OPERATING SYSTEMS

SEMESTER: 3rd

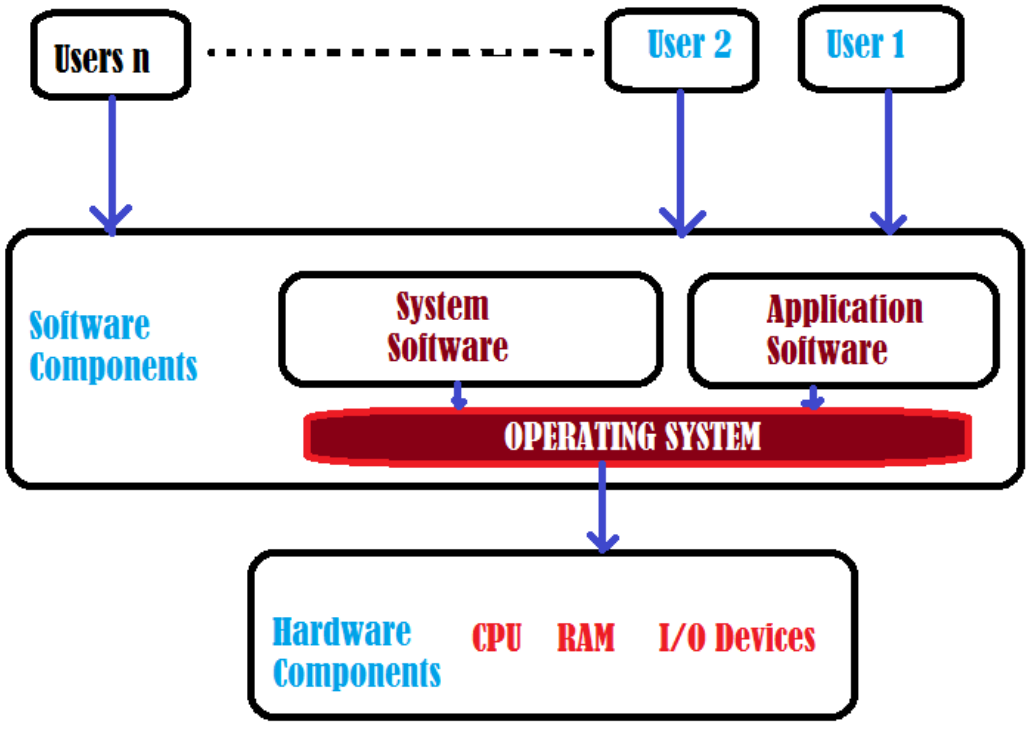
COMPUTER ENGINEERING

Unit 1

Overview of Operating Systems

Operating system (OS) is a system software that manages a computer's resources, especially the allocation of those resources among other programs. Typical resources include the central processing unit (CPU), computer memory, file storage, input/output (I/O) devices, and network connections. Management tasks include scheduling resource use to avoid conflicts and interference between programs. Unlike most programs, which complete a task and terminate, an operating system runs indefinitely and terminates only when the computer is turned off.

It works as an interface between the user and computer to perform all the tasks like memory management, file management, input & output handling, security, process management, Job accounting, error detecting, system performance controlling, peripheral devices controlling like printers & disk drives. The popular operating systems mainly include Windows, Linux, AIX, VMS, z/OS, etc.



Functions of Operating System: Following are some of important functions of an operating System.

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

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 be 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 softwares and users** – Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

User Interface

User Interface: A user interface (UI) refers to the part of an operating system, program, or device that allows a user to enter and receive information. A **Character user interface (CUI)**, displays text, and its commands are usually typed on a command line using a keyboard. With a **graphical user interface** (GUI), the functions are carried out by clicking or moving buttons, icons and menus by means of a pointing device.

Modern graphical user interfaces have evolved from text-based UIs. Some operating systems can still be used with a text-based user interface. In this case, the commands are entered as text.

In most operating systems, the primary user interface is graphical, i.e. instead of typing the commands you manipulate various graphical objects (such as icons) with a pointing device.

Types of Operating System: some of the important types of operating systems which are most commonly used are as:

Batch operating system:

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

The problems with Batch Systems are as follows –

- Lack of interaction between the user and the job.
- CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
- Difficult to provide the desired priority.

Time-sharing operating systems

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

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

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

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

Advantages of Timesharing operating systems are as follows –

- Provides the advantage of quick response.

- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages of Time-sharing operating systems are as follows −

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

Distributed operating System

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

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

The advantages of distributed systems are as follows −

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Network operating System

A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.

Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are as follows −

- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are as follows −

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

Real Time operating System

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

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

There are two types of real-time operating systems.

Hard real-time systems

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

Soft real-time systems

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

Operating System Services

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system −

- Program execution
- I/O operations
- File System manipulation
- Communication

- Error Detection
- Resource Allocation
- Protection

Program execution

Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management −

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

I/O Operation

An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

File system manipulation

A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management −

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.

- Operating System provides an interface to create the backup of file system.

Communication

In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.

The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication −

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

Error handling

Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling −

- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

Resource Management

In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management −

- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

Protection

Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection −

- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.

- The OS provides authentication features for each user by means of passwords.

System programs

System programs are utilities program that helps the user and may call further system calls. System program provides a convenient environment for program development and execution. Some are simply user interfaces two system calls and other are considerably more complex. They can be divided into these categories –

File Management

These programs **create, delete, copy, rename, print, dump, and generally manipulate files and directories.**

Status Information-

Some system programs simply ask information related to the system like date / time/ size of disc / number of users.

File Modification-

Several text editors may be available to create and modify the content of files stored on disk or other storage devices. There may also be a special commands to search contents of files or perform transformation of text.

Programming language support-

Compiler, assembler, debuggers and interpreters for common programming languages(such as C,C++, Java and more are open provided to the user with the operating system.

Program loading and execution-

Once a program is assembled or compiled, it must be loaded into memory to be executed. The system may provide absolute loaders, relocatable loaders, linkage editors, and overlay loaders. Debugging systems for either higher level languages machine languages are needed as well.

Communications-

These programs provide the mechanism for operating virtual connections Among processes, users and computer systems. They allow users to send messages to one another's screen, to browse web pages, to send electronic mail messages, to log in remotely, or to transfer files from one machine to another machine.

Some system programs supplied with operating systems are text formatters, spreadsheet, compilers, web browser, database, games and many more.

System Call :-

For performing any operation an user must have to request for a service call which is also known as **System Call** or we can say

Programming interface to the services provided by the operating system

Types of System Calls

Since an user needs to attach many resources, the type of system calls depends on the user of these resources.

Basically there are 5 broad categories of system calls-

1) Process Control System Calls

A process is a basic entity in the system. The processes in the system need to be created, deleted and aborted. Besides this many operations are required on the processes for their management.

