CHAPTER-1

- INTRODUCTION
- What Operating Systems Do
- Operating System Structure
- Operating System Operations
- Process Management
- Memory Management
- Storage Management

Operating System

- An operating system acts as an intermediary between the user of a computer and computer hardware.
- The purpose of an operating system is to provide an environment in which a user can execute programs conveniently and efficiently.
- An operating system is software that manages computer hardware.
- The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.
- A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being application programs.

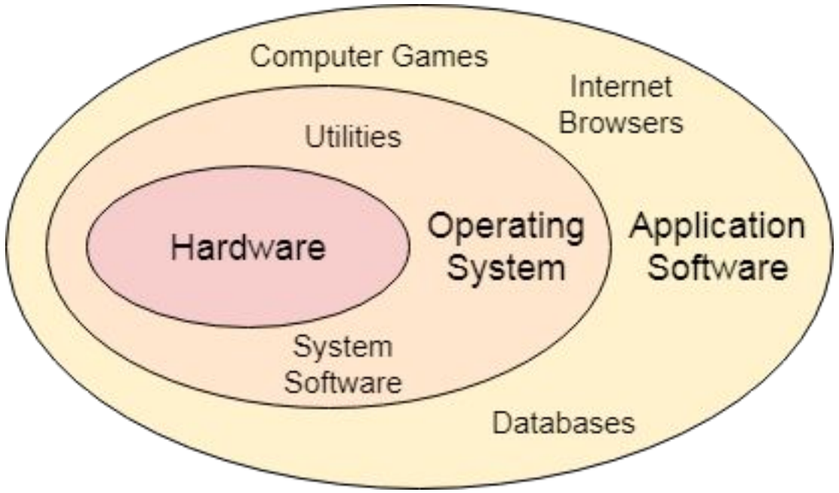
Different Types of Software

- Software: Set of tested programs with documentation (user manual)
- Application Software: this software is used by users to perform a specific task like MS word, VLC media player, etc.
- System Software: This software is used to manage the system resources like assembler, linker etc.

NOTE: An **Operating System** is a system software which act as an **interface between user and hardware**.

Why we need Operating System?

- In the Computer System (comprises of Hardware and software), Hardware can only understand machine code (in the form of 0 and 1) which doesn't make any sense to a naive user.
- We need a system which can act as an intermediary and manage all the processes and resources present in the system.



What is an OS?

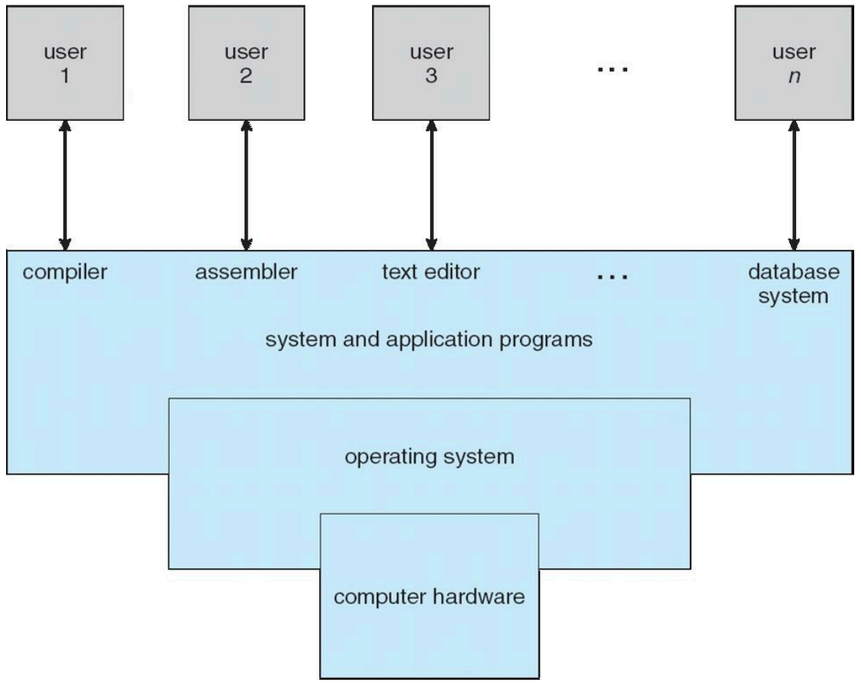
- An **Operating System** can be defined as an **interface between user and hardware**. It is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks.

