Operating System Syllabus

Unit-1

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.

An operating system is software that enables applications to interact with a computer's hardware. The software that contains the core components of the operating system is called the **kernel**.

The primary purposes of an **Operating System** are to enable applications (software) to interact with a computer's hardware and to manage a system's hardware and software resources.

Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc. Today, Operating systems is found almost in every device like mobile phones, personal computers, mainframe computers, automobiles, TV, Toys etc.

Definitions

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.

Following is another definition taken from Wikipedia:

An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.

Architecture

We can draw a generic architecture diagram of an Operating System which is as follows:



Operating System Generations

Operating systems have been evolving over the years. We can categorise this evaluation based on different generations which is briefed below:

0th Generation

The term 0th generation is used to refer to the period of development of computing when Charles Babbage invented the Analytical Engine and later John Atanasoff created a computer in 1940. The hardware component technology of this period was electronic vacuum tubes. There was no Operating System available for this generation computer and computer programs were written in machine language. This computers in this generation were inefficient and dependent on the varying competencies of the individual programmer as operators.

First Generation (1951-1956)

The first generation marked the beginning of commercial computing including the introduction of Eckert and Mauchly’s UNIVAC I in early 1951, and a bit later, the IBM 701.

System operation was performed with the help of expert operators and without the benefit of an operating system for a time though programs began to be written in higher level, procedure-oriented languages, and thus the operator’s routine expanded. Later mono-programmed operating system was developed, which eliminated some of the human intervention in running job and provided programmers with a number of desirable functions. These systems still continued to operate under the control of a human operator who used to follow a number of steps to execute a program. Programming language like FORTRAN was developed by John W. Backus in 1956.

Second Generation (1956-1964)

The second generation of computer hardware was most notably characterised by transistors replacing vacuum tubes as the hardware component technology. The first operating system GMOS was developed by the IBM computer. GMOS was based on single stream batch processing system, because it collects all similar jobs in groups or batches and then submits the jobs to the operating system using a punch card to complete all jobs in a machine. Operating system is cleaned after completing one job and then continues to read and initiates the next job in punch card.

Researchers began to experiment with multiprogramming and multiprocessing in their computing services called the time-sharing system. A noteworthy example is the Compatible Time-Sharing System (CTSS), developed at MIT during the early 1960s.

Third Generation (1964-1979)

The third generation officially began in April 1964 with IBM’s announcement of its System/360 family of computers. Hardware technology began to use integrated circuits (ICs) which yielded significant advantages in both speed and economy.

Operating system development continued with the introduction and widespread adoption of multiprogramming. The idea of taking fuller advantage of the computer’s data channel I/O capabilities continued to develop.

Another progress which leads to developing of personal computers in fourth generation is a new development of minicomputers with DEC PDP-1. The third generation was an exciting time, indeed, for the development of both computer hardware and the accompanying operating system.

Fourth Generation (1979 – Present)

The fourth generation is characterised by the appearance of the personal computer and the workstation. The component technology of the third generation, was replaced by very large-scale integration (VLSI). Many Operating Systems which we are using today like Windows, Linux, MacOS etc developed in the fourth generation.

Following are some of important functions of an operating System.

* Memory Management
* Processor Management
* Device Management
* File Management
* Network Management
* Security
* Control over system performance
* Job accounting
* Error detecting aids
* 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 are 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

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

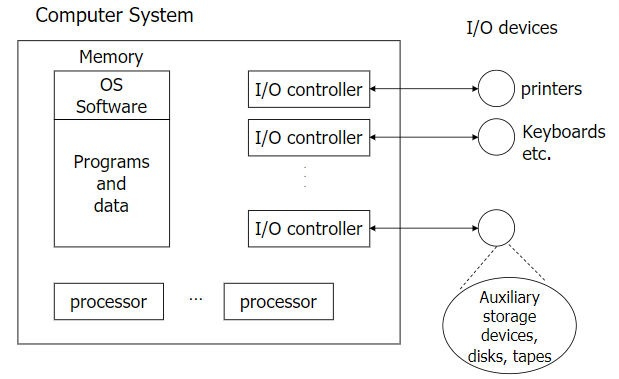
* **Security** − By means of password and similar other techniques, it prevents unauthorized access to programs and data.
* **Control over system performance** − Recording delays between request for a service and response from the system.
* **Job accounting** − Keeping track of time and resources used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

## Operating System as Resource Manager

Let us understand how the operating system works as a Resource Manager.

* Now-a-days all modern computers consist of processors, memories, timers, network interfaces, printers, and so many other devices.
* The operating system provides for an orderly and controlled allocation of the processors, memories, and I/O devices among the various programs in the bottom-up view.
* Operating system allows multiple programs to be in memory and run at the same time.
* Resource management includes multiplexing or sharing resources in two different ways: in time and in space.
* In time multiplexed, different programs take a chance of using CPU. First one tries to use the resource, then the next one that is ready in the queue and so on. For example: Sharing the printer one after another.
* In space multiplexing, instead of the customers taking a chance, each one gets part of the resource. For example − Main memory is divided into several running programs, so each one can be resident at the same time.

The diagram given below shows the functioning of OS as a resource manager −



<https://www.javatpoint.com/operating-system-structure>

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 system view respectively. An operating system is a framework that enables user application programs to interact with system hardware. The operating system does not perform any functions on its own, but it provides an atmosphere in which various programs and apps can do useful work. The operating system may be observed from the point of view of the user or the system, and it is known as the user view and the system view. In this article, you will learn the views of the operating system.

## Viewpoints of the Operating System

There are mainly two types of views of the operating system.

* **User view**
* **System view**

**User view** − The user viewpoint focuses on how the user interacts with the operating system through the usage of various application programs. Some systems are designed for a single user to monopolize the resources to maximize the user’s task.Therefore the Operating system is designed primarily for ease of use with little emphasis on quality and none of resource utilization.

### Single user view point

These systems are much more designed for a single user experience and meet the needs of single user where the performance is not given focus as the multiple user systems. Most computer users use a monitor,keyboard,printer,mouse and other accessories to operate their computer system. In some cases the system is designed to maximize the output of a single user. As a result more attention is laid on accessibility, and resource allocation is less important.

### Multiple user view point

These systems are designed for multiple user experience and meet the needs of multiple user. when there is one mainframe computer and many users on their computer trying to interact with their kernels over the mainframe to each other.

The client server architecture is a good example where many clients may interact through a remote server, and the same constraints of effective use of server resources may arise.

### Handled user view point

In the handled user viewpoint smartphones interact via wireless devices to perform numerous operations but they are not as efficient as a computer interface, limiting their usefulness. Smart phones have given you the best handheld technology ever. However their operating system is a great example of creating a device focused on the user’s point of view. The Touchscreen era has given you the best handheld technology ever.

### Embedded System user view Point

The embedded system lacks a user point of view. The remote control used to turn on or off the tv is all part of an embedded system in which the electronic device communicates with another program where the user view point is limited and allows the user to engage with the application.

**System view**− 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.The operating system may also be viewed as just a resource allocator. A computer system comprises various sources, such as hardware and software which must be managed effectively.The operating system is responsible for managing hardware resources and allocating them to programs and users to ensure maximum performance. In the system viewpoint the operating system is more involved with hardware services -CPU time,memory space, I/O operation , and so on.

From the system point of view, we are more focused on how the hardware has to interact with the operating system than the user. The hardware and the operating system interact with each other for the various purpose some of them are

### Resource Allocation

There are many resources which present in the hardware such as register, cache, RAM, ROM, processors, I/O interaction, etc. These resources demanded by the operating system when it is asked by any application program. This resource allocation has to be done only by the operating system which has used many techniques and strategies such that it brings the most out of its processing and memory space. There are various techniques such as paging, virtual memory, caching, etc.

The operating system allocates resources when a program needs them. When the program terminates, the resources are unallocated and allocated to other programs that need them.

There are two resource allocation techniques −

* **Resource partitioning approach** − It divides the resources in the system to many resource partitions, where each partition may include various resources -for example ,1MB memory, disk blocks and a printer. Then it allocates one resource partition to each user program before the program's initiation. A resource table records the resource partition and its current allocation status.

In this approach the operating system decides beforehand what resources should be allocated to which user program.

* **Pool based approach** −In the pool-based approach there is a common pool of resources. The operating system checks the allocation status in the resource table whenever a program makes a request for a resource. If the resource is free, it allocates the resources to the program.

### Control program

In the control programs it controls how input and output devices (hardware) interact with the operating system. The user may request an action that can only be done with I/O devices. The operating system must also have proper communication, control, detect and handle such devices.

## Conclusion

The user viewpoint is all about how the user has to interact with the operating system with the help of various application programs and from the system point of view we see how the hardware has to interact with the operating system for accomplishing the various tasks.

## ****Types of Operating Systems****

There are several types of Operating Systems which are mentioned below.

### ****1. Batch Operating System****

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirement and groups them into batches. It is the responsibility of the operator to sort jobs with similar needs.

*Batch Operating System*

**Advantages of Batch Operating System**

* It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in the queue.
* Multiple users can share the batch systems.
* The idle time for the 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 sometimes costly.
* The other jobs will have to wait for an unknown time if any job fails.

**Examples of Batch Operating Systems:** Payroll Systems, Bank Statements, etc.

### 2. Multi-Programming Operating System

[Multiprogramming Operating Systems](https://www.geeksforgeeks.org/difference-between-multitasking-multithreading-and-multiprocessing/) can be simply illustrated as more than one program is present in the main memory and any one of them can be kept in execution. This is basically used for better execution of resources.

*MultiProgramming*

**Advantages of Multi-Programming Operating System**

* Multi Programming increases the Throughput of the System.
* It helps in reducing the response time.

**Disadvantages of Multi-Programming Operating System**

* There is not any facility for user interaction of system resources with the system.

### 3. Multi-Processing Operating System

[Multi-Processing Operating System](https://www.geeksforgeeks.org/difference-between-multiprocessing-and-multiprogramming/)is a type of Operating System in which more than one CPU is used for the execution of resources. It betters the throughput of the System.

*Multiprocessing*

**Advantages of Multi-Processing Operating System**

* It increases the throughput of the system.
* As it has several processors, so, if one processor fails, we can proceed with another processor.

**Disadvantages of Multi-Processing Operating System**

* Due to the multiple CPU, it can be more complex and somehow difficult to understand.

### 4. Multi-Tasking Operating System

Multitasking Operating System is simply a multiprogramming Operating System with having facility of a Round-Robin Scheduling Algorithm. It can run multiple programs simultaneously.

There are two types of Multi-Tasking Systems which are listed below.