There are some examples

- fork()- create a process
- Exit()- terminate a process
- Kill()- terminate a process abnormally
- Nice ()- increase a Priority of a process

2) File Management System Calls -

Creation, Deletion, Opening, Closing, Reading, and Writing are some general operations on files. Similarly for organizing files, there is a directory system and thereby system calls for managing them

- Create()- to create a new file
- Open ()- to open a file
- Close ()- to close a file
- read()- to read a file
- write()- to write a file
- LSeek()- change the position of the read write pointer
- Link()- give another name to a file
- Unlink()- delete a file in the directory
- Mkdir()- create a new directory.

3) Device management system calls -

The general commands are:

- request off device,
- release of device,
- read and write operation and so on.

4) Information Maintenance System Calls-

Many system calls exist simply for the purpose of transferring information between the user program and the operating system. Like -

- Return current time and date,
- Number of current user,
- The version of the operating system,
- Amount of free memory.

Get time ()	set time()	get processor()
get date()	get system data()	set process()
Set time()	set custom data()	get file()

5) Communication System Calls-

There is a need for communication among the processes in the system. **General operations are**

opening and closing the connection,

sending and receiving the messages,

reading and writing messages and so on.

msgsnd() sending a message

msgrcv() receiving a message

The system calls may be related to communication between processes either on the same machine or between processes on different nodes of a network. Thus inter process communication is provided by the operating system through these communication related system calls.

Operating System Structure

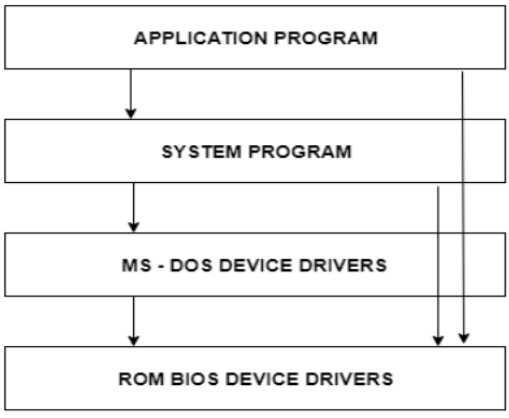
An operating system is a construct that allows the user application programs to interact with the system hardware. Since the operating system is such a complex structure, it should be created with utmost care so it can be used and modified easily. An easy way to do this is to

create the operating system in parts. Each of these parts should be well defined with clear inputs, outputs and functions.

Simple Structure

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

A diagram to illustrate the structure of MS-DOS is as follows –



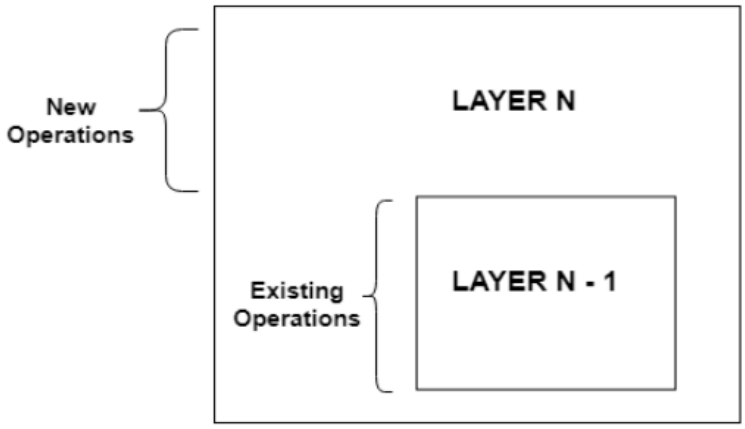
MS-DOS STRUCTURE

It is better that operating systems have a modular structure, unlike MS-DOS. That would lead to greater control over the computer system and its various applications. The modular structure would also allow the programmers to hide information as required and implement internal routines as they see fit without changing the outer specifications.

Layered Structure

One way to achieve modularity in the operating system is the layered approach. In this, the bottom layer is the hardware and the topmost layer is the user interface.

A diagram demonstrating the layered approach is as follows –



Layered Structure of Operating System

As seen from the image, each upper layer is built on the bottom layer. All the layers hide some structures, operations etc from their upper layers.

One problem with the layered structure is that each layer needs to be carefully defined. This is necessary because the upper layers can only use the functionalities of the layers below them.

Virtual Machines in Operating System

Virtual Machine abstracts the hardware of our personal computer such as CPU, disk drives, memory, NIC (Network Interface Card) etc, into many different execution environments as per our requirements, hence giving us a feel that each execution environment is a single computer. For example, VirtualBox.
When we run different processes on an operating system, it creates an illusion that each process is running on a different processor having its own virtual memory, with the help of CPU scheduling and virtual-memory techniques. There are additional features of a process that cannot be provided by the hardware alone like system calls and a file system. The virtual machine approach does not provide these additional functionalities but it only provides an interface that is same as basic hardware. Each process is provided with a virtual copy of the underlying computer system.

We can create a virtual machine for several reasons, all of which are fundamentally related to the ability to share the same basic hardware yet can also support different execution environments, i.e., different operating systems simultaneously.

The main **drawback** with the virtual-machine approach involves disk systems. Let us suppose that the physical machine has only three disk drives but wants to support seven virtual machines. Obviously, it cannot allocate a disk drive to each virtual machine, because virtual-machine software itself will need substantial disk space to provide virtual memory and spooling. The solution is to provide virtual disks.

Users are thus given their own virtual machines. After which they can run any of the operating systems or software packages that are available on the underlying machine. The

virtual-machine software is concerned with multi-programming multiple virtual machines onto a physical machine, but it does not need to consider any user-support software. This arrangement can provide a useful way to divide the problem of designing a multi-user interactive system, into two smaller pieces.

Advantages:

- There are no protection problems because each virtual machine is completely isolated from all other virtual machines.
- Virtual machine can provide an instruction set architecture that differs from real computers.
- Easy maintenance, availability and convenient recovery.

Disadvantages:

- When multiple virtual machines are simultaneously running on a host computer, one virtual machine can be affected by other running virtual machines, depending on the workload.
- Virtual machines are not as efficient as a real one when accessing the hardware.

Unit-2

Process Management

Process Concept:

A program in execution is called a process. In order to accomplish its task, process needs the computer resources.

There may exist more than one processes in the system which may require the same resource at the same time. Therefore, the operating system has to manage all the processes and the resources in a convenient and efficient way.

Some resources may need to be executed by one process at one time to maintain the consistency otherwise the system can become inconsistent and deadlock may occur.