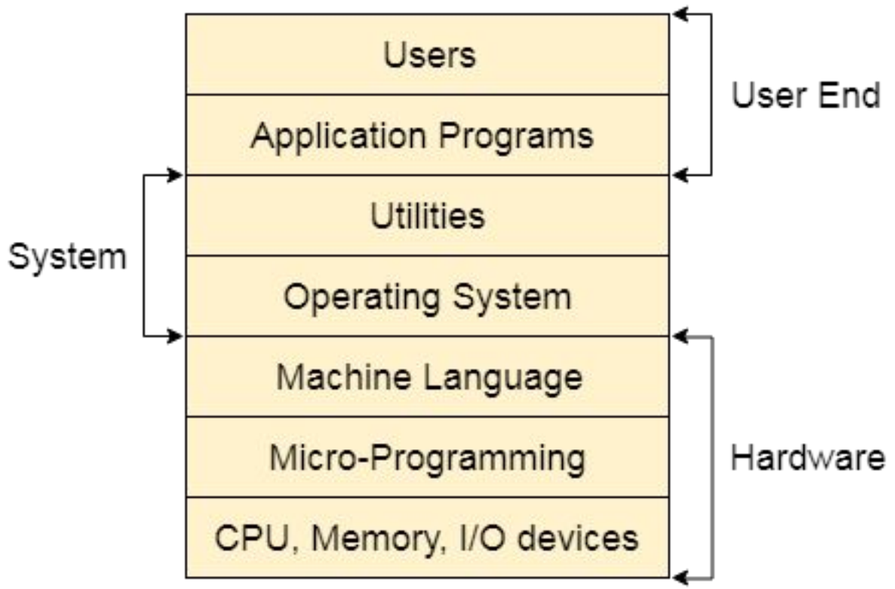
- The purpose of an operating system is to provide an environment in which a user can execute programs in convenient and efficient manner.

Components of a Computer System

A Computer System consists of:

- Users (people who are using the computer)
- Application Programs (Databases, Games, Video player, Browsers, etc.)
- System Programs (Shells, Editors, Compilers, etc.)
- Operating System (A special program which acts as an interface between user and hardware)
- Hardware (CPU, Disks, Memory etc.)





- In the above image, we can see that at level 0, the computer hardware are present and to access this hardware you need to take help from the Operating System which is present at level 1.
- At the upper level or at level 2, various application software (and system) are present.
- So, the Operating System is used for the communication of these Software's with the hardware.

Goals of the Operating System

Primary Goal: The primary goal of an Operating System is to provide a **user-friendly and convenient environment.**

We know that it is not compulsory to use the Operating System, but things become harder when the user has to perform all the process scheduling and converting the user code into machine code is also very difficult.

So, we make the use of an Operating System to act as an intermediate between user and the hardware. All you need to do is give commands to the Operating System and the Operating System will do the rest for you. So, the Operating System should be convenient to use.

Secondary Goal: The secondary goal of an Operating System is **efficiency**.

The Operating System should perform all the management of resources in such a way that the resources are fully utilized and no resource should be held idle if some request to that resource is there at that instant of time.

Note: So, in order to achieve the above primary and secondary goals, the Operating System performs a number of functions(will discuss in detail later on)

TYPES OF OS: Single-User/Single-Program OS

An operating system that allows a single user to perform only one task at a time is called a Single-User Single-Tasking Operating System.