* [Preemptive Multi-Tasking](https://www.geeksforgeeks.org/difference-between-preemptive-and-cooperative-multitasking/)
* [Cooperative Multi-Tasking](https://www.geeksforgeeks.org/difference-between-preemptive-and-cooperative-multitasking/)

*Multitasking*

**Advantages of Multi-Tasking Operating System**

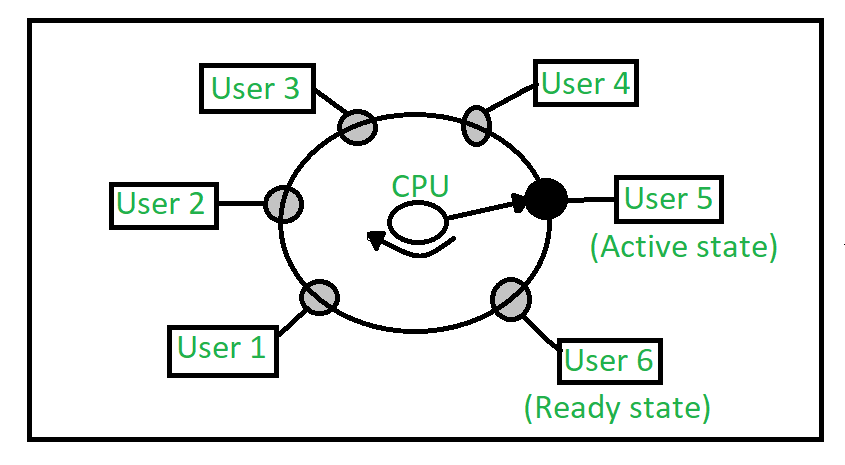
* Multiple Programs can be executed simultaneously in Multi-Tasking Operating System.
* It comes with proper memory management.

**Disadvantages of Multi-Tasking Operating System**

* The system gets heated in case of heavy programs multiple times.

### ****5. Time-Sharing Operating Systems****

Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of the CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.



*Time-Sharing OS*

**Advantages of Time-Sharing OS**

* Each task gets an equal opportunity.
* Fewer chances of duplication of software.
* CPU idle time can be reduced.
* Resource Sharing: Time-sharing systems allow multiple users to share hardware resources such as the CPU, memory, and peripherals, reducing the cost of hardware and increasing efficiency.
* Improved Productivity: Time-sharing allows users to work concurrently, thereby reducing the waiting time for their turn to use the computer. This increased productivity translates to more work getting done in less time.
* Improved User Experience: Time-sharing provides an interactive environment that allows users to communicate with the computer in real time, providing a better user experience than batch processing.

**Disadvantages of Time-Sharing OS**

* Reliability problem.
* One must have to take care of the security and integrity of user programs and data.
* Data communication problem.
* High Overhead: Time-sharing systems have a higher overhead than other operating systems due to the need for scheduling, context switching, and other overheads that come with supporting multiple users.
* Complexity: Time-sharing systems are complex and require advanced software to manage multiple users simultaneously. This complexity increases the chance of bugs and errors.
* Security Risks: With multiple users sharing resources, the risk of security breaches increases. Time-sharing systems require careful management of user access, authentication, and authorization to ensure the security of data and software.

**Examples of Time-Sharing OS with explanation**

* **IBM VM/CMS**: IBM VM/CMS is a time-sharing operating system that was first introduced in 1972. It is still in use today, providing a virtual machine environment that allows multiple users to run their own instances of operating systems and applications.
* **TSO (Time Sharing Option)**: TSO is a time-sharing operating system that was first introduced in the 1960s by IBM for the IBM System/360 mainframe computer. It allowed multiple users to access the same computer simultaneously, running their own applications.
* **Windows Terminal Services**: Windows Terminal Services is a time-sharing operating system that allows multiple users to access a Windows server remotely. Users can run their own applications and access shared resources, such as printers and network storage, in real-time.

### ****6. Distributed Operating System****

These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all over the world and, that too, at a great pace. Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred to as [loosely coupled systems or distributed systems](https://www.geeksforgeeks.org/difference-between-loosely-coupled-and-tightly-coupled-multiprocessor-system/). These systems’ processors differ in size and function. The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network.

*Distributed OS*

**Advantages of Distributed Operating System**

* Failure of one will not affect the other network communication, as all systems are independent of 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 is used 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.

**Examples of Distributed Operating Systems are** LOCUS, etc.

### ****7. Network Operating System****

These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access to files, printers, security, applications, and other networking functions over a small private network. One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections, etc. and that’s why these computers are popularly known as[tightly coupled systems](https://www.geeksforgeeks.org/difference-between-loosely-coupled-and-tightly-coupled-multiprocessor-system/).

*Network Operating System*

**Advantages of Network Operating System**

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

**Disadvantages of Network Operating System**

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

**Examples of Network Operating Systems are** Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, BSD, etc.

### ****8. Real-Time Operating System****

These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.   
**Real-time systems** are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.

**Types of Real-Time Operating Systems**

* **Hard Real-Time Systems:**   
  Hard Real-Time OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of an accident. Virtual memory is rarely found in these systems.
* **Soft Real-Time Systems:**   
  These OSs are for applications where time-constraint is less strict.

For more, refer to the [Difference Between Hard Real-Time OS and Soft Real-Time OS](https://www.geeksforgeeks.org/difference-between-hard-real-time-and-soft-real-time-system/).

*Real-Time Operating System*

**Advantages of RTOS**

* **Maximum Consumption:** Maximum utilization of devices and systems, thus more output from all the resources.
* **Task Shifting:** The time assigned for shifting tasks in these systems is very less. For example, in older systems, it takes about 10 microseconds in shifting from one task to another, and in the latest systems, it takes 3 microseconds.
* **Focus on Application:** Focus on running applications and less importance on applications that are in the queue.
* Real-time**operating system in**the **embedded system:** Since the size of programs is 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 types of systems.

**Disadvantages of RTOS**

* **Limited Tasks:** Very few tasks run at the same time and their concentration is very less on a 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 interrupts signal to respond 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.

<https://www.geeksforgeeks.org/types-of-operating-systems/>

## What is a Program?

A **program** is a set of instruction codes that has been designed to complete a certain task. It is a passive entity stored in the secondary memory of the computer system. A program is considered as a passive and static entity.

A program is like a file which contains a set of instruction codes stored on a disk in the form of an executable file. A program contains instructions written in any programming language. Programs have an unlimited span of time.

## What is a Process?

A **process** is an instance of a program that is being currently executed. It is a dynamic and active entity of a program. Processes are created when the programs are executing and they reside in the main memory.

A process exists only for a limited time, and hence gets terminated as soon as the task completes. A process always consists of instructions written in machine language. A process contains temporary data, data selection, etc.

## Difference between Program and Process

The following table highlights the important differences between a program and a process −

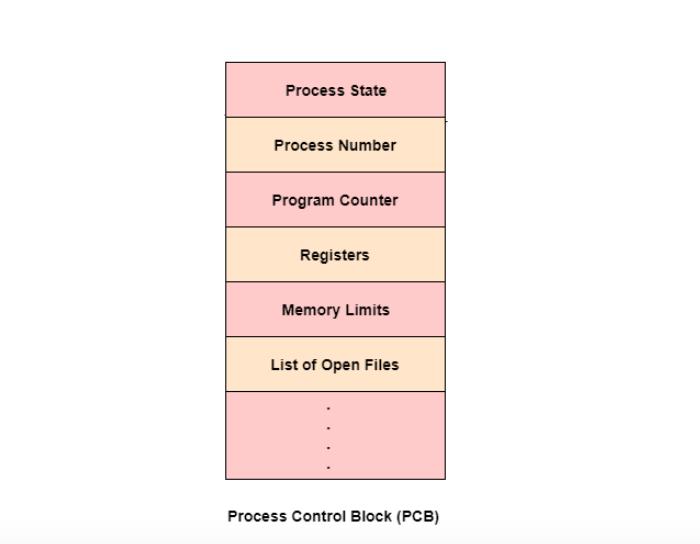
|  |  |
| --- | --- |
| **Program** | **Process** |
| It is a set of instructions that has been designed to complete a certain task. | It is an instance of a program that is being currently executed. |
| It is a passive entity. | It is an active entity. |
| It resides in the secondary memory of the system. | It is created when a program is in execution and is being loaded into the main memory. |
| It exists in a single place and continues to exist until it has been explicitly deleted. | It exists for a limited amount of time and it gets terminated once the task has been completed. |
| It is considered as a static entity. | It is considered as a dynamic entity. |
| It doesn't have a resource requirement. | It has a high resource requirement. |
| It requires memory space to store instructions. | It requires resources such as CPU, memory address, I/O during its working. |
| It doesn't have a control block. | It has its own control block, which is known as Process Control Block. |

**Process Control Block** is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.

It is very important for process management as the data structuring for processes is done in terms of the PCB. It also defines the current state of the operating system.

## Structure of the Process Control Block

The process control stores many data items that are needed for efficient process management. Some of these data items are explained with the help of the given diagram −



The following are the data items −

### Process State

This specifies the process state i.e. new, ready, running, waiting or terminated.

### Process Number

This shows the number of the particular process.

### Program Counter

This contains the address of the next instruction that needs to be executed in the process.

### Registers

This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

### List of Open Files

These are the different files that are associated with the process

### CPU Scheduling Information

The process priority, pointers to scheduling queues etc. is the CPU scheduling information that is contained in the PCB. This may also include any other scheduling parameters.

### Memory Management Information

The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc.

### I/O Status Information

This information includes the list of I/O devices used by the process, the list of files etc.

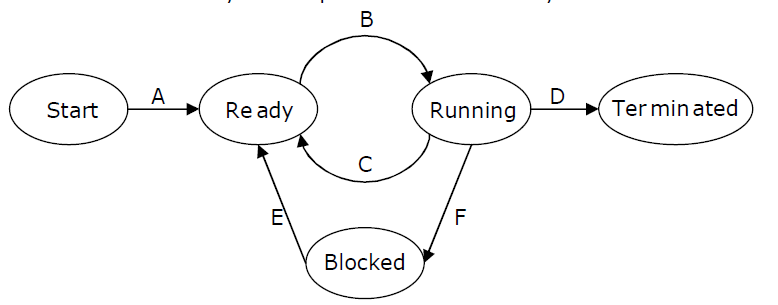
### Accounting information

The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.

### Location of the Process Control Block

The process control block is kept in a memory area that is protected from the normal user access. This is done because it contains important process information. Some of the operating systems place the PCB at the beginning of the kernel stack for the process as it is a safe location.

State Transition Diagram of a process: -

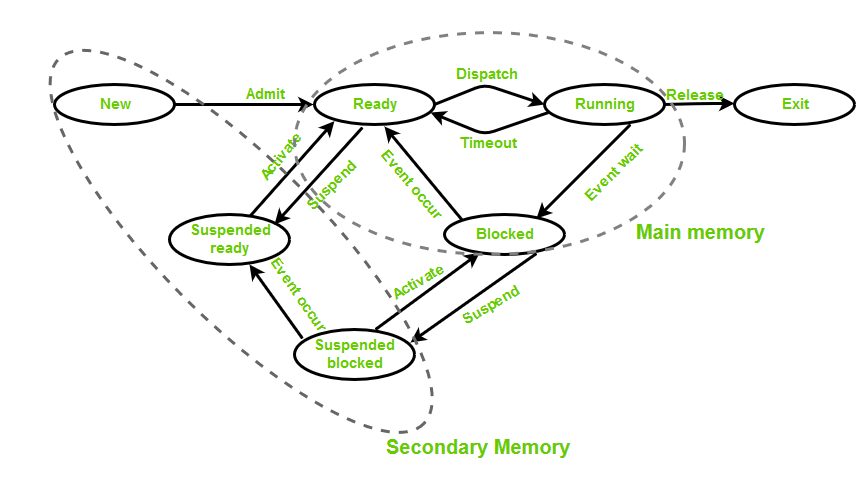


A process has several stages that it passes through from beginning to end. There must be a minimum of five states. Even though during execution, the process could be in one of these states, the names of the states are not standardized. Each [process](https://www.geeksforgeeks.org/introduction-of-process-management/)goes through several stages throughout its life cycle.