The operating system is responsible for the following activities in connection with Process Management

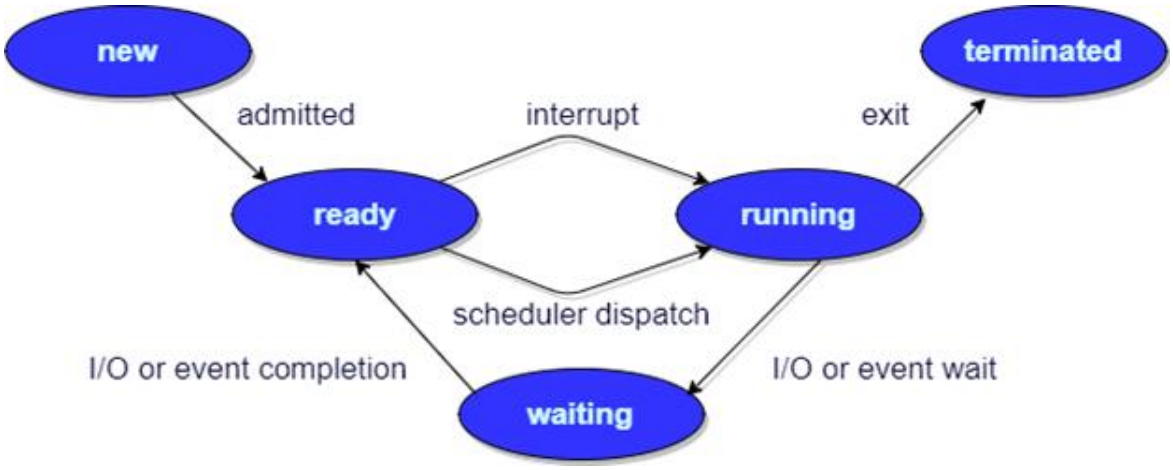
- Scheduling processes and threads on the CPUs.
- Creating and deleting both user and system processes.
- Suspending and resuming processes.
- Providing mechanisms for process synchronization.
- Providing mechanisms for process communication.

Process State:

When a process executes, it passes through different states. In general, a process can have one of the following five states at a time.

S.N.	State & Description
1	Start This is the initial state when a process is first started/created.
2	Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process.
3	Running Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.
4	Waiting Process moves into the waiting state if it needs to wait for a resource, such as waiting for some input/output.
5	Terminated or Exit

	Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

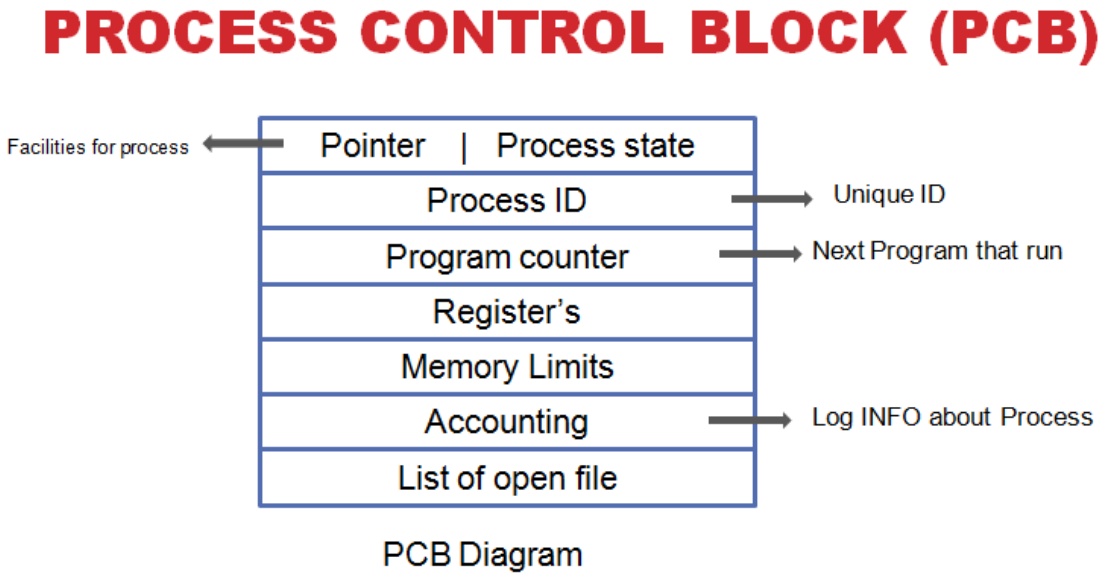


Process Control Block (PCB):

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.

Structure of the Process Control Block

The process control block 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 –



PCB Diagram

The following are some important components of PCB –

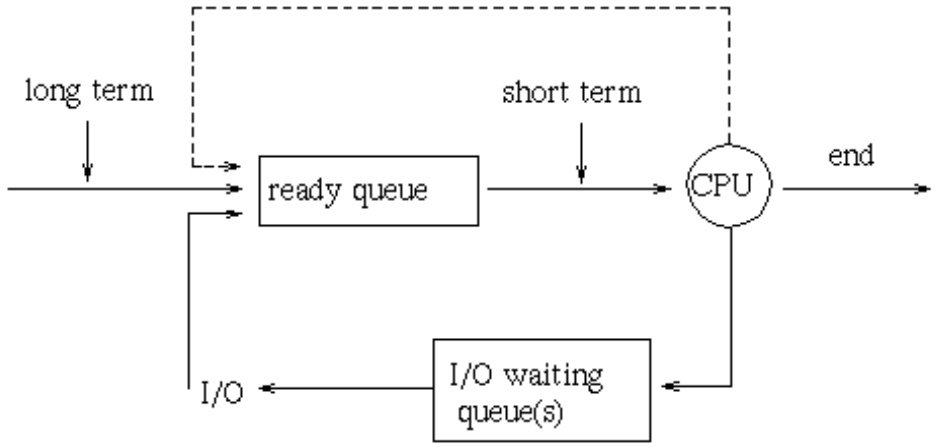
- Process state: A process can be new, ready, running, waiting, etc.
- Program counter: The program counter lets you know the address of the next instruction, which should be executed for that process.
- CPU registers: This component includes accumulators, index and general-purpose registers, and information of condition code.
- CPU scheduling information: This component includes a process priority, pointers for scheduling queues, and various other scheduling parameters.
- Accounting and business information: It includes the amount of CPU and time utilities like real time used, job or process numbers, etc.
- Memory-management information: This information includes the value of the base and limit registers, the page, or segment tables. This depends on the memory system, which is used by the operating system.
- I/O status information: This block includes a list of open files, the list of I/O devices that are allocated to the process, etc.