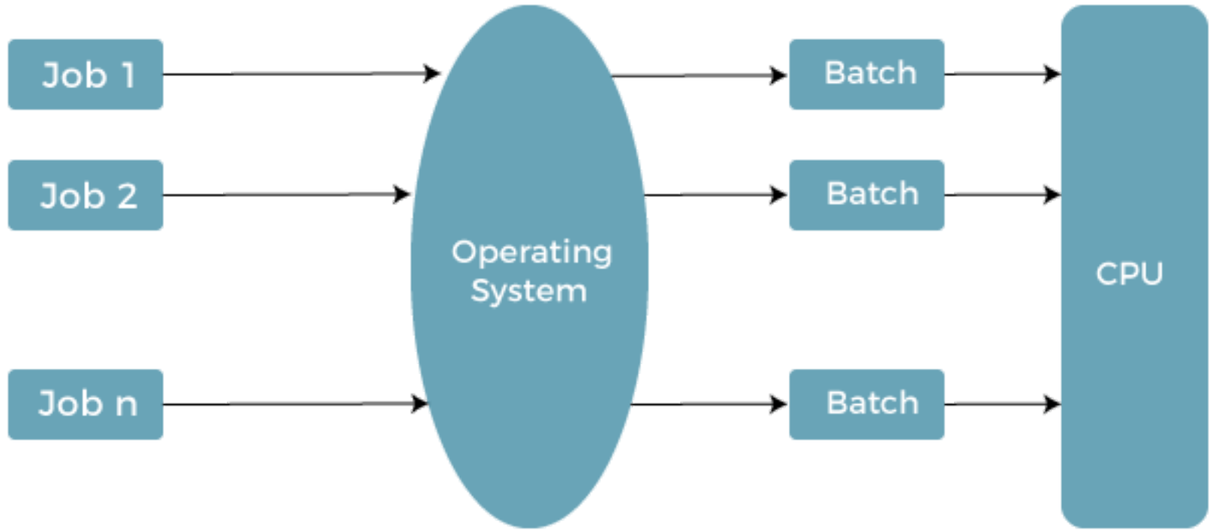
Functions like printing a document, downloading images, etc., can be performed only one at a time. Examples include MS-DOS, Palm OS, etc.

Advantages
This operating system occupies less space in memory.

Disadvantages
- It can perform only a single task at a time.
- Poor Utilization of CPU
- Poor Utilization of I/O devices

Batch Operating System

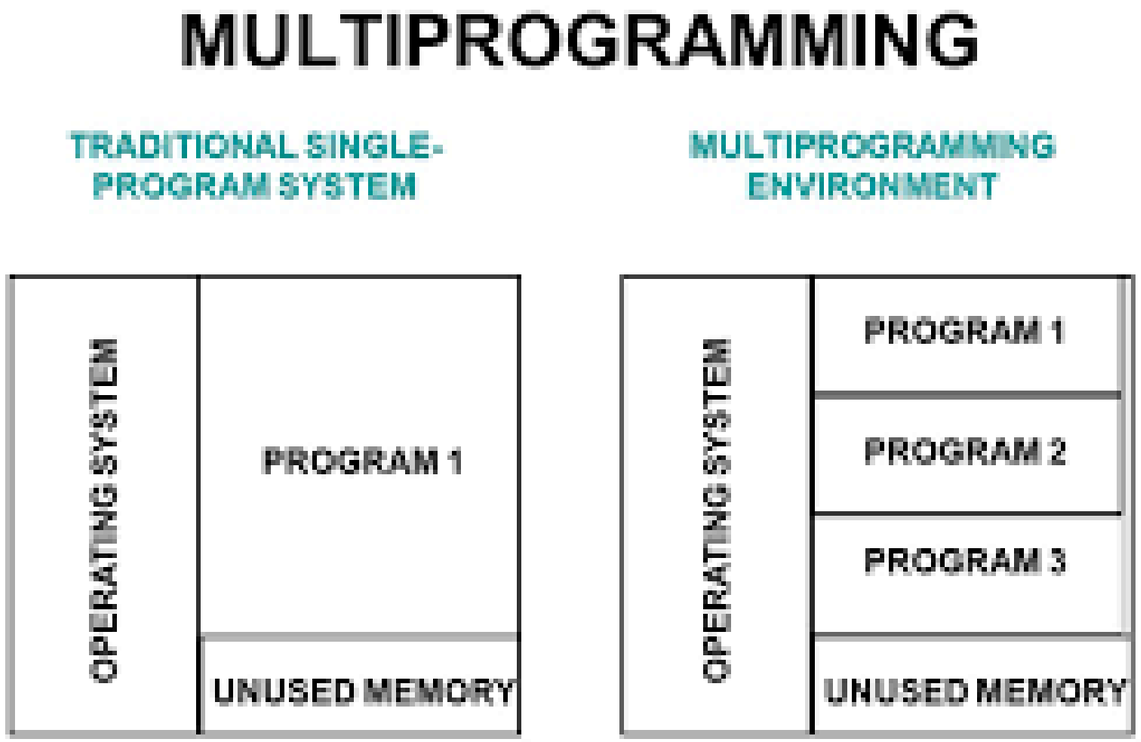
- The users of a batch operating system do not interact with the computer directly.
- Each user prepares his job on an off-line device like punch cards and submits it to the computer operator.
- To speed up processing, jobs with similar needs are batched together and run as a group.
- The programmers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



