Unit – 1

➢ Operating system

An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs. The operating system is a vital component of the system software in a computer system. This tutorial will take you through step by step approach while learning Operating System concepts.

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 Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

Following are some of important functions of an operating System:

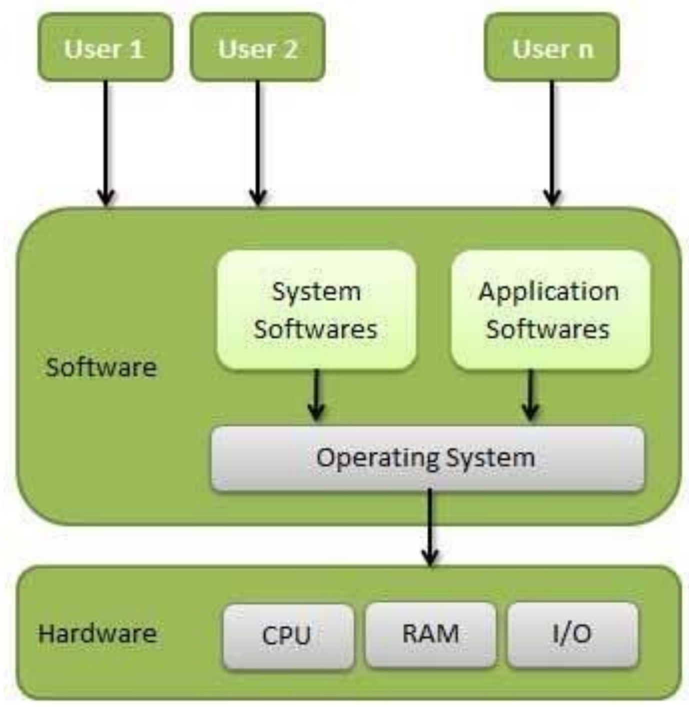
- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

➢ Applications of Operating System

Following are some of the important activities that an Operating System performs −

- **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.

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



For explanation of types of Operating system – refer hand written notes.

Advantages and disadvantages of the types of operating systems

Advantages of Batch Operating System:

- It is very difficult to guess or know the time required by any job to complete. Processors of the batch systems know how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time for batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometime costly
- The other jobs will have to wait for an unknown time if any job fails

Advantages of Time-Sharing OS:

- Each task gets an equal opportunity
- Less chances of duplication of software
- CPU idle time can be reduced

Disadvantages of Time-Sharing OS:

- Reliability problem
- One must have to take care of security and integrity of user programs and data
- Data communication problem

Advantages of Distributed Operating System:

- Failure of one will not affect the other network communication, as all systems are independent from each other
- Electronic mail increases the data exchange speed
- Since resources are being shared, computation is highly fast and durable
- Load on host computer reduces
- These systems are easily scalable as many systems can be easily added to the network
- Delay in data processing reduces

Disadvantages of Distributed Operating System:

- Failure of the main network will stop the entire communication
- To establish distributed systems the language which are used are not well defined yet
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet

Advantages of Network Operating System:

- Highly stable centralized servers
- Security concerns are handled through servers
- New technologies and hardware up-gradation are easily integrated to the system
- Server access are possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- Servers are costly
- User has to depend on central location for most operations
- Maintenance and updates are required regularly

Advantages of RTOS (real time operating system):

- **Maximum Consumption:** Maximum utilization of devices and system,thus more output from all the resources
- **Task Shifting:** Time assigned for shifting tasks in these systems are very less. For example in older systems it takes about 10 micro seconds in shifting one task to another and in latest systems it takes 3 micro seconds.
- **Focus on Application:** Focus on running applications and less importance to applications which are in queue.
- **Real time operating system in embedded system:** Since size of programs are small, RTOS can also be used in embedded systems like in transport and others.
- **Error Free:** These types of systems are error free.
- **Memory Allocation:** Memory allocation is best managed in these type of systems.

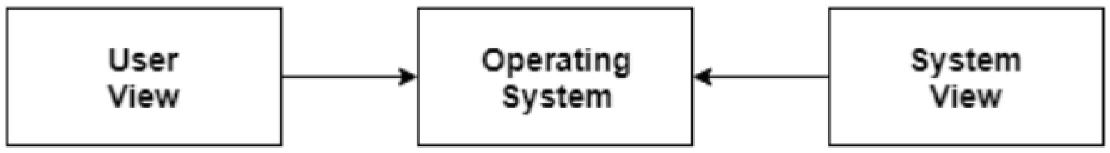
Disadvantages of RTOS:

- **Limited Tasks:** Very few tasks run at the same time and their concentration is very less on few applications to avoid errors.
- **Use heavy system resources:** Sometimes the system resources are not so good and they are expensive as well.
- **Complex Algorithms:** The algorithms are very complex and difficult for the designer to write on.
- **Device driver and interrupt signals:** It needs specific device drivers and interrupt signals to response earliest to interrupts.
- **Thread Priority:** It is not good to set thread priority as these systems are very less prone to switching tasks.