Scheduling Queues:

The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues –

- Job queue – This queue keeps all the processes in the system.
- Ready queue – This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- Device queues – The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Schedulers

Schedulers are special system software which handles process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler

It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of

the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Comparison among Schedulers:

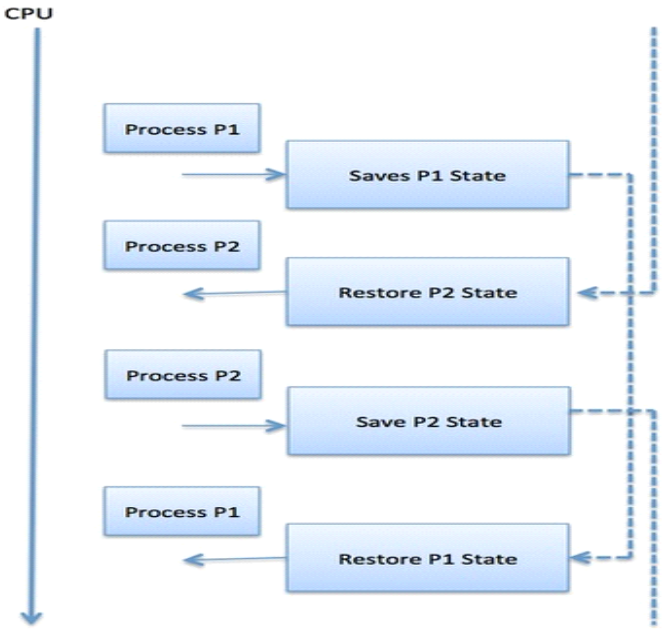
S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch

A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a

single CPU. Context switching is an essential part of a multitasking operating system features.

When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.



Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.

- Program Counter
- Scheduling information
- Base and limit register value
- Currently used register
- Changed State
- I/O State information
- Accounting information

Operation on a Process:

The execution of a process is a complex activity. It involves various operations. Following are the operations that are performed while execution of a process:

- **Creation:** This is the initial step of process execution activity. Process creation means the construction of a new process for the execution. This might be performed by

system, user or old process itself. There are several events that leads to the process creation. Some of the such events are following:

- When we start the computer, system creates several background processes.
- A user may request to create a new process.
- A process can create a new process itself while executing.
- Batch system takes initiation of a batch job.

- **Scheduling/Dispatching:** The event or activity in which the state of the process is changed from ready to running. It means the operating system puts the process from ready state into the running state. Dispatching is done by operating system when the resources are free or the process has higher priority than the ongoing process. There are various other cases in which the process in running state is preempted and process in ready state is dispatched by the operating system.
- **Blocking**: When a process invokes an input-output system call that blocks the process and operating system put in block mode. Block mode is basically a mode where process waits for input-output. Hence on the demand of process itself, operating system blocks the process and dispatches another process to the processor. Hence, in process blocking operation, the operating system puts the process in 'waiting' state.
- **Preemption**: When a timeout occurs that means the process hadn't been terminated in the allotted time interval and next process is ready to execute, then the operating system preempts the process. This operation is only valid where CPU scheduling supports preemption. Basically this happens in priority scheduling where on the incoming of high priority process the ongoing process is preempted. Hence, in process preemption operation, the operating system puts the process in 'ready' state.
- **Termination:** Process termination is the activity of ending the process. In other words, process termination is the relaxation of computer resources taken by the process for the execution. Like creation, in termination also there may be several events that may lead to the process termination. Some of them are:
    - Process completes its execution fully and it indicates to the OS that it has finished.
    - Operating system itself terminates the process due to service errors.
    - There may be problem in hardware that terminates the process.
    - One process can be terminated by another process.

Inter-process communication (IPC):

Many applications are structured as multiple processes, so these multi processes have to able to interact with each other in order to achieve a common goal. As we have already studied that operating system isolates the process from each other in order to protect each other's memory space, OS controls the amount of CPU each application gets. So some communication mechanism is required to build without sacrificing the protection. These mechanisms are called inter-process communication (IPC). Their task is to transfer data/info between address spaces without sacrificing the protection and isolation that OS provides. As communication

can vary from the continuous stream of data sharing, periodic data sharing, a single piece of data sharing, etc so the IPC mechanism has to be flexible with good performance.

- **Message-Passing IPC:** In this mechanism, the operating system establishes a communication channel (like shared buffer), and processes interact with each other by writing or sending data to the channel and reading or receiving data from the channel.
- Advantage: Advantage of this mechanism is that OS manages both writing and reading data from the channel and provides APIs. So both process uses the exact same APIs.
- Disadvantage: One disadvantage of this mechanism is that data has to first copy from sending process memory space to shared channel and then back to receiving process memory space.
- **Shared Memory IPC:** In this mechanism, the OS creates a shared memory channel and then maps it to each process memory space, and then processes are allowed to read and write to the channel as if they would do to any memory space that is part of their memory space.
- Advantage: The advantage of this process is that OS is not involved in the communication.
- Disadvantage: As OS is not involved in the communication this mechanism does not support fixed and well-defined APIs for reading and writing data, so this mechanism is error-prone and sometimes the developers have to re-implement the code.

CPU Scheduling

CPU Scheduling is a process of determining which process will own CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution. The selection process will be carried out by the CPU scheduler. It selects one of the processes in memory that are ready for execution.

Preemptive Scheduling

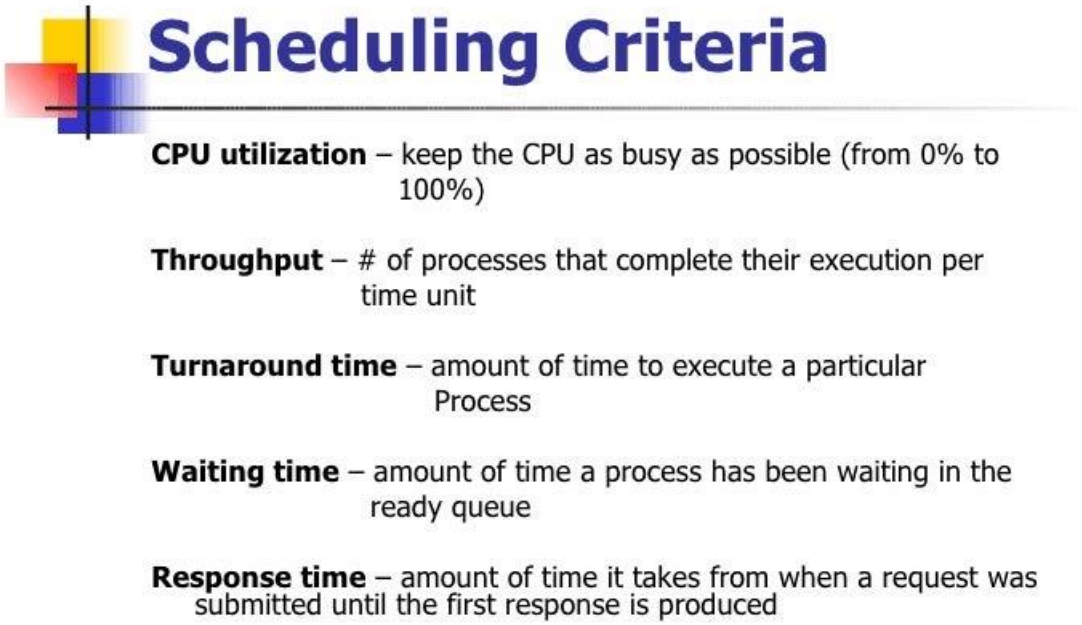
In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

Non-Preemptive Scheduling

In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That's because it doesn't need special hardware (for example, a timer) like preemptive scheduling.