### Process States in Operating System

The states of a [process](https://www.geeksforgeeks.org/introduction-of-process-management/)are as follows:

* **New (Create):** In this step, the process is about to be created but not yet created. It is the program that is present in secondary memory that will be picked up by OS to create the process.
* **Ready:**New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue called ready queue for ready processes.
* **Run:**The process is chosen from the ready queue by the CPU for execution and the instructions within the process are executed by any one of the available CPU cores.
* **Blocked or Wait:**Whenever the process requests access to I/O or needs input from the user or needs access to a critical region(the lock for which is already acquired) it enters the blocked or waits for the state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.
* **Terminated or Completed:**Process is killed as well as [PCB](https://www.geeksforgeeks.org/process-table-and-process-control-block-pcb/)is deleted. The resources allocated to the process will be released or deallocated.
* **Suspend Ready:**Process that was initially in the ready state but was swapped out of main memory(refer to Virtual Memory topic) and placed onto external storage by the scheduler is said to be in suspend ready state. The process will transition back to a ready state whenever the process is again brought onto the main memory.
* **Suspend wait or suspend blocked:**Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory. When work is finished it may go to suspend ready.



**CPU and I/O Bound Processes:** If the process is intensive in terms of CPU operations, then it is called CPU bound process. Similarly, If the process is intensive in terms of I/O operations then it is called I/O bound process.

**How does a process move between different states in an operating system?**  
A process can move between different states in an operating system based on its execution status and resource availability. Here are some examples of how a process can move between different states:

* **New to ready:** When a process is created, it is in a new state. It moves to the ready state when the operating system has allocated resources to it and it is ready to be executed.
* **Ready to running:** When the CPU becomes available, the operating system selects a process from the ready queue depending on various scheduling algorithms and moves it to the running state.
* **Running to blocked:**When a process needs to wait for an event to occur (I/O operation or system call), it moves to the blocked state. For example, if a process needs to wait for user input, it moves to the blocked state until the user provides the input.
* **Running to ready:** When a running process is preempted by the operating system, it moves to the ready state. For example, if a higher-priority process becomes ready, the operating system may preempt the running process and move it to the ready state.
* **Blocked to ready:**When the event a blocked process was waiting for occurs, the process moves to the ready state. For example, if a process was waiting for user input and the input is provided, it moves to the ready state.
* **Running to terminated:**When a process completes its execution or is terminated by the operating system, it moves to the terminated state.

## ****Types of Schedulers****

1. **Long-term – performance:**Decides how many processes should be made to stay in the ready state. This decides the degree of multiprogramming. Once a decision is taken it lasts for a long time which also indicates that it runs infrequently. Hence it is called a long-term scheduler.
2. **Short-term – Context switching time:**Short-term scheduler will decide which process is to be executed next and then it will call the dispatcher. A dispatcher is a software that moves the process from ready to run and vice versa. In other words, it is context switching. It runs frequently. Short-term scheduler is also called CPU scheduler.
3. **Medium-term – Swapping time:**Suspension decision is taken by the medium-term scheduler. The medium-term scheduler is used for [swapping](https://www.geeksforgeeks.org/swapping-in-operating-system/)which is moving the process from main memory to secondary and vice versa. The swapping is done to reduce degree of multiprogramming.

### [****Multiprogramming****](https://www.geeksforgeeks.org/multiprogramming-in-operating-system/)

We have many processes ready to run. There are two types of multiprogramming:

1. **Preemption –** Process is forcefully removed from CPU. Pre-emotion is also called time sharing or multitasking.
2. **Non-preemption –** Processes are not removed until they complete the execution. Once control is given to the CPU for a process execution, till the CPU releases the control by itself, control cannot be taken back forcibly from the CPU.

### ****Degree of Multiprogramming****

The number of processes that can reside in the ready state at maximum decides the degree of multiprogramming, e.g., if the degree of programming = 100, this means 100 processes can reside in the ready state at maximum.

### ****Operation on the Process****

1. **Creation:**The process will be ready once it has been created, enter the ready queue (main memory), and be prepared for execution.

2. **Planning:**The operating system picks one process to begin executing from among the numerous processes that are currently in the ready queue. Scheduling is the process of choosing the next process to run.

3. **Application:**The processor begins running the process as soon as it is scheduled to run. During execution, a process may become blocked or wait, at which point the processor switches to executing the other processes.

4. **Killing or Deletion:**The OS will terminate the process once its purpose has been fulfilled. The process’s context will be over there.

5. **Blocking:** When a process is waiting for an event or resource, it is blocked. The operating system will place it in a blocked state, and it will not be able to execute until the event or resource becomes available.

6. **Resumption:**When the event or resource that caused a process to block becomes available, the process is removed from the blocked state and added back to the ready queue.

7. **Context Switching:** When the operating system switches from executing one process to another, it must save the current process’s context and load the context of the next process to execute. This is known as context switching.

8.**Inter-Process Communication:** Processes may need to communicate with each other to share data or coordinate actions. The operating system provides mechanisms for inter-process communication, such as shared memory, message passing, and synchronization primitives.

9. **Process Synchronization:** Multiple processes may need to access a shared resource or critical section of code simultaneously. The operating system provides synchronization mechanisms to ensure that only one process can access the resource or critical section at a time.

10. **Process States:**Processes may be in one of several states, including ready, running, waiting, and terminated. The operating system manages the process states and transitions between them.

What are Scheduling Queues?

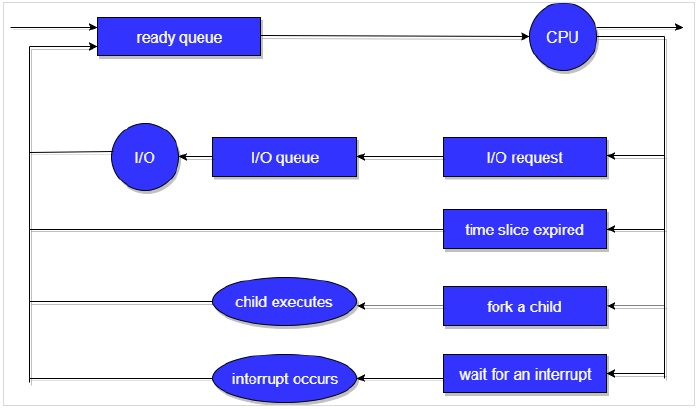
The processes that are entering into the system are stored in the **Job Queue**. Suppose if the processes are in the Ready state are generally placed in the **Ready Queue**.

The processes waiting for a device are placed in **Device Queues**. There are unique device queues which are available for every I/O device.

First place a new process in the **Ready queue** and then it waits in the ready queue till it is selected for execution.

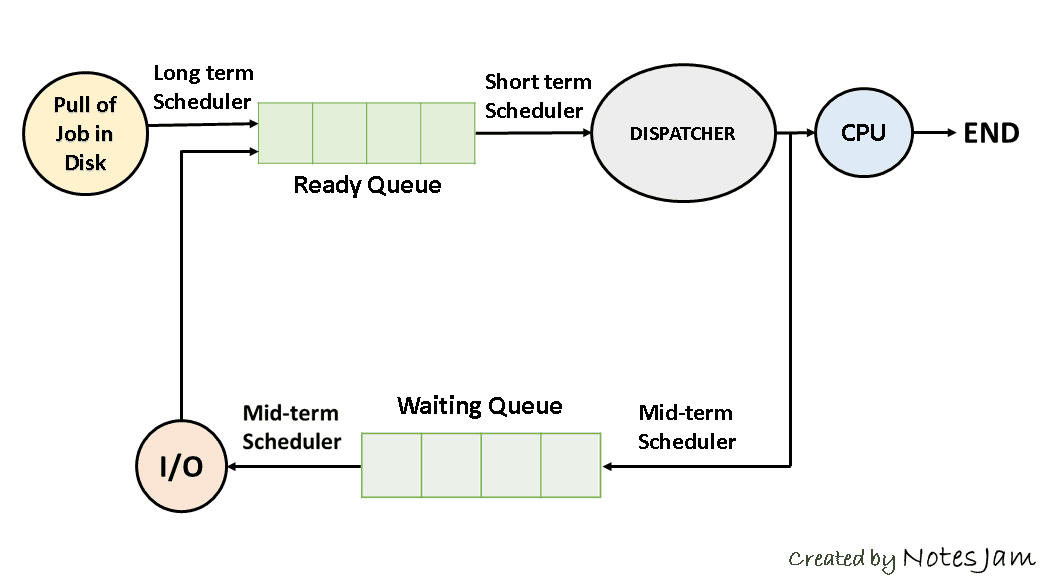
Once the process is assigned to the CPU and is executing, any one of the following events occur −

* The process issues an I/O request, and then placed in the I/O queue.
* The process may create a new sub process and wait for termination.
* The process may be removed forcibly from the CPU, which is an interrupt, and it is put back in the ready queue.



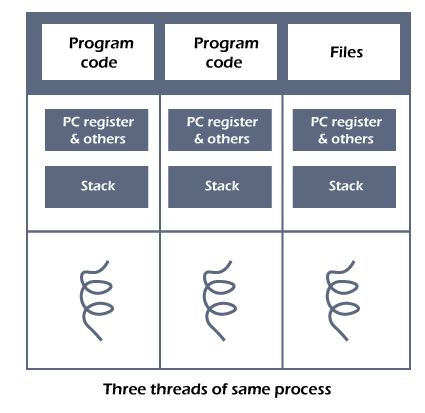
In the first two cases, the process switches from the waiting state to the ready state, and then puts it back in the ready queue. A process continues this cycle till it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.

1. **Long Term or job scheduler:**   
   It brings the new process to the ‘Ready State’. It controls the ***Degree of Multi-programming***, i.e., the number of processes present in a ready state at any point in time. It is important that the long-term scheduler make a careful selection of both I/O and CPU-bound processes. I/O-bound tasks are which use much of their time in input and output operations while CPU-bound processes are which spend their time on the CPU. The job scheduler increases efficiency by maintaining a balance between the two. They operate at a high level and are typically used in batch-processing systems.
2. **Short-term or CPU scheduler:**   
   It is responsible for selecting one process from the ready state for scheduling it on the running state. Note: Short-term scheduler only selects the process to schedule it doesn’t load the process on running.  Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring there is no starvation owing to high burst time processes.  
   ***The dispatcher*** is responsible for loading the process selected by the ort-term scheduler on the CPU (Ready to Running State) Context switching is done by the dispatcher only. A dispatcher does the following:
   1. Switching context.
   2. Switching to user mode.
   3. Jumping to the proper location in the newly loaded program.
3. **Medium-term scheduler:**   
   It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.



# **Threads in Operating System (OS)**

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.



The process can be split down into so many threads. **For example**, in a browser, many tabs can be viewed as threads. MS Word uses many threads - formatting text from one thread, processing input from another thread, etc.

## Need of Thread:

* It takes far less time to create a new thread in an existing process than to create a new process.
* Threads can share the common data, they do not need to use Inter- Process communication.
* Context switching is faster when working with threads.
* It takes less time to terminate a thread than a process.

## Types of Threads

In the [operating system](https://www.javatpoint.com/os-tutorial), there are two types of threads.