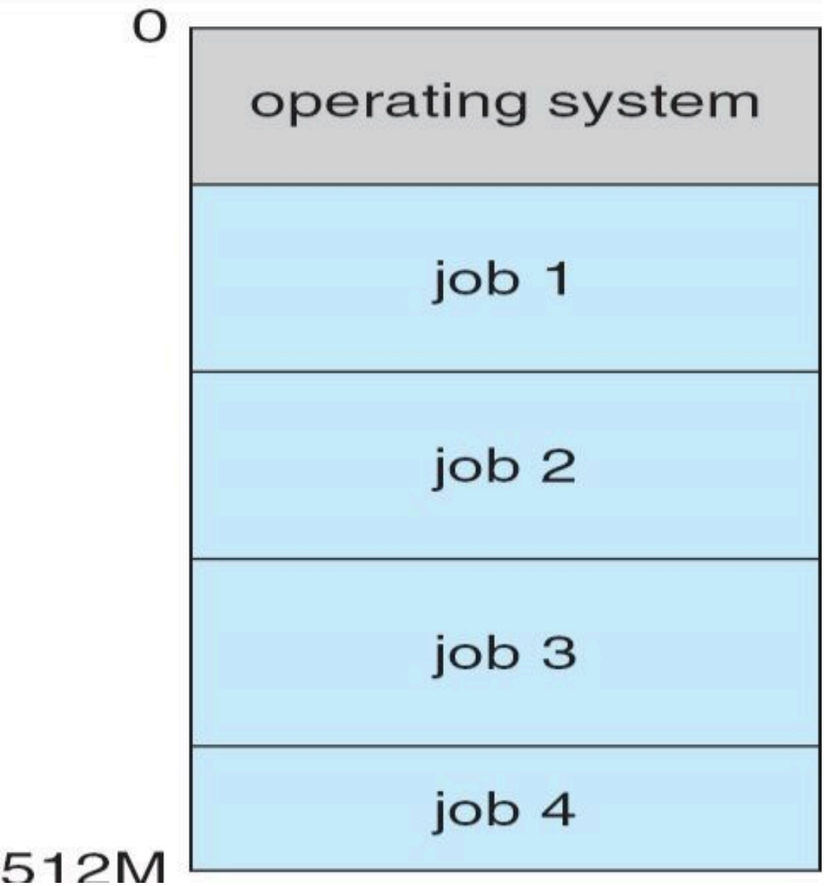
Disadvantages:

- 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.

Single Program Vs Multi program OS



Multiprogramming System



- When two or more programs are loaded in the memory at the same time, referred to the multiprogramming operating system.
- Multiprogramming assumes a single processor that is being shared. It increases CPU utilization by organizing jobs so that the CPU always has one to execute.
- The operating system keeps several jobs in memory at a time. This set of jobs is a subset of the jobs kept in the job pool. The operating system picks and begins to execute one of the job in the memory.