CPU Scheduling Criteria

A CPU scheduling algorithm tries to maximize and minimize the following:



Maximize:

CPU utilization: CPU utilization is the main task in which the operating system needs to make sure that CPU remains as busy as possible. It can range from 0 to 100 percent. However, for the RTOS, it can be range from 40 percent for low-level and 90 percent for the high-level system.

Throughput: The number of processes that finish their execution per unit time is known Throughput. So, when the CPU is busy executing the process, at that time, work is being done, and the work completed per unit time is called Throughput.

Minimize:

Waiting time: Waiting time is an amount that specific process needs to wait in the ready queue.

Response time: It is an amount to time in which the request was submitted until the first response is produced.

Turnaround Time: Turnaround time is an amount of time to execute a specific process. It is the calculation of the total time spent waiting to get into the memory, waiting in the queue and, executing on the CPU. The period between the time of process submission to the completion time is the turnaround time.

Dispatcher

It is a module that provides control of the CPU to the process. The Dispatcher should be fast so that it can run on every context switch. **Dispatch latency** is the amount of time needed by the CPU scheduler to stop one process and start another.

Functions performed by Dispatcher:

- Context Switching
- Switching to user mode
- Moving to the correct location in the newly loaded program.

Types of CPU scheduling Algorithm

There are mainly six types of process scheduling algorithms

- First Come First Served (FCFS)
- Shortest-Job-First (SJF) Scheduling
- Shortest Remaining Time
- Priority Scheduling
- Round Robin Scheduling
- Multilevel Queue Scheduling

First Come First Served (FCFS):

First Come First Serve is the full form of FCFS. It is the easiest and most simple CPU scheduling algorithm. In this type of algorithm, the process which requests the CPU gets the CPU allocation first. This scheduling method can be managed with a FIFO queue.

As the process enters the ready queue, its PCB (Process Control Block) is linked with the tail of the queue. So, when CPU becomes free, it should be assigned to the process at the beginning of the queue.

Characteristics of FCFS method:

- It offers non-preemptive and pre-emptive scheduling algorithm.
- Jobs are always executed on a first-come, first-served basis
- It is easy to implement and use.
- However, this method is poor in performance, and the general wait time is quite high.

Shortest Remaining Time

The full form of SRT is Shortest remaining time. It is also known as SJF preemptive scheduling. In this method, the process will be allocated to the task, which is closest to its completion. This method prevents a newer ready state process from holding the completion of an older process.

Characteristics of SRT scheduling method:

- This method is mostly applied in batch environments where short jobs are required to be given preference.
- This is not an ideal method to implement it in a shared system where the required CPU time is unknown.
- Associate with each process as the length of its next CPU burst. So that operating system uses these lengths, which helps to schedule the process with the shortest possible time.

Priority Based Scheduling

Priority scheduling is a method of scheduling processes based on priority. In this method, the scheduler selects the tasks to work as per the priority.

Priority scheduling also helps OS to involve priority assignments. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority can be decided based on memory requirements, time requirements, etc.

Round-Robin Scheduling

Round robin is the oldest, simplest scheduling algorithm. The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turn. It is mostly used for scheduling algorithms in multitasking. This algorithm method helps for starvation free execution of processes.

Characteristics of Round-Robin Scheduling

- Round robin is a hybrid model which is clock-driven
- Time slice should be minimum, which is assigned for a specific task to be processed. However, it may vary for different processes.
- It is a real time system which responds to the event within a specific time limit.

Shortest Job First

SJF is a full form of (Shortest job first) is a scheduling algorithm in which the process with the shortest execution time should be selected for execution next. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution.

Characteristics of SJF Scheduling

- It is associated with each job as a unit of time to complete.
- In this method, when the CPU is available, the next process or job with the shortest completion time will be executed first.
- It is Implemented with non-preemptive policy.
- This algorithm method is useful for batch-type processing, where waiting for jobs to complete is not critical.
- It improves job output by offering shorter jobs, which should be executed first, which mostly have a shorter turnaround time.

Multiple-Level Queues Scheduling

This algorithm separates the ready queue into various separate queues. In this method, processes are assigned to a queue based on a specific property of the process, like the process priority, size of the memory, etc.

However, this is not an independent scheduling OS algorithm as it needs to use other types of algorithms in order to schedule the jobs.

Characteristic of Multiple-Level Queues Scheduling:

- Multiple queues should be maintained for processes with some characteristics.
- Every queue may have its separate scheduling algorithms.
- Priorities are given for each queue.

Process synchronization

When two or more process cooperates with each other, their order of execution must be preserved otherwise there can be conflicts in their execution and inappropriate outputs can be produced.

A cooperative process is the one which can affect the execution of other process or can be affected by the execution of other process. Such processes need to be synchronized so that their order of execution can be guaranteed.

The procedure involved in preserving the appropriate order of execution of cooperative processes is known as Process Synchronization. There are various synchronization mechanisms that are used to synchronize the processes.

Race Condition

A Race Condition typically occurs when two or more processes try to read, write and possibly make the decisions based on the memory that they are accessing concurrently.

Critical Section

The regions of a program that try to access shared resources and may cause race conditions are called critical section. To avoid race condition among the processes, we need to assure that only one process at a time can execute within the critical section.

The Critical Section Problem

Critical Section is the part of a program which tries to access shared resources. That resource may be any resource in a computer like a memory location, Data structure, CPU or any IO device.

The critical section cannot be executed by more than one process at the same time; operating system faces the difficulties in allowing and disallowing the processes from entering the critical section.

The critical section problem is used to design a set of protocols which can ensure that the Race condition among the processes will never arise.

In order to synchronize the cooperative processes, our main task is to solve the critical section problem. We need to provide a solution in such a way that the following conditions can be satisfied.