1. Kernel level thread.
2. User-level thread.

### User-level thread

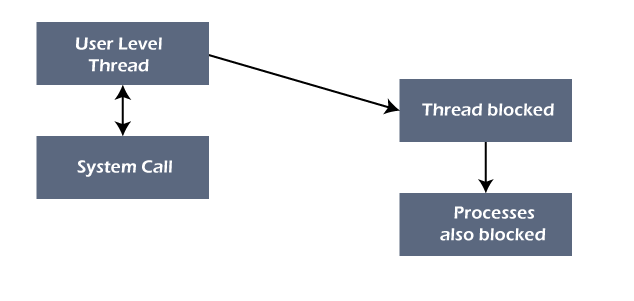
The [operating system](https://www.javatpoint.com/operating-system) does not recognize the user-level thread. User threads can be easily implemented and it is implemented by the user. If a user performs a user-level thread blocking operation, the whole process is blocked. The kernel level thread does not know nothing about the user level thread. The kernel-level thread manages user-level threads as if they are single-threaded processes. examples: [Java](https://www.javatpoint.com/java-tutorial) thread, POSIX threads, etc.

**Advantages of User-level threads**

1. The user threads can be easily implemented than the kernel thread.
2. User-level threads can be applied to such types of operating systems that do not support threads at the kernel-level.
3. It is faster and efficient.
4. Context switch time is shorter than the kernel-level threads.
5. It does not require modifications of the operating system.
6. User-level threads representation is very simple. The register, PC, stack, and mini thread control blocks are stored in the address space of the user-level process.
7. It is simple to create, switch, and synchronize threads without the intervention of the process.

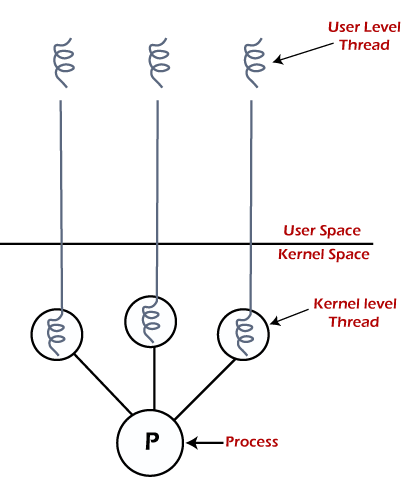
**Disadvantages of User-level threads**

1. User-level threads lack coordination between the thread and the kernel.
2. If a thread causes a page fault, the entire process is blocked.



### Kernel level thread

The kernel thread recognizes the operating system. There is a thread control block and process control block in the system for each thread and process in the kernel-level thread. The kernel-level thread is implemented by the operating system. The kernel knows about all the threads and manages them. The kernel-level thread offers a system call to create and manage the threads from user-space. The implementation of kernel threads is more difficult than the user thread. Context switch time is longer in the kernel thread. If a kernel thread performs a blocking operation, the Banky thread execution can continue. Example: Window Solaris.



**Advantages of Kernel-level threads**

1. The kernel-level thread is fully aware of all threads.
2. The scheduler may decide to spend more CPU time in the process of threads being large numerical.
3. The kernel-level thread is good for those applications that block the frequency.

**Disadvantages of Kernel-level threads**

1. The kernel thread manages and schedules all threads.
2. The implementation of kernel threads is difficult than the user thread.
3. The kernel-level thread is slower than user-level threads.

## Components of Threads

Any thread has the following components.

1. Program counter
2. Register set
3. Stack space

## Benefits of Threads

* **Enhanced throughput of the system:** When the process is split into many threads, and each thread is treated as a job, the number of jobs done in the unit time increases. That is why the throughput of the system also increases.
* **Effective Utilization of Multiprocessor system:** When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
* **Faster context switch:** The context switching period between threads is less than the process context switching. The process context switch means more overhead for the CPU.
* **Responsiveness:** When the process is split into several threads, and when a thread completes its execution, that process can be responded to as soon as possible.
* **Communication:** Multiple-thread communication is simple because the threads share the same address space, while in process, we adopt just a few exclusive communication strategies for communication between two processes.
* **Resource sharing:** Resources can be shared between all threads within a process, such as code, data, and files. Note: The stack and register cannot be shared between threads. There is a stack and register for each thread.

# **Introduction of Process Synchronization**

Introduction:

Process Synchronization is the coordination of execution of multiple processes in a multi-process system to ensure that they access shared resources in a controlled and predictable manner. It aims to resolve the problem of race conditions and other synchronization issues in a concurrent system.

The main objective of process synchronization is to ensure that multiple processes access shared resources without interfering with each other, and to prevent the possibility of inconsistent data due to concurrent access. To achieve this, various synchronization techniques such as semaphores, monitors, and critical sections are used.

In a multi-process system, synchronization is necessary to ensure data consistency and integrity, and to avoid the risk of deadlocks and other synchronization problems. Process synchronization is an important aspect of modern operating systems, and it plays a crucial role in ensuring the correct and efficient functioning of multi-process systems.

On the basis of synchronization, processes are categorized as one of the following two types:

* **Independent Process**: The execution of one process does not affect the execution of other processes.
* **Cooperative Process**: A process that can affect or be affected by other processes executing in the system.

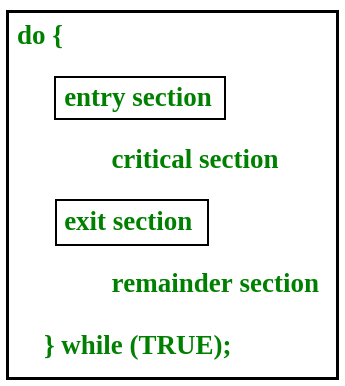
Process synchronization problem arises in the case of Cooperative process also because resources are shared in Cooperative processes.

### Race Condition:

 When more than one process is executing the same code or accessing the same memory or any shared variable in that condition there is a possibility that the output or the value of the shared variable is wrong so for that all the processes doing the race to say that my output is correct this condition known as a race condition. Several processes access and process the manipulations over the same data concurrently, then the outcome depends on the particular order in which the access takes place. A race condition is a situation that may occur inside a critical section. This happens when the result of multiple thread execution in the critical section differs according to the order in which the threads execute. Race conditions in critical sections can be avoided if the critical section is treated as an atomic instruction. Also, proper thread synchronization using locks or atomic variables can prevent race conditions.

### Critical Section Problem:

A critical section is a code segment that can be accessed by only one process at a time. The critical section contains shared variables that need to be synchronized to maintain the consistency of data variables. So, the critical section problem means designing a way for cooperative processes to access shared resources without creating data inconsistencies.

[](https://www.geeksforgeeks.org/wp-content/uploads/gq/2015/06/critical-section-problem.png)

In the entry section, the process requests for entry in the **Critical Section.**

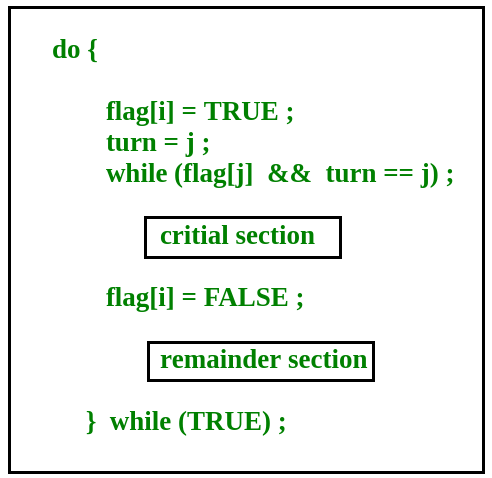
Any solution to the critical section problem must satisfy three requirements:

* **Mutual Exclusion**: If a process is executing in its critical section, then no other process is allowed to execute in the critical section.
* **Progress**: If no process is executing in the critical section and other processes are waiting outside the critical section, then only those processes that are not executing in their remainder section can participate in deciding which will enter in the critical section next, and the selection cannot be postponed indefinitely.
* **Bounded Waiting**: A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

### Peterson’s Solution:

Peterson’s Solution is a classical software-based solution to the critical section problem. In Peterson’s solution, we have two shared variables:

* Boolean flag[i]: Initialized to FALSE, initially no one is interested in entering the critical section
* int turn: The process whose turn is to enter the critical section.

[](https://www.geeksforgeeks.org/wp-content/uploads/gq/2015/06/peterson.png)

**Peterson’s Solution preserves all three conditions:**

* Mutual Exclusion is assured as only one process can access the critical section at any time.
* Progress is also assured, as a process outside the critical section does not block other processes from entering the critical section.
* Bounded Waiting is preserved as every process gets a fair chance.

**Disadvantages of Peterson’s solution:**

* It involves busy waiting. (In the Peterson’s solution, the code statement- “while(flag[j] && turn == j);” is responsible for this. Busy waiting is not favoured because it wastes CPU cycles that could be used to perform other tasks.)
* It is limited to 2 processes.
* Peterson’s solution cannot be used in modern CPU architectures.

<https://www.tutorialspoint.com/dining-philosophers-problem-dpp>

### Semaphores:

A semaphore is a signalling mechanism and a thread that is waiting on a semaphore can be signaled by another thread. This is different than a mutex as the mutex can be signaled only by the thread that is called the wait function.

A semaphore uses two atomic operations, wait and signal for process synchronization.  
A Semaphore is an integer variable, which can be accessed only through two operations wait() and signal().  
There are two types of semaphores: Binary Semaphores and Counting Semaphores.

* **Binary Semaphores:**They can only be either 0 or 1. They are also known as mutex locks, as the locks can provide mutual exclusion. All the processes can share the same mutex semaphore that is initialized to 1. Then, a process has to wait until the lock becomes 0. Then, the process can make the mutex semaphore 1 and start its critical section. When it completes its critical section, it can reset the value of the mutex semaphore to 0 and some other process can enter its critical section.
* **Counting Semaphores:**They can have any value and are not restricted over a certain domain. They can be used to control access to a resource that has a limitation on the number of simultaneous accesses. The semaphore can be initialized to the number of instances of the resource. Whenever a process wants to use that resource, it checks if the number of remaining instances is more than zero, i.e., the process has an instance available. Then, the process can enter its critical section thereby decreasing the value of the counting semaphore by 1. After the process is over with the use of the instance of the resource, it can leave the critical section thereby adding 1 to the number of available instances of the resource.

### Advantages and Disadvantages: Advantages of Process Synchronization:

* Ensures data consistency and integrity
* Avoids race conditions
* Prevents inconsistent data due to concurrent access
* Supports efficient and effective use of shared resources

### Disadvantages of Process Synchronization:

* Adds overhead to the system
* Can lead to performance degradation
* Increases the complexity of the system
* Can cause deadlocks if not implemented properly.

**CPU SCHEDULING**

**In Multiprogramming systems**, the Operating system schedules the processes on the CPU to have the maximum utilization of it and this procedure is called **CPU scheduling**. The Operating System uses various scheduling algorithm to schedule the processes.

This is a task of the short-term scheduler to schedule the CPU for the number of processes present in the Job Pool. Whenever the running process requests some IO operation then the short-term scheduler saves the current context of the process (also called PCB) and changes its state from running to waiting. During the time, process is in waiting state; the Short-term scheduler picks another process from the ready queue and assigns the CPU to this process. This procedure is called **context switching**.

## Why do we need Scheduling?

In Multiprogramming, if the long-term scheduler picks more I/O bound processes then most of the time, the CPU remains idol. The task of Operating system is to optimize the utilization of resources.

If most of the running processes change their state from running to waiting then there may always be a possibility of deadlock in the system. Hence to reduce this overhead, the OS needs to schedule the jobs to get the optimal utilization of CPU and to avoid the possibility to deadlock.