Multiprogramming System

- In this the operating system picks up and begins to execute one of the jobs from memory.
- Once this job needs an I/O operation operating system switches to another job (CPU and OS always busy).
- Jobs in the memory are always less than the number of jobs on disk(Job Pool).
- If several jobs are ready to run at the same time, then the system chooses which one to run through the process of **CPU Scheduling**.

In Non-multi programmed system, there are moments when CPU sits idle and does not do any work. In Multiprogramming system, CPU will never be idle and keeps on processing.

Multiprogramming System

Advantage

1. Maximise CPU utilization. Maximise I/O utilization.
2. It appears that many programs are allotted CPU almost simultaneously.

Disadvantages

1. CPU scheduling is requires.
2. To accommodate many jobs in memory, memory management is required.

Time-sharing operating systems

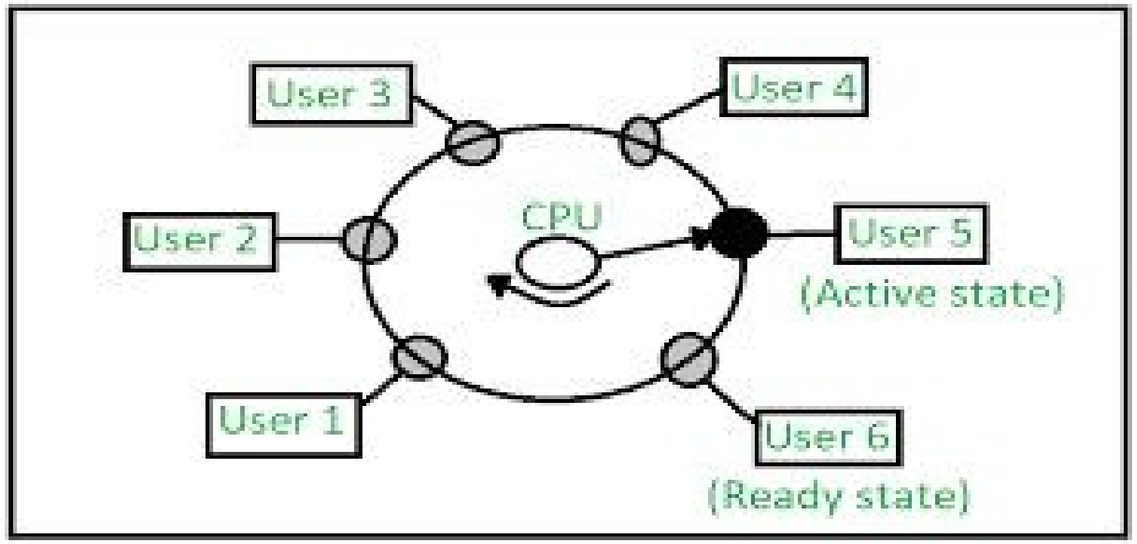
- Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. **Time-sharing** or **multitasking** is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed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

- ***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***.

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

- 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.

NOTE: **Response Time-The time taken by the system to respond to an input and display of required updated information.**



Advantages of Timesharing operating systems are as follows –

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

Disadvantages of Time-sharing operating systems are as follows –

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

Multiprocessing Operating system

- In operating systems, to improve the performance of more than one CPU can be used within one computer system called Multiprocessor operating system.
- Multiple CPUs are interconnected so that a job can be divided among them for faster execution.
- When a job finishes, results from all CPUs are collected and compiled to give the final output.
- Jobs needed to share main memory and they may also share other system resources among themselves.
- Multiple CPUs can also be used to run multiple jobs simultaneously.
- For Example: UNIX Operating system is one of the most widely used multiprocessing systems.

Symmetrical multiprocessing operating system:

- In a Symmetrical multiprocessing system, each processor executes the same copy of the operating system, takes its own decisions, and cooperates with other processes to smooth the entire functioning of the system.
- The CPU scheduling policies are very simple. Any new job submitted by a user can be assigned to any processor that is least burdened. It also results in a system in which all processors are equally burdened at any time.

