprogramming increases the CPU's utilization. Multi-programmed system provides an environment in which various computer resources are utilized effectively. The CPU always remains busy to run one of the jobs until all jobs complete their execution. In multi-programmed system, the hardware must have the facilities to support multiprogramming.



Multiprogramming example

Sl.No.	Simple Batched Systems	Multi-programmed Batched Systems
1	In this system, processes are processed one after another	In this system, multiple processes can be executed at a time.
2	As one process gets processed at a time, it performs low.	Processes are executed in a parallel fashion, thus it is faster.
3	CPU remains in idle states for long times.	CPU do not need to remain in idle state.
4	Example: CP/M, MS DOS, PC DOS etc.	Example: Windows 95, MaxOS etc.

Time Sharing Systems

Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of Multiprogrammed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.

Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, the processor executes each user program in a short burst or quantum of computation. That is, if users are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most.

The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

- Multiprogramming didn't provide the user interaction with the computer system.
- Time sharing or Multitasking is a logical extension of Multiprogramming that provides user interaction.
- There are more than one user interacting the system at the same time
- The switching of CPU between two users is so fast that it gives the impression to user that he is only working on the system but actually it is shared among different users.
- CPU bound is divided into different time slots depending upon the number of users using the system.
- Just as multiprogramming allows the processor to handle multiple batch jobs at a time, multiprogramming can also be used to handle multiple interactive jobs. In this latter case, the technique is referred to as time sharing, because processor time is shared among multiple users
- A multitasking system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time shared computer. Each user has at least one separate program in memory.
- Multitasking are more complex than multiprogramming and must provide a mechanism for jobs synchronization and

communication and it may ensure that system does not go in deadlock.

Although batch processing is still in use but most of the system today available uses the concept of multitasking and Multiprogramming.

Advantages

- Provides the advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

Parallel Systems

Parallel Processing Systems are designed to speed up the execution of programs by dividing the program into multiple fragments and processing these fragments simultaneously. Such systems are multiprocessor systems also known as tightly coupled systems. Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both.

In computers, parallel processing is the processing of program instructions by dividing them among multiple processors with the objective of running a program in less time. In the earliest computers, only one program ran at a time. A computation-intensive program that took one hour to run and a tape copying program that took one hour to run would take a total of two hours to run. An early form of parallel processing allowed the interleaved execution of both programs together. The computer would start an I/O operation, and while it was waiting for the operation to complete, it would execute the processor-intensive program. The total execution time for the two jobs would be a little over one hour.

Parallel computing is an evolution of serial computing where the jobs are broken into discrete parts that can be executed concurrently. Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different CPUs.

Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized. Several models for connecting processors and memory modules exist, and each topology requires a different programming model. The three models that are most commonly used in building parallel computers include synchronous processors each with its own memory, asynchronous processors each with its own memory and asynchronous processors with a common, shared memory. Flynn has classified the computer systems based on parallelism in the instructions and in the data streams. These are:

1. Single instruction stream, single data stream (SISD).
2. Single instruction stream, multiple data stream (SIMD).
3. Multiple instruction streams, single data stream (MISD).
4. Multiple instruction stream, multiple data stream (MIMD).

Distributed Systems

Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.

The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

Advantages

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.

- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Real Time Systems.

A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the response time. So in this method, the response time is very less as compared to online processing.

Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

There are two types of real-time operating systems.

1. **Hard real-time systems**

Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.

2. **Soft real-time systems**

Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects likes undersea exploration and planetary rovers, etc.