<https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms>

<https://www.tutorialspoint.com/fcfs-scheduling>

<https://www.javatpoint.com/os-sjf-scheduling>

<https://www.javatpoint.com/os-round-robin-scheduling-algorithm>

<https://www.javatpoint.com/multilevel-queue-scheduling-in-operating-system>

<https://www.geeksforgeeks.org/multilevel-queue-mlq-cpu-scheduling/>

## Unit- 2

## Memory Management

## <https://www.geeksforgeeks.org/memory-management-in-operating-system/>

**What is Memory Management?**

In a multiprogramming computer, the Operating System resides in a part of memory and the rest is used by multiple processes. The task of subdividing the memory among different processes is called Memory Management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilization of memory.

**Why Memory Management is Required?**

* Allocate and de-allocate memory before and after process execution.
* To keep track of used memory space by processes.
* To minimize fragmentation issues.
* To proper utilization of main memory.
* To maintain data integrity while executing of process.

Now we are discussing the concept of Logical Address Space and Physical Address Space

**Logical and Physical Address Space**

* **Logical Address Space:**An address generated by the CPU is known as a “Logical Address”. It is also known as a Virtual address. Logical address space can be defined as the size of the process. A logical address can be changed.
* **Physical Address Space:**An address seen by the memory unit (i.e. the one loaded into the memory address register of the memory) is commonly known as a “Physical Address”. A Physical address is also known as a Real address. The set of all physical addresses corresponding to these logical addresses is known as Physical address space. A physical address is computed by MMU. The run-time mapping from virtual to physical addresses is done by a hardware device Memory Management Unit (MMU). The physical address always remains constant.

## Static and Dynamic Loading

Loading a process into the main memory is done by a loader. There are two different types of loading:

* **Static Loading:** Static Loading is basically loading the entire program into a fixed address. It requires more memory space.
* **Dynamic Loading:** The entire program and all data of a process must be in physical memory for the process to execute. So, the size of a process is limited to the size of physical memory. To gain proper memory utilization, dynamic loading is used. In dynamic loading, a routine is not loaded until it is called. All routines are residing on disk in a relocatable load format. One of the advantages of dynamic loading is that the unused routine is never loaded. This loading is useful when a large amount of code is needed to handle it efficiently.

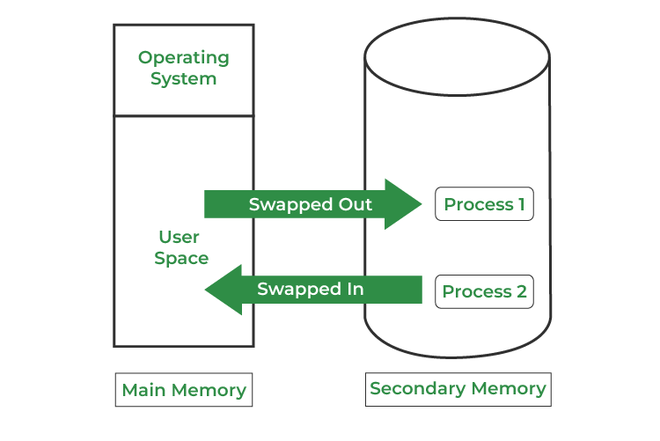
## Static and Dynamic Linking

To perform a linking task a linker is used. A linker is a program that takes one or more object files generated by a compiler and combines them into a single executable file.

* **Static Linking:**In [static linking](https://www.geeksforgeeks.org/static-and-dynamic-linking-in-operating-systems/), the linker combines all necessary program modules into a single executable program. So there is no runtime dependency. Some operating systems support only static linking, in which system language libraries are treated like any other object module.
* **Dynamic Linking:** The basic concept of dynamic linking is similar to dynamic loading. In dynamic linking, “Stub” is included for each appropriate library routine reference. A stub is a small piece of code. When the stub is executed, it checks whether the needed routine is already in memory or not. If not available then the program loads the routine into memory.

## Swapping

When a process is executed it must have resided in memory. [Swapping](https://www.geeksforgeeks.org/swapping-in-operating-system/) is a process of swapping a process temporarily into a secondary memory from the main memory, which is fast compared to secondary memory. A swapping allows more processes to be run and can be fit into memory at one time. The main part of swapping is transferred time and the total time is directly proportional to the amount of memory swapped. Swapping is also known as roll-out, or roll because if a higher priority process arrives and wants service, the memory manager can swap out the lower priority process and then load and execute the higher priority process. After finishing higher priority work, the lower priority process swapped back in memory and continued to the execution process.



*swapping in memory management*

### ****Advantages of Memory Management****

* It is a simple management approach

### ****Disadvantages of Memory Management****

* It does not support multiprogramming
* Memory is wasted

### ****Multiprogramming with Fixed Partitions (Without Swapping)****

* Memory partitions scheme with a fixed number of partitions was introduced to support multiprogramming. this scheme is based on contiguous allocation
* Each partition is a block of contiguous memory
* Memory is partitioned into a fixed number of partitions
* Each partition is of fixed size

**Example:** As shown in fig. memory is partitioned into 5 regions the region is reserved for updating the system the remaining four partitions are for the user program

| **operating system** |
| --- |
| p1 |
| p2 |
| p3 |
| p4 |

**Fixed Size Partitioning**

### ****Partition Table****

Once partitions are defined operating system keeps track of the status of  memory partitions it is done through a data structure called a partition table

| **Starting Address of Partition** | **Size of Partition** | **Status** |
| --- | --- | --- |
| 0k | 200k | allocated |
| 200k | 100k | free |
| 300k | 150k | free |
| 450k | 250k | allocated |

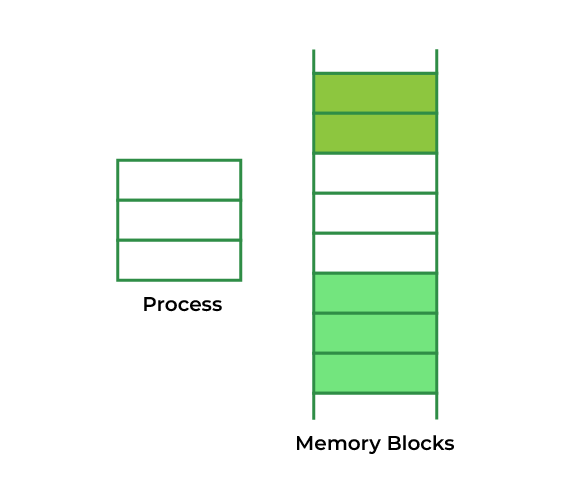
**Sample Partition Table**

### ****Logical vs Physical Address****

An address generated by the CPU is commonly referred to as a logical address. the address seen by the memory unit is known as the physical address. The logical address can be mapped to a physical address by hardware with the help of a base register this is known as dynamic relocation of memory reference.

## ****Contiguous Memory Allocation****

The main memory should oblige both the operating system and the different client processes.  Therefore, the allocation of memory becomes an important task in the operating system.  The memory is usually divided into two partitions: one for the resident operating system and one for the user processes. We normally need several user processes to reside in memory simultaneously. Therefore, we need to consider how to allocate available memory to the processes that are in the input queue waiting to be brought into memory. In adjacent memory allotment, each process is contained in a single contiguous segment of memory.



*Contiguous Memory Allocation*

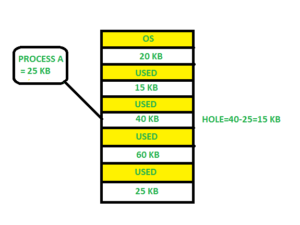
## Memory Allocation

To gain proper memory utilization, memory allocation must be allocated efficient manner. One of the simplest methods for allocating memory is to divide memory into several fixed-sized partitions and each partition contains exactly one process. Thus, the degree of multiprogramming is obtained by the number of partitions.

* **Multiple partition allocation:** In this method, a process is selected from the input queue and loaded into the free partition. When the process terminates, the partition becomes available for other processes.
* **Fixed partition allocation:** In this method, the operating system maintains a table that indicates which parts of memory are available and which are occupied by processes. Initially, all memory is available for user processes and is considered one large block of available memory. This available memory is known as a “Hole”. When the process arrives and needs memory, we search for a hole that is large enough to store this process. If the requirement is fulfilled then we allocate memory to process, otherwise keeping the rest available to satisfy future requests. While allocating a memory sometimes dynamic storage allocation problems occur, which concerns how to satisfy a request of size n from a list of free holes. There are some solutions to this problem:

### ****First Fit****

In the [First Fit](https://www.geeksforgeeks.org/first-fit-allocation-in-operating-systems/), the first available free hole fulfil the requirement of the process allocated.

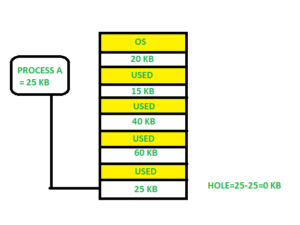


*First Fit*

Here, in this diagram, a 40 KB memory block is the first available free hole that can store process A (size of 25 KB), because the first two blocks did not have sufficient memory space.

### ****Best Fit****

In the [Best Fit](https://www.geeksforgeeks.org/best-fit-allocation-in-operating-system/), allocate the smallest hole that is big enough to process requirements. For this, we search the entire list, unless the list is ordered by size.

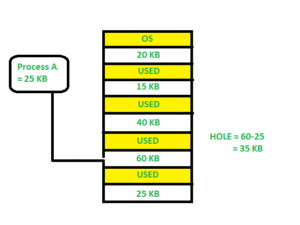


*Best Fit*

Here in this example, first, we traverse the complete list and find the last hole 25KB is the best suitable hole for Process A(size 25KB). In this method, memory utilization is maximum as compared to other memory allocation techniques.

### ****Worst Fit****

In the [Worst Fit](https://www.geeksforgeeks.org/worst-fit-allocation-in-operating-systems/), allocate the largest available hole to process. This method produces the largest leftover hole.



*Worst Fit*

Here in this example, Process A (Size 25 KB) is allocated to the largest available memory block which is 60KB. Inefficient memory utilization is a major issue in the worst fit.

## Fragmentation

[Fragmentation](https://www.geeksforgeeks.org/fragmentation-network-layer/) is defined as when the process is loaded and removed after execution from memory, it creates a small free hole. These holes cannot be assigned to new processes because holes are not combined or do not fulfill the memory requirement of the process.  To achieve a degree of multiprogramming, we must reduce the waste of memory or fragmentation problems. In the operating systems two types of fragmentation:

1. **Internal fragmentation:**[Internal fragmentation](https://www.geeksforgeeks.org/difference-between-internal-and-external-fragmentation/) occurs when memory blocks are allocated to the process more than their requested size. Due to this some unused space is left over and creating an internal fragmentation problem. **Example:** Suppose there is a fixed partitioning used for memory allocation and the different sizes of blocks 3MB, 6MB, and 7MB space in memory. Now a new process p4 of size 2MB comes and demands a block of memory. It gets a memory block of 3MB but 1MB block of memory is a waste, and it cannot be allocated to other processes too. This is called internal fragmentation.
2. **External fragmentation:**In [External Fragmentation](https://www.geeksforgeeks.org/difference-between-internal-and-external-fragmentation/), we have a free memory block, but we cannot assign it to a process because blocks are not contiguous. **Example:** Suppose (consider the above example) three processes p1, p2, and p3 come with sizes 2MB, 4MB, and 7MB respectively. Now they get memory blocks of size 3MB, 6MB, and 7MB allocated respectively. After allocating the process p1 process and the p2 process left 1MB and 2MB. Suppose a new process p4 comes and demands a 3MB block of memory, which is available, but we cannot assign it because free memory space is not contiguous.  This is called external fragmentation.