Characteristics of Symmetrical multiprocessing operating system:

- In this system, any processor can run any job or process.
- In this, any processor initiates an Input and Output operation.

Asymmetric multiprocessing operating system

- In an asymmetric multiprocessing system, there is a master slave relationship between the processors.
- In which each processor is assigned a specific task. A master processor controls the system;
- the other processors(Slaves)either look to the master for instruction or have predefined tasks. This scheme defines a master-slave relationship.

Advantages of multi- processor system

- **Enhanced Throughput** If multiple processors are working in tandem, then the throughput of the system increases i.e. number of processes getting executed per unit of time increase.

- **More Economic Systems** Multiprocessor systems are cheaper than single processor systems in the long run because they share the data storage, peripheral devices, power supplies etc

- **More reliable Systems** In a multiprocessor system, even if one processor fails, the system will not halt.

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.

Users view cont..

- 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.

System view

- 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.

Dual Mode Operations

There are two modes of operation in the operating system to make sure it works correctly. These are user mode and kernel mode.

- **User Mode:** if a program is executing in user mode then that program does not have direct access to memory, hardware and such resources. The **mode bit is set to 1 in the user mode. It is changed from 1 to 0 when switching from user mode to kernel mode**.

- **Kernel Mode:** if a program is executing in kernel mode then that program has direct access to memory, hardware and such resources. It means program is executing in kernel mode then it is said to be in privileged mode because it is having direct access to memory and hardware. The mode bit is set to 0 in the kernel mode. It is changed from 0 to 1 when switching from kernel mode to user mode.

IMPORTANCE: if a program is executing in kernel mode and if that program happens to crash during execution then the entire system will crash or entire system will come to a halt. But if it is executing in user mode and if that program happens to crash during execution then the entire system does not crash or entire system does not come to a halt. That's why user mode is safer mode of operation and therefore most of the programs run in user mode.

NOTE: when a program is executing in user mode then it may needs to be access to some of the resources then it makes a call to OS telling that it needs to access certain resources via system calls. Then program switch from user mode to kernel mode known as context switch.

User Mode vs Kernel Mode

Note: Necessity of Dual Mode (User Mode and Kernel Mode) in Operating System(IMP)

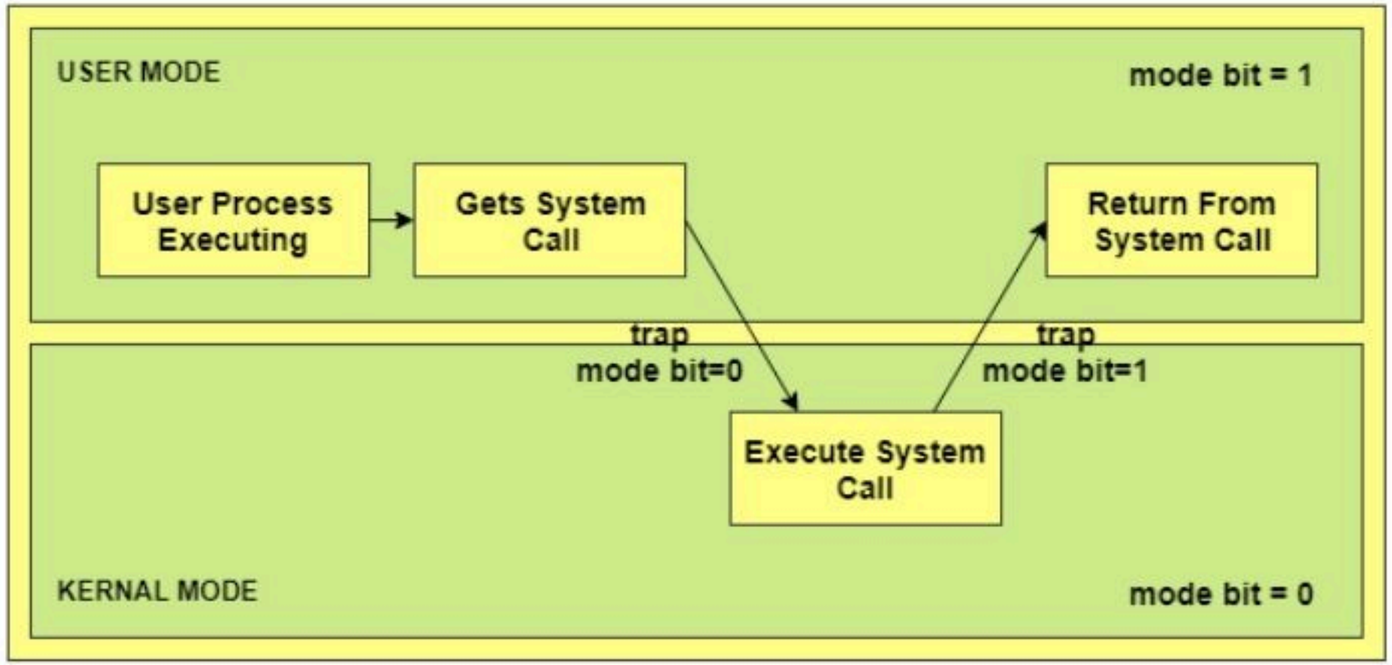
The lack of a dual mode i.e user mode and kernel mode in an operating system can cause serious problems. Some of these are −