Examples of Real-Time Operating Systems are: Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

➢ Operating system – System view and user view

The operating system can be observed from the point of view of the user or the system. This is known as the user view and the system view respectively. More details about these are given as follows −



User View

The user view depends on the system interface that is used by the users. The different types of user view experiences can be explained as follows −

- If the user is using a personal computer, the operating system is largely designed to make the interaction easy. Some attention is also paid to the performance of the system, but there is no need for the operating system to worry about resource utilization. This is because the personal computer uses all the resources available and there is no sharing.
- If the user is using a system connected to a mainframe or a minicomputer, the operating system is largely concerned with resource utilization. This is because there may be multiple terminals connected to the mainframe and the operating system makes sure that all the resources such as CPU,memory, I/O devices etc. are divided uniformly between them.
- If the user is sitting on a workstation connected to other workstations through networks, then the operating system needs to focus on both individual usage of resources and sharing though the network. This happens because the workstation exclusively uses its own resources but it also needs to share files etc. with other workstations across the network.

- If the user is using a handheld computer such as a mobile, then the operating system handles the usability of the device including a few remote operations. The battery level of the device is also taken into account.

There are some devices that contain very less or no user view because there is no interaction with the users. Examples are embedded computers in home devices, automobiles etc.

System View

According to the computer system, the operating system is the bridge between applications and hardware. It is most intimate with the hardware and is used to control it as required.

The different types of system view for operating system can be explained as follows:

- The system views the operating system as a resource allocator. There are many resources such as CPU time, memory space, file storage space, I/O devices etc. that are required by processes for execution. It is the duty of the operating system to allocate these resources judiciously to the processes so that the computer system can run as smoothly as possible.
- The operating system can also work as a control program. It manages all the processes and I/O devices so that the computer system works smoothly and there are no errors. It makes sure that the I/O devices work in a proper manner without creating problems.
- Operating systems can also be viewed as a way to make using hardware easier.
- Computers were required to easily solve user problems. However it is not easy to work directly with the computer hardware. So, operating systems were developed to easily communicate with the hardware.
- An operating system can also be considered as a program running at all times in the background of a computer system (known as the kernel) and handling all the application programs. This is the definition of the operating system that is generally followed.

➢ History of Operating system

The 1940's - First Generations

The earliest electronic digital computers had no operating systems. Machines of the time were so primitive that programs were often entered one bit at time on rows of mechanical switches (plug boards). Programming languages were unknown (not even assembly languages). Operating systems were unheard of .

The 1950's - Second Generation

By the early 1950's, the routine had improved somewhat with the introduction of punch cards. The General Motors Research Laboratories implemented the first operating systems in early 1950's for their IBM 701. The system of the 50's generally ran one job at a time. These were called single-stream batch processing systems because programs and data were submitted in groups or batches.

The 1960's - Third Generation

The systems of the 1960's were also batch processing systems, but they were able to take better advantage of the computer's resources by running several jobs at once. So operating systems designers developed the concept of multiprogramming in which several jobs are in main memory at once; a processor is switched from job to job as needed to keep several jobs advancing while keeping the peripheral devices in use.

For example, on the system with no multiprogramming, when the current job paused to wait for other I/O operation to complete, the CPU simply sat idle until the I/O finished. The solution for this problem that evolved was to partition memory into several pieces, with a different job in each partition. While one job was waiting for I/O to complete, another job could be using the CPU.

Another major feature in third-generation operating system was the technique called spooling (simultaneous peripheral operations on line). In spooling, a high-speed device like a disk interposed between a running program and a low-speed device involved with the program in input/output. Instead of writing directly to a printer, for example, outputs are written to the disk. Programs can run to completion faster, and other programs can be initiated sooner when the printer becomes available, the outputs may be printed.

Note that spooling technique is much like thread being spun to a spool so that it may be later be unwound as needed.

Another feature present in this generation was time-sharing technique, a variant of multiprogramming technique, in which each user has an on-line (i.e., directly connected) terminal. Because the user is present and interacting with the computer, the computer system must respond quickly to user requests, otherwise user productivity could suffer. Timesharing systems were developed to multiprogram large number of simultaneous interactive users.