Both the first-fit and best-fit systems for memory allocation are affected by external fragmentation. To overcome the external fragmentation problem Compaction is used. In the compaction technique, all free memory space combines and makes one large block. So, this space can be used by other processes effectively.

Another possible solution to the external fragmentation is to allow the logical address space of the processes to be non-contiguous, thus permitting a process to be allocated physical memory wherever the latter is available.

## Paging

[Paging](https://www.geeksforgeeks.org/paging-in-operating-system/) is a memory management scheme that eliminates the need for a contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non-contiguous.

* **Logical Address or Virtual Address (represented in bits):** An address generated by the CPU
* **Logical Address Space or Virtual Address Space (represented in words or bytes):**The set of all logical addresses generated by a program
* **Physical Address (represented in bits):**An address actually available on a memory unit
* **Physical Address Space (represented in words or bytes):** The set of all physical addresses corresponding to the logical addresses

**Example:**

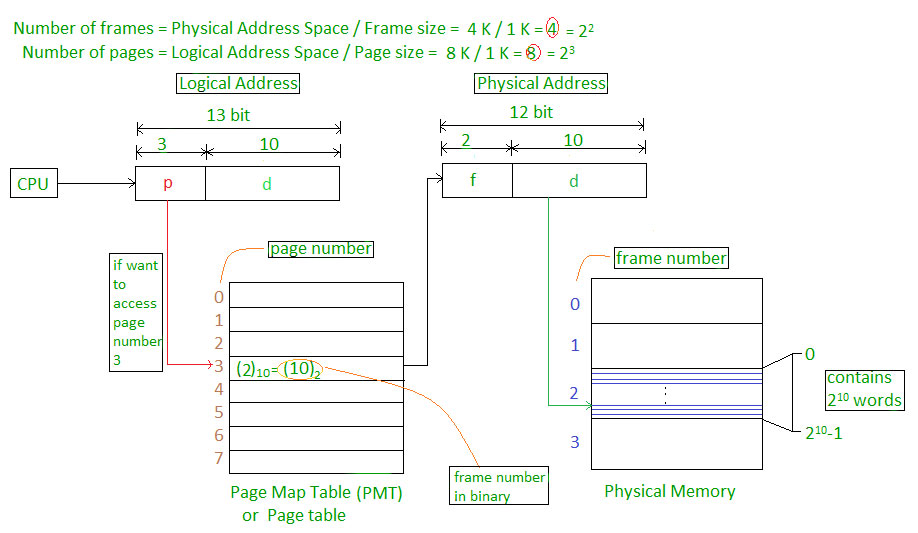
* If Logical Address = 31 bits, then Logical Address Space = 231 words = 2 G words (1 G = 230)
* If Logical Address Space = 128 M words = 27 \* 220 words, then Logical Address = log2 227 = 27 bits
* If Physical Address = 22 bits, then Physical Address Space = 222 words = 4 M words (1 M = 220)
* If Physical Address Space = 16 M words = 24 \* 220 words, then Physical Address = log2 224 = 24 bits

The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as the paging technique.

* The Physical Address Space is conceptually divided into several fixed-size blocks, called **frames**.
* The Logical Address Space is also split into fixed-size blocks, called **pages**.
* Page Size = Frame Size

Let us consider an example:

* Physical Address = 12 bits, then Physical Address Space = 4 K words
* Logical Address = 13 bits, then Logical Address Space = 8 K words
* Page size = frame size = 1 K words (assumption)



*Paging*

The address generated by the CPU is divided into:

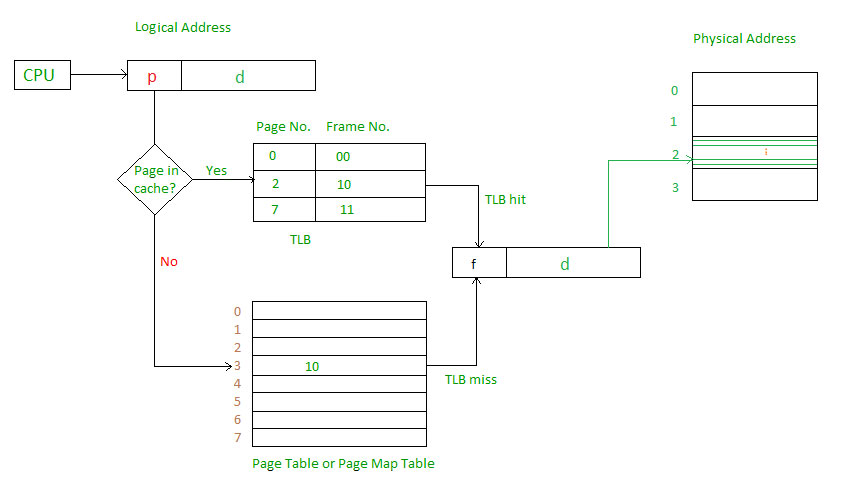
* **Page Number(p):** Number of bits required to represent the pages in Logical Address Space or Page number
* **Page Offset(d):** Number of bits required to represent a particular word in a page or page size of Logical Address Space or word number of a page or page offset.

Physical Address is divided into:

* **Frame Number(f):** Number of bits required to represent the frame of Physical Address Space or Frame number frame
* **Frame Offset(d):** Number of bits required to represent a particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

The hardware implementation of the page table can be done by using dedicated registers. But the usage of the register for the page table is satisfactory only if the page table is small. If the page table contains a large number of entries then we can use TLB(translation Look-aside buffer), a special, small, fast look-up hardware cache.

* The TLB is an associative, high-speed memory.
* Each entry in TLB consists of two parts: a tag and a value.
* When this memory is used, then an item is compared with all tags simultaneously. If the item is found, then the corresponding value is returned.



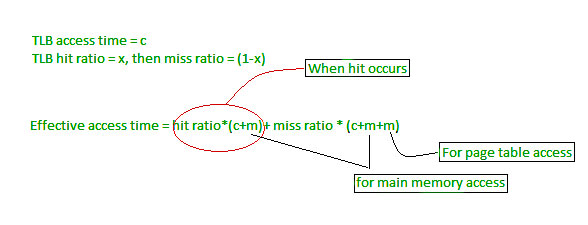
*Page Map Table*

Main memory access time = m

If page table are kept in main memory,

Effective access time = m(for page table)

+ m(for particular page in page table)



*TLB Hit and Miss*

For more details, must-read [**Paging in Operating System**](https://www.geeksforgeeks.org/paging-in-operating-system/)

## <https://www.geeksforgeeks.org/difference-between-paging-and-segmentation/>

## <https://www.geeksforgeeks.org/memory-protection-in-operating-systems/>

## <https://www.geeksforgeeks.org/virtual-memory-in-operating-system/>

## <https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/>

# Difference Between Paging and Segmentation

[**Paging**](https://www.geeksforgeeks.org/paging-in-operating-system/)**:**   
Paging is a method or technique which is used for non-contiguous memory allocation. It is a fixed-size partitioning theme (scheme). In paging, both main memory and secondary memory are divided into equal fixed-size partitions. The partitions of the secondary memory area unit and main memory area unit are known as pages and frames respectively.

Paging is a memory management method accustomed fetch processes from the secondary memory into the main memory in the form of pages. in paging, each process is split into parts wherever the size of every part is the same as the page size. The size of the last half could also be but the page size. The pages of the process area unit hold on within the frames of main memory relying upon their accessibility.

[**Segmentation**](https://www.geeksforgeeks.org/segmentation-in-operating-system/)**:**   
Segmentation is another non-contiguous memory allocation scheme like paging. like paging, in segmentation, the process isn’t divided indiscriminately into mounted(fixed) size pages. It is a variable-size partitioning theme. like paging, in segmentation, secondary and main memory are not divided into partitions of equal size. The partitions of secondary memory area units are known as segments. The details concerning every segment are hold in a table known as segmentation table. Segment table contains two main data concerning segment, one is Base, which is the bottom address of the segment and another is Limit, which is the length of the segment.

In segmentation, the CPU generates a logical address that contains the Segment number and segment offset. If the segment offset is a smaller amount than the limit then the address called valid address otherwise it throws miscalculation because the address is invalid.

The above figure shows the translation of a logical address to a physical address.

| **S.NO** | **Paging** | **Segmentation** |
| --- | --- | --- |
| 1. | In paging, the program is divided into fixed or mounted size pages. | In segmentation, the program is divided into variable size sections. |
| 2. | For the paging operating system is accountable. | For segmentation compiler is accountable. |
| 3. | Page size is determined by hardware. | Here, the section size is given by the user. |
| 4. | It is faster in comparison to segmentation. | Segmentation is slow. |
| 5. | Paging could result in internal fragmentation. | Segmentation could result in external fragmentation. |
| 6. | In paging, the logical address is split into a page number and page offset. | Here, the logical address is split into section number and section offset. |
| 7. | Paging comprises a page table that encloses the base address of every page. | While segmentation also comprises the segment table which encloses segment number and segment offset. |
| 8. | The page table is employed to keep up the page data. | Section Table maintains the section data. |
| 9. | In paging, the operating system must maintain a free frame list. | In segmentation, the operating system maintains a list of holes in the main memory. |
| 10. | Paging is invisible to the user. | Segmentation is visible to the user. |
| 11. | In paging, the processor needs the page number, and offset to calculate the absolute address. | In segmentation, the processor uses segment number, and offset to calculate the full address. |
| 12. | It is hard to allow sharing of procedures between processes. | Facilitates sharing of procedures between the processes. |
| 13 | In paging, a programmer cannot efficiently handle data structure. | It can efficiently handle data structures. |
| 14. | This protection is hard to apply. | Easy to apply for protection in segmentation. |
| 15. | The size of the page needs always be equal to the size of frames. | There is no constraint on the size of segments. |
| 16. | A page is referred to as a physical unit of information. | A segment is referred to as a logical unit of information. |
| 17. | Paging results in a less efficient system. | Segmentation results in a more efficient system. |

## Unit-3

## Device management in an operating system means controlling the Input/Output devices like disk, microphone, keyboard, printer, magnetic tape, USB ports, camcorder, scanner, other accessories, and supporting units like supporting units control channels. A process may require various resources, including main memory, file access, and access to disk drives, and others. If resources are available, they could be allocated, and control returned to the CPU. Otherwise, the procedure would have to be postponed until adequate resources become available. The system has multiple devices, and in order to handle these physical or virtual devices, the operating system requires a separate program known as an ad device controller. It also determines whether the requested device is available.

## Features of Device Management

Here, you will learn the features of device management in the operating system. Various features of the device management are as follows:

1. The OS interacts with the device controllers via the device drivers while allocating the device to the multiple processes executing on the system.
2. Device drivers can also be thought of as system software programs that bridge processes and device controllers.
3. The device management function's other key job is to implement the API.
4. Device drivers are software programs that allow an operating system to control the operation of numerous devices effectively.
5. The device controller used in device management operations mainly contains three registers: command, status, and data.