1. In any Operating System, it is necessary to have **Dual Mode Operation** to ensure protection and security of the System from unauthorized or errant users . This Dual Mode separates the User Mode from the System Mode or Kernel Mode.
2. A running user program can accidentally wipe out the operating system by overwriting it with user data.
3. Multiple processes can write in the same system at the same time, with **disastrous results.**
4. These problems could have occurred in the MS-DOS operating system which had no mode bit and so no dual mode.

User Mode vs Kernel Mode

An image that illustrates the transition from user mode to kernel mode and back again is −

In this image, the user process executes in the user mode until it gets a system call. Then a system trap is generated and the mode bit is set to zero. The system call gets executed in kernel mode. After the execution is completed, again a system trap is generated and the mode bit is set to 1. The system control returns to kernel mode and the process execution continues.



Privileged Instructions

The Instructions that can **run only in Kernel Mode** are called Privileged Instructions.
Privileged Instructions possess the following characteristics :

I. **If any attempt is made to execute a Privileged Instruction in User Mode, then it will not be executed and treated as an illegal instruction**. The Hardware traps it to the Operating System.

II. Before transferring the control to any User Program, it is the responsibility of the Operating System to ensure that the **Timer** is set to interrupt. Thus, if the timer interrupts then the Operating System regains the control.

III. Thus, any instruction which can modify the contents of the Timer is a Privileged Instruction.

Privileged Instructions are used by the Operating System in order to achieve correct operation.

Various examples of Privileged Instructions include:

1. I/O instructions and Halt instructions
2. Turn off all Interrupts
3. Set the Timer Context Switching
4. Clear the Memory or Remove a process
5. Memory Modify entries in Device-status table

What are Non-Privileged Instructions?

The Instructions that can **run only in User Mode** are called Non-Privileged Instructions .

Various examples of Non-Privileged Instructions include:

1. Reading the status of Processor
2. Reading the System Time
3. Generate any Trap Instruction
4. Sending the final printout of Printer
5. Also, it is important to note that in order to change the mode from Privileged to Non-Privileged, we require a Non-privileged Instruction that does not generate any interrupt.

HARDWARE PROTECTION

CPU protection

- **We must ensure that the OS maintains control over the CPU. We can't allow a user program to get stuck in an infinite loop and never return control to the OS.**
- **To accomplish this goal we use TIMER. A timer can set to interrupt the computer after a specified period of time. The period may be fixed or variable timer.**
- **A variable timer is generally implemented by a fixed rate clock and a counter.**
- **The OS sets the counter and every time the clock ticks, the counter is decremented.**
- **When the counter reaches zero, an interrupt occurs.**
- **We can use time to prevent a user program from running too long by initializing a counter with a amount of time that a program is allowed to run.**
- Example: A program with 7 min. time limit, then would have its counter initialized to 420. Every second, the timer interrupts and the count is decremented by 1. As soon as the counter is +ve, control is returned to user program and when counter becomes – ve, the OS will terminate the program.