Fourth Generation

With the development of LSI (Large Scale Integration) circuits, chips, operating system entered in the system entered in the personal computer and the workstation age. Microprocessor technology evolved to the point that it become possible to build desktop computers as powerful as the mainframes of the 1970s. Two operating systems have dominated the personal computer scene: MS-DOS, written by Microsoft, Inc. for the IBM PC and other machines using the Intel 8088 CPU and its successors, and UNIX, which is dominant on the large personal computers using the Motorola 6899 CPU family.

Early Evolution

- 1945: **ENIAC**, Moore School of Engineering, University of Pennsylvania.
- 1949: **EDSAC** and **EDVAC**
- 1949: **BINAC** - a successor to the ENIAC
- 1951: **UNIVAC** by Remington
- 1952: **IBM** 701
- 1956: The interrupt

- 1954-1957: **FORTRAN** was developed

Operating Systems - Late 1950s

By the late 1950s Operating systems were well improved and started supporting following usages:

- It was able to perform **Single stream batch processing**.
- It could use Common, standardized, input/output routines for device access.
- Program transition capabilities to reduce the overhead of starting a new job was added.
- **Error recovery** to clean up after a job terminated abnormally was added.
- Job control languages that allowed users to specify the job definition and resource requirements were made possible.

Operating Systems - In 1960s

- 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
- 1960s: Disks became mainstream
- 1966: Minicomputers got cheaper, more powerful, and really useful.
- 1967-1968: **Mouse** was invented.
- 1964 and onward: Multics
- 1969: The UNIX Time-Sharing System from Bell Telephone Laboratories.

Supported OS Features by 1970s

- **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.

Accomplishments after 1970

- 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
- August 12, 1981: IBM introduces the IBM PC
- 1983 Microsoft begins work on MS-Windows
- 1984 Apple Macintosh comes out
- 1990 Microsoft Windows 3.0 comes out
- 1991 GNU/Linux
- 1992 The first Windows virus comes out
- 1993 Windows NT
- 2007: iOS
- 2008: Android OS

➢ **Multiprogramming, multitasking, multithreading and multiprocessing**

1. **Multiprogramming** – A computer running more than one program at a time (like running Excel and Firefox simultaneously).

2. **Multiprocessing** – A computer using more than one CPU at a time.

3. **Multitasking** – Tasks sharing a common resource (like 1 CPU).

4. **Multithreading** is an extension of multitasking.

1. Multi programming –
In a modern computing system, there are usually several concurrent application processes which want to execute. Now it is the responsibility of the Operating System to manage all the processes effectively and efficiently.
One of the most important aspects of an Operating System is to multi program.
In a computer system, there are multiple processes waiting to be executed, i.e. they are waiting when the CPU will be allocated to them and they begin their execution. These processes are also known as jobs. Now the main memory is too small to accommodate all of these processes or jobs into it. Thus, these processes are initially kept in an area called job pool. This job pool consists of all those processes awaiting allocation of main memory and CPU.
CPU selects one job out of all these waiting jobs, brings it from the job pool to main memory and starts executing it. The processor executes one job until it is interrupted by some external factor or it goes for an I/O task.

2. Multiprocessing –
In a uni-processor system, only one process executes at a time.
Multiprocessing is the use of two or more CPUs (processors) within a single Computer system. The term also refers to the ability of a system to support more than one processor within a single computer system. Now since there are multiple processors available, multiple processes can be executed at a time. These multi processors share the computer bus, sometimes the clock, memory and peripheral devices also.

3. Multitasking –
As the name itself suggests, multi tasking refers to execution of multiple tasks (say processes, programs, threads etc.) at a time. In the modern operating systems, we are able to play MP3 music, edit documents in Microsoft Word, surf the Google Chrome all simultaneously, this is accomplished by means of multi tasking.

Multitasking is a logical extension of multi programming. The major way in which multitasking differs from multi programming is that multi programming works solely on the concept of context switching whereas multitasking is based on time sharing alongside the concept of context switching.

➢ Context Switching

A context switching is a process that involves switching of the CPU from one process or task to another. In this phenomenon, the execution of the process that is present in the running state is suspended by the kernel and another process that is present in the ready state is executed by the CPU.

It is one of the essential features of the multitasking operating system. The processes are switched so fastly that it gives an illusion to the user that all the processes are being executed at the same time.

But the context switching process involved a number of steps that need to be followed. You can't directly switch a process from the running state to the ready state. You have to save the context of that process. If you are not saving the context of any process P then after some time, when the process P comes in the CPU for execution again, then the process will start executing from starting. But in reality, it should continue from that point where it left the CPU in its previous execution. So, the context of the process should be saved before putting any other process in the running state.