Functions of the device management in the operating system

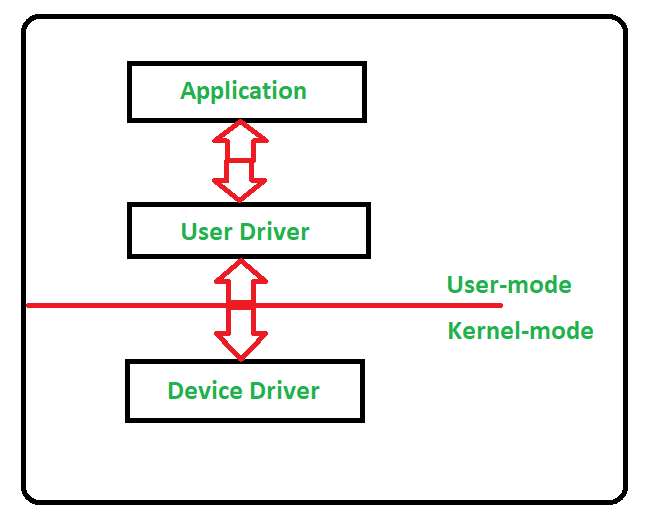
The operating system (OS) handles communication with the devices via their drivers. The OS component gives a uniform interface for accessing devices with various physical features. There are various functions of device management in the operating system. Some of them are as follows:

1. It keeps track of data, status, location, uses, etc. The file system is a term used to define a group of facilities.
2. It enforces the pre-determined policies and decides which process receives the device when and for how long.
3. It improves the performance of specific devices.
4. It monitors the status of every device, including printers, storage drivers, and other devices.
5. It allocates and effectively deallocates the device. De-allocating differentiates the devices at two levels: first, when an I/O command is issued and temporarily freed. Second, when the job is completed, and the device is permanently release

# Device Driver and It’s Purpose

**Device Driver** in computing refers to a special kind of software program or a specific type of software application that controls a specific hardware device that enables different hardware devices to communicate with the computer’s Operating System. A device driver communicates with the computer hardware by computer subsystem or computer bus connected to the hardware.

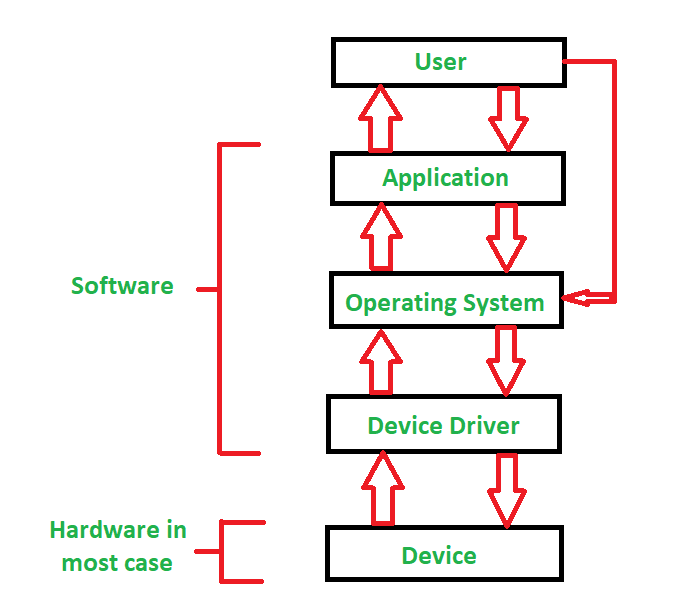
**Device Drivers** are essential for a computer system to work properly because without a device driver the particular hardware fails to work accordingly, which means it fails in doing the function/action it was created to do. Most use the term **Driver,**but some may say **Hardware Driver**, which also refers to the **Device Driver.**



**Working of Device Driver:**

Device Drivers depend upon the Operating System’s instruction to access the device and perform any particular action. After the action, they also show their reactions by delivering output or status/message from the hardware device to the Operating system. For example, a printer driver tells the printer in which format to print after getting instruction from OS, similarly, A sound card driver is there due to which 1’s and 0’s data of the MP3 file is converted to audio signals and you enjoy the music. Card reader, controller, modem, network card, sound card, printer, video card, USB devices, RAM, Speakers, etc need Device Drivers to operate.

The following figure illustrates the interaction between the user, OS, Device driver, and the devices:



**Types of Device Driver:**

For almost every device associated with the computer system there exist a Device Driver for the particular hardware. But it can be broadly classified into two types i.e.,

1. **Kernel-mode Device Driver –**   
   This Kernel-mode device driver includes some generic hardware that loads with the operating system as part of the OS these are BIOS, motherboard, processor, and some other hardware that are part of kernel software. These include the minimum system requirement device drivers for each operating system.
2. **User-mode Device Driver –**   
   Other than the devices which are brought by the kernel for working the system the user also brings some devices for use during the using of a system that devices need device drivers to function those drivers fall under User mode device driver. For example, the user needs any plug-and-play action that comes under this.

**Virtual Device Driver:**

There are also virtual device drivers (VxD), which manage the virtual device. Sometimes we use the same hardware virtually at that time virtual driver controls/manages the data flow from the different applications used by different users to the same hardware.

It is essential for a computer to have the required device drivers for all its parts to keep the system running efficiently. Many device drivers are provided by manufacturers from the beginning and also, we can later include any required device driver for our system.

**Disk Scheduling**

**Disk scheduling**is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling. Disk scheduling is important because:

* Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus, other I/O requests need to wait in the waiting queue and need to be scheduled.
* Two or more requests may be far from each other so can result in greater disk arm movement.
* Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

## 

File

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Structure

A File Structure should be according to a required format that the operating system can understand.

* A file has a certain defined structure according to its type.
* A text file is a sequence of characters organized into lines.
* A source file is a sequence of procedures and functions.
* An object file is a sequence of bytes organized into blocks that are understandable by the machine.
* When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

Ordinary files

* These are the files that contain user information.
* These may have text, databases or executable program.
* The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files

* These files contain list of file names and other information related to these files.

Special files

* These files are also known as device files.
* These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

* **Character special files** − data is handled character by character as in case of terminals or printers.
* **Block special files** − data is handled in blocks as in the case of disks and tapes.

File Access Mechanisms

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

* Sequential access
* Direct/Random access
* Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

* Random access file organization provides, accessing the records directly.
* Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
* The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

* This mechanism is built up on base of sequential access.
* An index is created for each file which contains pointers to various blocks.
* Index is searched sequentially and its pointer is used to access the file directly.

Space Allocation

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files.

* Contiguous Allocation
* Linked Allocation
* Indexed Allocation

Contiguous Allocation

* Each file occupies a contiguous address space on disk.
* Assigned disk address is in linear order.
* Easy to implement.
* External fragmentation is a major issue with this type of allocation technique.

Linked Allocation

* Each file carries a list of links to disk blocks.
* Directory contains link / pointer to first block of a file.
* No external fragmentation
* Effectively used in sequential access file.
* Inefficient in case of direct access file.

Indexed Allocation

* Provides solutions to problems of contiguous and linked allocation.
* A index block is created having all pointers to files.
* Each file has its own index block which stores the addresses of disk space occupied by the file.
* Directory contains the addresses of index blocks of files.

# **Operations on the File**

A file is a collection of logically related data that is recorded on the secondary storage in the form of sequence of operations. The content of the files is defined by its creator who is creating the file. The various operations which can be implemented on a file such as read, write, open and close etc. are called file operations. These operations are performed by the user by using the commands provided by the operating system. Some common operations are as follows:

**1.Create operation:**

This operation is used to create a file in the file system. It is the most widely used operation performed on the file system. To create a new file of a particular type the associated application program calls the file system. This file system allocates space to the file. As the file system knows the format of directory structure, so entry of this new file is made into the appropriate directory.

**2. Open operation:**

This operation is the common operation performed on the file. Once the file is created, it must be opened before performing the file processing operations. When the user wants to open a file, it provides a file name to open the particular file in the file system. It tells the operating system to invoke the open system call and passes the file name to the file system.

**3. Write operation:**

This operation is used to write the information into a file. A system call write is issued that specifies the name of the file and the length of the data has to be written to the file. Whenever the file length is increased by specified value and the file pointer is repositioned after the last byte written.

**4. Read operation:**

This operation reads the contents from a file. A Read pointer is maintained by the OS, pointing to the position up to which the data has been read.

**5. Re-position or Seek operation:**

The seek system call re-positions the file pointers from the current position to a specific place in the file i.e. forward or backward depending upon the user's requirement. This operation is generally performed with those file management systems that support direct access files.

**6. Delete operation:**

Deleting the file will not only delete all the data stored inside the file it is also used so that disk space occupied by it is freed. In order to delete the specified file the directory is searched. When the directory entry is located, all the associated file space and the directory entry is released.

**7. Truncate operation:**

Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file gets replaced.

**8. Close operation:**

When the processing of the file is complete, it should be closed so that all the changes made permanent and all the resources occupied should be released. On closing it deallocates all the internal descriptors that were created when the file was opened.

**9. Append operation:**

This operation adds data to the end of the file.

**10. Rename operation:**

This operation is used to rename the existing file.

**ATTRIBUTES**

A file can be defined as a data structure which stores the sequence of records. Files are stored in a file system, which may exist on a disk or in the main memory. Files can be simple (plain text) or complex (specially-formatted).

The collection of files is known as Directory. The collection of directories at the different levels, is known as File System.



Attributes of the File

**1.Name**

Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.

**2.Identifier**

Along with the name, Each File has its own extension which identifies the type of the file. For example, a text file has the extension **.txt,** A video file can have the extension **.mp4.**

**3.Type**

In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.

**4.Location**

In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.

**5.Size**

The Size of the File is one of its most important attributes. By size of the file, we mean the number of bytes acquired by the file in the memory.

**6.Protection**

The Admin of the computer may want the different protections for the different files. Therefore, each file carries its own set of permissions to the different group of Users.

**7.Time and Date**

Every file carries a time stamp which contains the time and date on which the file is last modified.

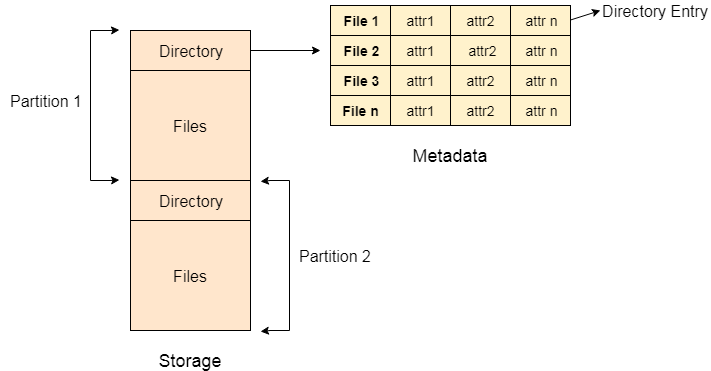
# **Directory Structure in OS (Operating System)**

## What is a directory?

Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.

To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes. The partitions are also called volumes or mini disks.

Each partition must have at least one directory in which, all the files of the partition can be listed. A directory entry is maintained for each file in the directory which stores all the information related to that file.



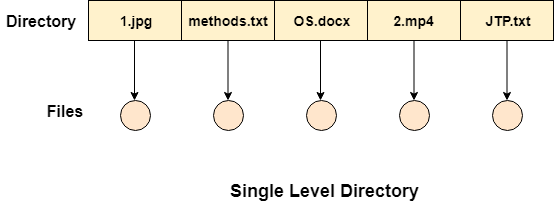
A directory can be viewed as a file which contains the Meta data of the bunch of files.