Functions of an Operating System: Memory Management

The operating system manages the Primary Memory or Main Memory.

Main memory is made up of a large **array of bytes or words where each byte or word is assigned a certain address.**

Main memory is fast storage and it can be accessed directly by the CPU.

For a program to be executed, it should be first loaded in the main memory.

An operating system manages the allocation and deallocation of memory to various processes and ensures that the other process does not consume the memory allocated to one process.

An Operating System performs the following activities for **Memory Management**:

- It keeps track of primary memory, i.e., which bytes of memory are used by which user program. The memory addresses that have already been allocated and the memory addresses of the memory that has not yet been used.

- In multiprogramming, the OS decides the order in which processes are granted memory access, and for how long.

- It Allocates the memory to a process when the process requests it and deallocates the memory when the process has terminated or is performing an I/O operation.

Process Management

A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity;*** process is an ***active entity***.

Process needs resources to accomplish its task
- CPU, memory, I/O, files
- Initialization data

Process termination requires reclaim of any reusable resources

Single-threaded process has one **program counter** specifying location of next instruction to execute
- Process executes instructions sequentially, one at a time, until completion

Multi-threaded process has one program counter per thread

Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
- Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

File Management

File management in an operating system refers to the processes and techniques involved in creating, organizing, accessing, manipulating, and controlling files stored on storage devices.

It includes tasks such as file creation, deletion, naming, classification, and access control.

File-System management:

- Files usually organized into directories
- Access control on most systems to determine who can access what

OS activities include:

- Creating and deleting files and directories
- Primitives to manipulate files and directories
- Mapping files onto secondary storage
- Backup files onto stable (non-volatile) storage media

Mass Storage Managemant

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time

- Proper management is of central importance

- Entire speed of computer operation hinges on disk subsystem and its algorithms

OS activities:

- Mounting and unmounting
- Free-space management
- Storage allocation
- Disk scheduling
- Partitioning
- Protection

Caching

- Caching – is the process of storing data in a cache, which is a temporary storage area that facilitates faster access to data with the goal of improving application and system performance.

- Information in use copied from slower to faster storage temporarily

- Faster storage (cache) checked first to determine if information is there

➢ If it is, information used directly from the cache (fast)
➢ If not, data copied to cache and used there

➢ Cache management important design problem
➢ Cache size and replacement policy

Characteristics of Various Types of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Cache coherence

Cache coherence in an operating system (OS) is the process of ensuring that data stored in different caches within a multiprocessor is consistent and synchronized.

Cache coherence is important because it helps maintain data integrity and program correctness when multiple processors access and modify the same data at the same time.

Cache coherence protocols manage the flow of data between caches to ensure that updates to shared data are propagated to all relevant caches in a timely fashion. This way, all processors see a consistent view of memory.

Summary:

Operating Systems: It is the interface between the user and the computer hardware.
Types of Operating System (OS):

Batch OS – A set of similar jobs are stored in the main memory for execution. A job gets assigned to the CPU, only when the execution of the previous job completes.

Multiprogramming OS – The main memory consists of jobs waiting for CPU time. The OS selects one of the processes and assigns it to the CPU. Whenever the executing process needs to wait for any other operation (like I/O), the OS selects another process from the job queue and assigns it to the CPU. This way, the CPU is never kept idle and the user gets the flavor of getting multiple tasks done at once.

Multitasking OS – Multitasking OS combines the benefits of Multiprogramming OS and CPU scheduling to perform quick switches between jobs. The switch is so quick that the user can interact with each program as it runs

Time Sharing OS – Time-sharing systems require interaction with the user to instruct the OS to perform various tasks. The OS responds with an output. The instructions are usually given through an input device like the keyboard.

Real Time OS – Real-Time OS are usually built for dedicated systems to accomplish a specific set of tasks within deadlines.