A context is the contents of a CPU's registers and program counter at any point in time.
Context switching can happen due to the following reasons:

- When a process of high priority comes in the ready state. In this case, the execution of the running process should be stopped and the higher priority process should be given the CPU for execution.
- When an interruption occurs then the process in the running state should be stopped and the CPU should handle the interrupt before doing something else.

- When a transition between the user mode and kernel mode is required then you have to perform the context switching.

➢ <u>System Calls in OS</u>

In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to **interact with the operating system**. A computer program makes a system call when it makes a request to the operating system's kernel. System call **provides** the services of the operating system to the user programs via Application Program Interface(API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

Services Provided by System Calls :
1. Process creation and management
2. Main memory management
3. File Access, Directory and File system management
4. Device handling(I/O)
5. Protection
6. Networking, etc.

Types of System Calls : There are 5 different categories of system calls –
1. **Process control:** end, abort, create, terminate, allocate and free memory.
2. **File management:** create, open, close, delete, read file etc.
3. Device management
4. Information maintenance
5. Communication

➢ <u>Operating System Structure</u>

A structure of an Operating System determines how it has been designed and how it functions. There are numerous ways of designing a new structure of an Operating system. In this post, we will learn about six combinations that have been tested and tried.

<u>TYPES OF OPERATING SYSTEM STRUCTURE (for diagrams of these structures refer PPT)</u>

- MONOLYTHIC STRUCTURE
- SIMPLE STRUCTURE
- LAYERED STRUCTURE

<u>Monolithic System structure in an Operating System</u>

In this organizational structure, the entire operating system runs as a single program in the kernel mode. An operating system is a collection of various procedures linked together in a binary file. In this system, any procedure can call any other procedure. Since it is running in kernel mode itself, it has all the permissions to call whatever it wants.

In terms of information hiding, there is none. All procedures are running in kernel mode, so they have access to all modules and packages of other procedures.

However, using this approach without any restrictions can lead to thousands of procedure calls, and this can lead to a messy system. For this purpose, the actual OS is constructed in a hierarchy. All the individual procedures are compiled into a single executable file using the system linker.

Even a monolithic system has a structure in which it can run in user mode. There already is a basic structure given by the organization

1. The main procedure that invokes the requested service procedures.
2. A set of service procedures that carry out system calls.
3. A set of utility procedures that help out the system procedures.

<u>Layered Systems Structure in Operating Systems</u>

As the name suggests, this system works in layers.

Working

There are six layers in the system, each with different purposes.

Layer	**Function**
5	The operator
4	User Programs
3	Input/Output Management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Layer 0 – Processor Allocation and Multiprogramming – This layer deals with the allocation of processor, switching between the processes when interrupts occur or when the timers expire.

The sequential processes can be programmed individually without having to worry about other processes running on the processor. That is, layer 0 provides that basic multiprogramming of the CPU

Layer 1 – Memory and Drum Management – This layer deals with allocating memory to the processes in the main memory. The drum is used to hold parts of the processes (pages) for which space couldn't be provided in the main memory. The processes don't have to worry if there is available memory or not as layer 1 software takes care of adding pages wherever necessary.

Layer 2 – Operator-Process communication – In this layer, each process communicates with the operator (user) through the console. Each process has its own operator console and can directly communicate with the operator.

Layer 3 – Input/Output Management – This layer handles and manages all the I/O devices, and it buffers the information streams that are made available to it. Each process can communicate directly with the abstract I/O devices with all of its properties.

Layer 4 – User Programs – The programs used by the user are operated in this layer, and they don't have to worry about I/O management, operator/processes communication, memory management, or the processor allocation.

Layer 5 – The Operator – The system operator process is located in the outer most layer.

Simple Structure

There are many operating systems that have a rather simple structure. These started as small systems and rapidly expanded much further than their scope. A common example of this is MS-DOS. It was designed simply for a niche amount for people. There was no indication that it would become so popular.

- **Client-Server Model in Operating Systems**

The client-server model in an operating system is a variation of the microkernel system. The middle layer in the microkernel system is the one with servers. These servers provide some kind of service to clients. This makes up the client-server model.

Communication between clients and servers is obtained by message passing. To receive a service, one of the client processes constructs a message saying what it wants and sends it to the appropriate service. The service then does it work and sends back the answer.

If the clients and servers are on the same machine, then some optimizations are possible. But generally speaking, they are on different systems and are connected via a network link like LAN or WAN.