Every Directory supports a number of common operations on the file:

1. File Creation
2. Search for the file
3. File deletion
4. Renaming the file
5. Traversing Files
6. Listing of files

# **Single Level Directory**

The simplest method is to have one big list of all the files on the disk. The entire system will contain only one directory which is supposed to mention all the files present in the file system. The directory contains one entry per each file present on the file system.



This type of directories can be used for a simple system.

## Advantages

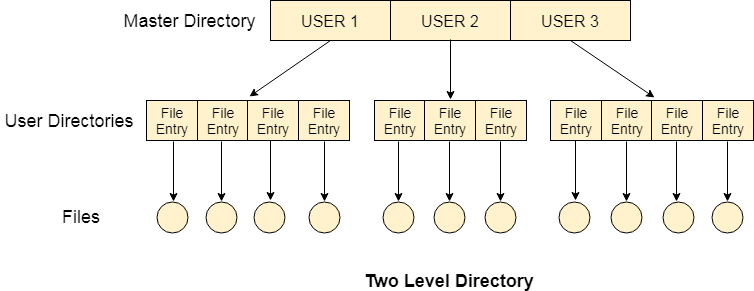
1. Implementation is very simple.
2. If the sizes of the files are very small then the searching becomes faster.
3. File creation, searching, deletion is very simple since we have only one directory.

## Disadvantages

1. We cannot have two files with the same name.
2. The directory may be very big therefore searching for a file may take so much time.
3. Protection cannot be implemented for multiple users.
4. There are no ways to group same kind of files.
5. Choosing the unique name for every file is a bit complex and limits the number of files in the system because most of the Operating System limits the number of characters used to construct the file name.

# **Two Level Directory**

In two level directory systems, we can create a separate directory for each user. There is one master directory which contains separate directories dedicated to each user. For each user, there is a different directory present at the second level, containing group of user's files. The system doesn't let a user to enter in the other user's directory without permission.



## Characteristics of two-level directory system

1. Each file has a path name as ***/User-name/directory-name/***
2. Different users can have the same file name.
3. Searching becomes more efficient as only one user's list needs to be traversed.
4. The same kind of files cannot be grouped into a single directory for a particular user.

Every Operating System maintains a variable as **PWD** which contains the present directory name (present user name) so that the searching can be done appropriately.

# Remote File System (RFS) in File Management

Files can be shared across the network via variety of methods –

* Using FTP i.e., file transfer protocol is used to transfer file from one computer to other.
* Using distributed file system (DFS) in which remote directories are visible from local machine.
* Using Remote File System (RFS) in which the arrival of networks has allowed communication between remote computer. These networks allow various hardware and software resources to be shared throughout the world.

**Remote file sharing (RFS)** is a type of **distributed file system technology**. It was developed in 1980 by AT&T. Later, it was delivered with UNIX System version V (five) release 3 (SVR3). It enables file and/or data access to multiple remote users over the Internet or a network connection. It is also known as a general process of providing remote user access to locally stored files and/or data.

It was relied on the STREAMS Transport Provider Interface feature of the operating system.

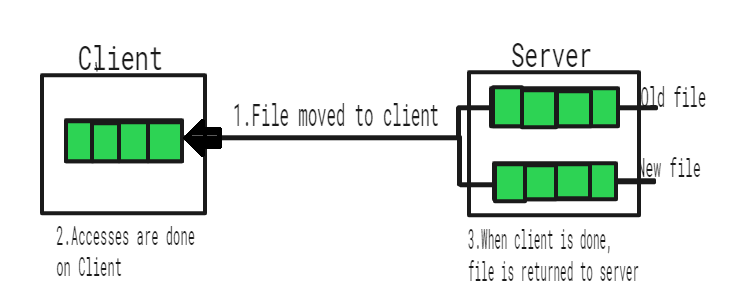
To implement remote file system, we use **client-server model.**It was one of the basic applications of Remote File System.

**Client-Server Model in RFS :**  
RFS allows a computer to support one or more file systems from one or more remote machines. In this case, the machine containing the files is server and the machine wanting access to the files is the client. The server specifies which file can be accessed by a particular client(s). Files are usually specified on a partition level.

A server can serve **multiple clients**, and a client can access **multiple servers**, depending on the implementation details of a given client-server facility. Once it is mounted, file operation requests are sent on the behalf of the user to the server, via network.

**For example**, a user sends a file open request to the server along with its ID. The server then checks file access to determine if the user has rights to access the file requested mode. This request is either allowed or denied. If it is allowed, a file is returned to the client application, and the application then may perform read, write and other operations on file.

After the required operation is performed, the client closes the files.



## What is File protection?

File protection in an operating system refers to the various mechanisms and techniques used to secure files from unauthorized access, alteration, or deletion. It involves controlling access to files, ensuring their security and confidentiality, and preventing data breaches and other security incidents.

Operating systems provide several file protection features, including file permissions, encryption, access control lists, auditing, and physical file security. These measures allow administrators to manage access to files, determine who can access them, what actions can be performed on them, and how they are stored and backed up. Proper file protection requires ongoing updates and patches to fix vulnerabilities and prevent security breaches. It is crucial for data security in the digital age where cyber threats are prevalent. By implementing file protection measures, organizations can safeguard their files, maintain data confidentiality, and minimize the risk of data breaches and other security incidents.

## Type of File protection

File protection is an essential component of modern operating systems, ensuring that files are secured from unauthorized access, alteration, or deletion. In this context, there are several types of file protection mechanisms used in operating systems to provide robust data security.

* **File Permissions** − File permissions are a basic form of file protection that controls access to files by setting permissions for users and groups. File permissions allow the system administrator to assign specific access rights to users and groups, which can include read, write, and execute privileges. These access rights can be assigned at the file or directory level, allowing users and groups to access specific files or directories as needed. File permissions can be modified by the system administrator at any time to adjust access privileges, which helps to prevent unauthorized access.
* **Encryption** − Encryption is the process of converting plain text into ciphertext to protect files from unauthorized access. Encrypted files can only be accessed by authorized users who have the correct encryption key to decrypt them. Encryption is widely used to secure sensitive data such as financial information, personal data, and other confidential information. In an operating system, encryption can be applied to individual files or entire directories, providing an extra layer of protection against unauthorized access.
* **Access Control Lists (ACLs)** − Access control lists (ACLs) are lists of permissions attached to files and directories that define which users or groups have access to them and what actions they can perform on them. ACLs can be more granular than file permissions, allowing the system administrator to specify exactly which users or groups can access specific files or directories. ACLs can also be used to grant or deny specific permissions, such as read, write, or execute privileges, to individual users or groups.
* **Auditing and Logging** − Auditing and logging are mechanisms used to track and monitor file access, changes, and deletions. It involves creating a record of all file access and changes, including who accessed the file, what actions were performed, and when they were performed. Auditing and logging can help to detect and prevent unauthorized access and can also provide an audit trail for compliance purposes.
* **Physical File Security** − Physical file security involves protecting files from physical damage or theft. It includes measures such as file storage and access control, backup and recovery, and physical security best practices. Physical file security is essential for ensuring the integrity and availability of critical data, as well as compliance with regulatory requirements.

Overall, these types of file protection mechanisms are essential for ensuring data security and minimizing the risk of data breaches and other security incidents in an operating system. The choice of file protection mechanisms will depend on the specific requirements of the organization, as well as the sensitivity and volume of the data being protected. However, a combination of these file protection mechanisms can provide comprehensive protection against various types of threats and vulnerabilities.

## Advantages of File protection

File protection is an important aspect of modern operating systems that ensures data security and integrity by preventing unauthorized access, alteration, or deletion of files. There are several advantages of file protection mechanisms in an operating system, including −

* **Data Security** − File protection mechanisms such as encryption, access control lists, and file permissions provide robust data security by preventing unauthorized access to files. These mechanisms ensure that only authorized users can access files, which helps to prevent data breaches and other security incidents. Data security is critical for organizations that handle sensitive data such as personal data, financial information, and intellectual property.
* **Compliance** − File protection mechanisms are essential for compliance with regulatory requirements such as GDPR, HIPAA, and PCI-DSS. These regulations require organizations to implement appropriate security measures to protect sensitive data from unauthorized access, alteration, or deletion. Failure to comply with these regulations can result in significant financial penalties and reputational damage.
* **Business Continuity** − File protection mechanisms are essential for ensuring business continuity by preventing data loss due to accidental or malicious deletion, corruption, or other types of damage. File protection mechanisms such as backup and recovery, auditing, and logging can help to recover data quickly in the event of a data loss incident, ensuring that business operations can resume as quickly as possible.
* **Increased Productivity** − File protection mechanisms can help to increase productivity by ensuring that files are available to authorized users when they need them. By preventing unauthorized access, alteration, or deletion of files, file protection mechanisms help to minimize the risk of downtime and data loss incidents that can impact productivity.
* **Enhanced Collaboration** − File protection mechanisms can help to enhance collaboration by allowing authorized users to access and share files securely. Access control lists, file permissions, and encryption can help to ensure that files are only accessed by authorized users, which helps to prevent conflicts and misunderstandings that can arise when multiple users access the same file.
* **Reputation** − File protection mechanisms can enhance an organizations reputation by demonstrating a commitment to data security and compliance. By implementing robust file protection mechanisms, organizations can build trust with their customers, partners, and stakeholders, which can have a positive impact on their reputation and bottom line.

Overall, these advantages of file protection mechanisms highlight the importance of data security and the need for organizations to implement appropriate measures to protect their sensitive data. File protection mechanisms can help to prevent data breaches and other security incidents, ensure compliance with regulatory requirements, and ensure business continuity in the event of a data loss incident. By implementing a comprehensive file protection strategy, organizations can enhance productivity, collaboration, and reputation, while minimizing the risk of data loss and other security incidents.

## Disadvantages of File protection

There are also some potential disadvantages of file protection in an operating system, including −

* **Overhead** − Some file protection mechanisms such as encryption, access control lists, and auditing can add overhead to system performance. This can impact system resources and slow down file access and processing times.
* **Complexity** − File protection mechanisms can be complex and require specialized knowledge to implement and manage. This can lead to errors and misconfigurations that compromise data security.
* **Compatibility Issues** − Some file protection mechanisms may not be compatible with all types of files or applications, leading to compatibility issues and limitations in file usage.
* **Cost** − Implementing robust file protection mechanisms can be expensive, especially for small organizations with limited budgets. This can make it difficult to achieve full data protection.
* **User Frustration** − Stringent file protection mechanisms such as complex passwords, frequent authentication requirements, and restricted access can frustrate users and impact productivity.

Overall, these potential disadvantages of file protection mechanisms need to be balanced against the advantages they offer in terms of data security, compliance, and business continuity. Careful planning and implementation are necessary to minimize the impact of these disadvantages and ensure effective file protection in an operating system.

## Conclusion

In conclusion, file protection mechanisms are essential for ensuring data security, compliance, and business continuity in modern operating systems. These mechanisms provide several advantages, including data security, compliance with regulatory requirements, business continuity, increased productivity, enhanced collaboration, and reputation. However, there are also some disadvantages, such as increased system overhead and complexity, and potential limitations on user flexibility. Despite these limitations, the benefits of file protection mechanisms outweigh the disadvantages, and organizations should implement appropriate file protection mechanisms to protect their sensitive data and ensure their operations are not impacted by security incidents.

**UNIT-4**

<https://www.tutorialspoint.com/operating_system/os_linux.htm>