Rules for Critical Section

The critical section need to must enforce all three rules:

- **Mutual Exclusion:** Mutual Exclusion is a special type of binary semaphore which is used for controlling access to the shared resource. It includes a priority inheritance mechanism to avoid extended priority inversion problems. Not more than one process can execute in its critical section at one time.
- **Progress:** This solution is used when no one is in the critical section, and someone wants in. Then those processes not in their reminder section should decide who should go in, in a finite time.
- **Bound Waiting:** When a process makes a request for getting into critical section, there is a specific limit about number of processes can get into their critical section. So, when the limit is reached, the system must allow request to the process to get into its critical section.

Solutions to The Critical Section

In Process Synchronization, critical section plays the main role so that the problem must be solved. Here are some widely used methods to solve the critical section problem.

Peterson Solution

- Peterson's solution is widely used solution to critical section problems. This algorithm was developed by a computer scientist Peterson that's why it is named as a Peterson's solution.
- In this solution, when a process is executing in a critical state, then the other process only executes the rest of the code, and the opposite can happen. This method also helps to make sure that only a single process runs in the critical section at a specific time.

```
PROCESS Pi
FLAG[i] = true
while( (turn != i) AND (CS is !free) ){ wait;
}
CRITICAL SECTION FLAG[i] = false
turn = j; //choose another process to go to CS
```

Mutex Locks

Synchronization hardware not simple method to implement for everyone, so strict software method known as Mutex Locks was also introduced.

In this approach, in the entry section of code, a LOCK is obtained over the critical resources used inside the critical section. In the exit section that lock is released.

Semaphore Solution

Semaphore is simply a variable that is non-negative and shared between processes. It is another algorithm or solution to the critical section problem. It is a signaling mechanism and a thread that is waiting on a semaphore, which can be signaled by another thread.

It uses two atomic operations, 1) **wait,** and 2) **signal** for the process synchronization.

Example

WAIT (S):
```
while ( S <= 0 );
S = S - 1;
```

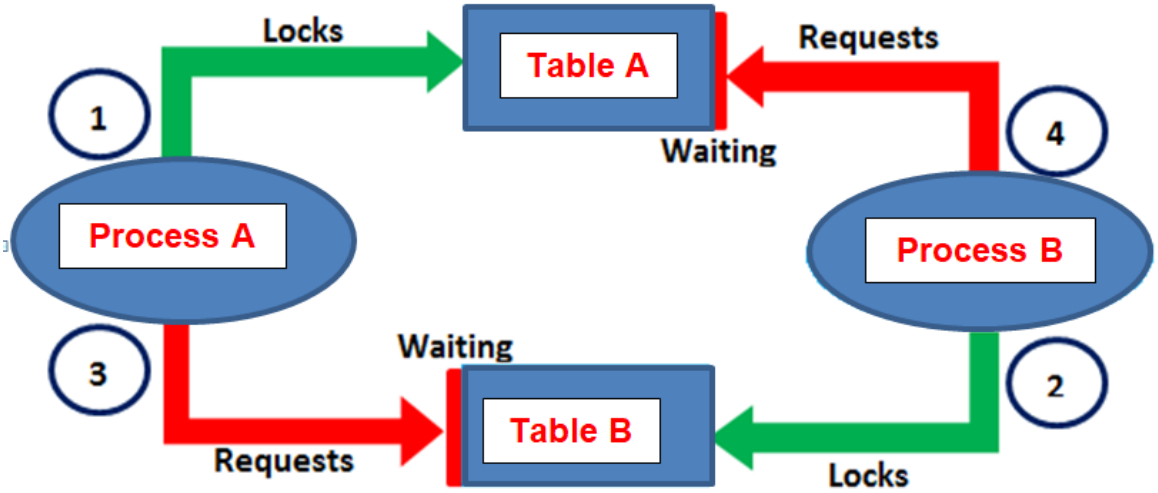
SIGNAL (S):
```
S = S + 1;
```

Unit 3

Deadlocks—(Principles and Brief Concept)

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other. A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other(s). For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.



SQL Server Deadlock

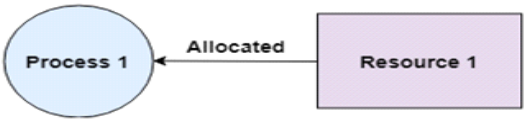
Sr.	Deadlock	Starvation
1	Deadlock is a situation where a process gets blocked and no process proceeds	Starvation is a situation where the low priority process gets blocked and the high priority processes proceed.
2	Deadlock is an infinite waiting.	Starvation is a long waiting but not infinite.
3	Every Deadlock is always a starvation.	Every starvation need not be deadlock.
4	The requested resource is blocked by the other process.	The requested resource is continuously be used by the higher priority processes.
5	Deadlock happens when Mutual exclusion, hold and wait, No preemption and circular wait occurs simultaneously.	It occurs due to the uncontrolled priority and resource management.

Conditions for Deadlock:

There are four conditions for deadlock to occur:

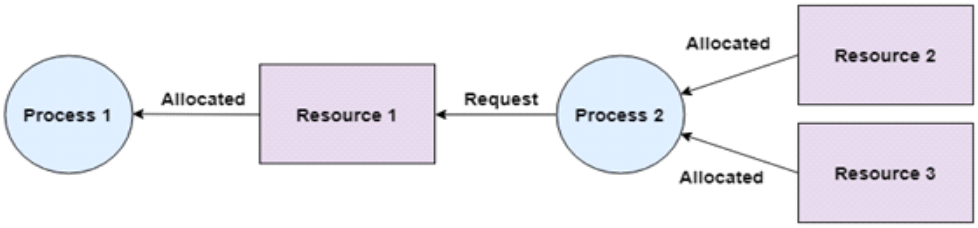
- **Mutual Exclusion**

  There should be a resource that can only be held by one process at a time. In the diagram below, there is a single instance of Resource 1 and it is held by Process 1 only.



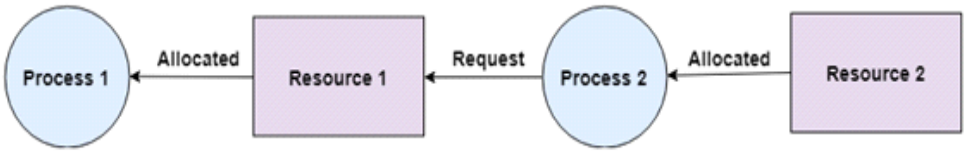
- **Hold and Wait**

  A process can hold multiple resources and still request more resources from other processes which are holding them. In the diagram given below, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.



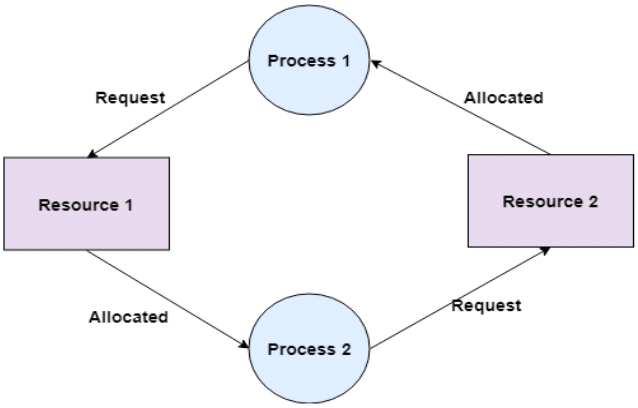
- **No Preemption**

  A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In the diagram below, Process 2 cannot preempt Resource 1 from Process 1. It will only be released when Process 1 relinquishes it voluntarily after its execution is complete.



- **Circular Wait**

  A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example: Process 1 is allocated Resource2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.



Resource-Allocation Graph

In some cases deadlocks can be understood more clearly through the use of Resource-Allocation Graphs, having the following properties:

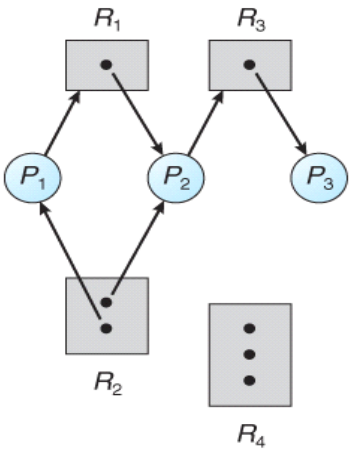
A set of resource categories, { R1, R2, R3, . . ., RN }, which appear as square nodes on the graph. Dots inside the resource nodes indicate specific instances of the resource. (E.g. two dots might represent two laser printers.)

A set of processes, { P1, P2, P3, . . ., PN }

Request Edges - A set of directed arcs from Pi to Rj, indicating that process Pi has requested Rj, and is currently waiting for that resource to become available.

Assignment Edges - A set of directed arcs from Rj to Pi indicating that resource Rj has been allocated to process Pi, and that Pi is currently holding resource Rj.

Note that a request edge can be converted into an assignment edge by reversing the direction of the arc when the request is granted. to the category box, For example:

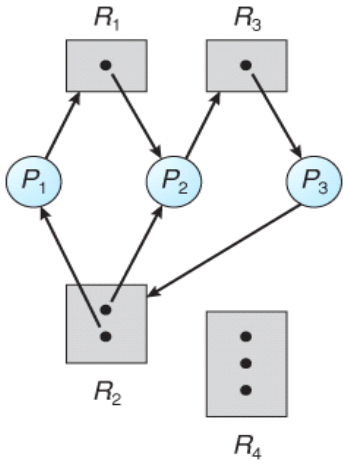


Resource allocation graph

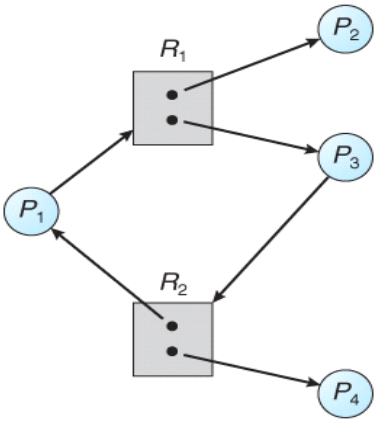
If a resource-allocation graph contains no cycles, then the system is not deadlocked.

If a resource-allocation graph does contain cycles AND each resource category contains only a single instance, then a deadlock exists.

If a resource category contains more than one instance, then the presence of a cycle in the resource-allocation graph indicates the possibility of a deadlock, but does not guarantee one



Resource allocation graph with a deadlock



Resource allocation graph with a cycle but no deadlock

Methods for handling Deadlock

1. Deadlock Ignorance

Deadlock Ignorance is the most widely used approach among all the mechanism. This is being used by many operating systems mainly for end user uses. In this approach, the Operating system assumes that deadlock never occurs. It simply ignores deadlock. This approach is best suitable for a single end user system where User uses the system only for browsing and other normal stuff.

In these types of systems, the user has to simply restart the computer in the case of deadlock. Windows and Linux are mainly using this approach.

2. Deadlock prevention

Deadlock happens only when Mutual Exclusion, hold and wait, No preemption and circular wait holds simultaneously. If it is possible to violate one of the four conditions at any time then the deadlock can never occur in the system.

The idea behind the approach is very simple that we have to fail one of the four conditions but there can be a big argument on its physical implementation in the system.

3. Deadlock avoidance

In deadlock avoidance, the operating system checks whether the system is in safe state or in unsafe state at every step which the operating system performs. The process continues until the system is in safe state. Once the system moves to unsafe state, the OS has to backtrack one step.

In simple words, The OS reviews each allocation so that the allocation doesn't cause the deadlock in the system.

4. Deadlock detection and recovery

This approach let the processes fall in deadlock and then periodically check whether deadlock occur in the system or not. If it occurs then it applies some of the recovery methods to the system to get rid of deadlock.

Deadlock Prevention

We can prevent Deadlock by eliminating any of the above four conditions.

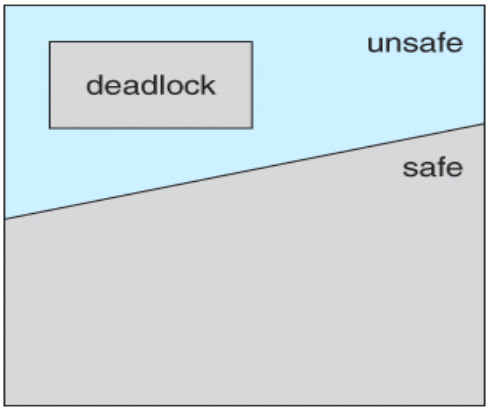
- **Eliminate Mutual Exclusion**
  It is not possible to dis-satisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable.
- **Eliminate Hold and wait**
  - Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilization. for example, if a process requires printer at a later time and we have allocated printer before the start of its execution printer will remain blocked till it has completed its execution.
  - The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.
- **Eliminate No Preemption**
  Preempt resources from the process when resources required by other high priority processes.
- **Eliminate Circular Wait**
  Each resource will be assigned with a numerical number. A process can request the resources increasing/decreasing. order of numbering.
  For Example, if P1 process is allocated R5 resources, now next time if P1 ask for R4, R3 lesser than R5 such request will not be granted, only request for resources more than R5 will be granted.

Deadlock Avoidance

In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system. The state of the system will continuously be checked for safe and unsafe states.

Safe State

- A state is ***safe*** if the system can allocate all resources requested by all processes (up to their stated maximums) without entering a deadlock state.
- More formally, a state is safe if there exists a ***safe sequence*** of processes { P0, P1, P2, ..., PN } such that all of the resource requests for Pi can be granted using the resources currently allocated to Pi and all processes Pj where j < i. (I.e. if all the processes prior to Pi finish and free up their resources, then Pi will be able to finish also, using the resources that they have freed up.)
- If a safe sequence does not exist, then the system is in an unsafe state, which ***MAY*** lead to deadlock. (All safe states are deadlock free, but not all unsafe states lead to deadlocks.)



Safe, unsafe, and deadlocked state spaces.

- For example, consider a system with 12 tape drives, allocated as follows. Is this a safe state? What is the safe sequence?

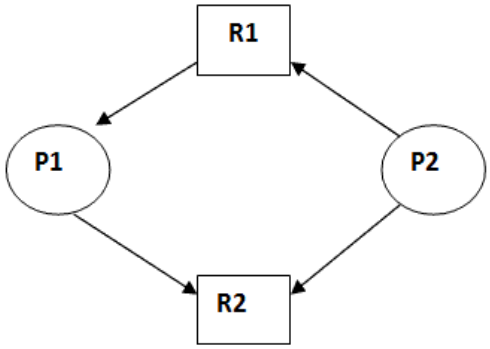
	Maximum Needs	Current Allocation
P0	10	5
P1	4	2
P2	9	2

- What happens to the above table if process P2 requests and is granted one more tape drive?
- Key to the safe state approach is that when a request is made for resources, the request is granted only if the resulting allocation state is a safe one.

Resource allocation graph

This graph is also kind of graphical bankers' algorithm where a process is denoted by a circle Pi and resources is denoted by a rectangle R_J (.dots) inside the resources represents copies.

Presence of a cycle in the resources allocation graph is necessary but not sufficient condition for detection of deadlock. If the type of every resource has exactly one copy than the presence of cycle is necessary as well as sufficient condition for detection of deadlock.



This is in unsafe state (cycle exist) if P1 request P2 and P2 request R1 then deadlock will occur.

Resource-Request Algorithm (The Bankers Algorithm)

Bankers's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it checks for the safe state, if after granting request system remains in the safe state it allows the request and if there is no safe state it doesn't allow the request made by the process.

- Now that we have a tool for determining if a particular state is safe or not, we are now ready to look at the Banker's algorithm itself.
- This algorithm determines if a new request is safe, and grants it only if it is safe to do so.
- When a request is made (that does not exceed currently available resources), pretend it has been granted, and then see if the resulting state is a safe one. If so, grant the request, and if not, deny the request, as follows:
    - Let Request[n][m] indicate the number of resources of each type currently requested by processes. If Request[i] > Need[i] for any process i, raise an error condition.
    - If Request[i] > Available for any process i, then that process must wait for resources to become available. Otherwise the process can continue to step 3.
    - Check to see if the request can be granted safely, by pretending it has been granted and then seeing if the resulting state is safe. If so, grant the request, and if not, then the process must wait until its request can be granted safely.The procedure for granting a request (or pretending to for testing purposes) is:

- Available = Available - Request
- Allocation = Allocation + Request
- Need = Need - Request

Example

Consider the following situation:

	Allocation	Max	Available	Need
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2		1 2 2
P_2	3 0 2	9 0 2		6 0 0
P_3	2 1 1	2 2 2		0 1 1
P_4	0 0 2	4 3 3		4 3 1

- And now consider what happens if process P1 requests 1 instance of A and 2 instances of C. (Request[1] = (1, 0, 2))

	Allocation	Need	Available
	A B C	A B C	A B C
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

- What about requests of (3, 3,0) by P4? or (0, 2, 0) by P0? Can these be safely granted? Why or why not?

Detection and recovery

When the system is in deadlock then one method is to inform the operates and then operator deal with deadlock manually and the second method is system will automatically recover from deadlock. There are two ways to recover from deadlock:

- Process termination:
  Deadlock can be eliminated by aborting a process. Abort all deadlock process. Abort is processed at a time until the deadlock cycle is eliminated. This can help to recover the system from file deadlock.
- Resources preemption:
  To eliminate deadlock using resources preemption, we prompt the same resources pas processes and give these resources to another process until the deadlock cycle is broken.
  Here a process is partially rollback until the last checkpoint or and hen detection algorithm is executed.

Unit 4

Memory Management Function (Principles and Brief Concept)

Memory management is concerned with managing the primary memory. Memory consists of array of bytes or words each with their own address. The instructions are fetched from the memory by the CPU based on the value program counter.

Functions of memory management:

- Keeping track of status of each memory location.
- Determining the allocation policy.
- Memory allocation technique.
- De-allocation technique.

Address Binding:

Programs are stored on the secondary storage disks as binary executable files. When the programs are to be executed they are brought in to the main memory and placed within a process. The collection of processes on the disk waiting to enter the main memory forms the input queue. One of the processes which are to be executed is fetched from the queue and placed in the main memory. During the execution it fetches instruction and data from main memory. After the process terminates it returns back the memory space. During execution the process will go through different steps and in each step the address is represented in different ways. In source program the address is symbolic. The compiler converts the symbolic address to re-locatable address. The loader will convert this re-locatable address to absolute address.Binding of instructions and data can be done at any step along the way:

Compile time:-If we know whether the process resides in memory then absolute code can be generated. If the static address changes then it is necessary to re-compile the code from the beginning.

Load time:-If the compiler doesn't know whether the process resides in memory then it generates the re-locatable code. In this the binding is delayed until the load time.

Execution time:-If the process is moved during its execution from one memory segment to another then the binding is delayed until run time. Special hardware is used for this. Most of the general purpose operating system uses this method

Logical versus physical address

The address generated by the CPU is called logical address or virtual address. The address seen by the memory unit i.e., the one loaded in to the memory register is called the physical address. Compile time and load time address binding methods generate some logical and physical address. The execution time addressing binding generate different logical and physical address. Set of logical address space generated by the programs is the logical address space. Set of physical address corresponding to these logical addresses is the physical address space. The mapping of virtual address to physical address during run time is done by the hardware device called memory management unit (MMU). The base register is also called re-location register. Value of the re-location register is added to every address generated by the user process at the time it is sent to memory. Dynamic re-location using a re-location registers.

For a process to be executed it should be loaded in to the physical memory. The size of the process is limited to the size of the physical memory. Dynamic loading is used to obtain better memory utilization. In dynamic loading the routine or procedure will not be loaded until it is called. Whenever a routine is called, the calling routine first checks whether the called routine is already loaded or not. If it is not loaded it cause the loader to load the desired program in to the memory and updates the programs address table to indicate the change and control is passed to newly called routine.

Advantage: Gives better memory utilization. Unused routine is never loaded. Do not need special operating system support. This method is useful when large amount of codes are needed to handle in frequently occurring cases.

Dynamic linking and Shared libraries:

Some operating system supports only the static linking. In dynamic linking only the main program is loaded in to the memory. If the main program requests a procedure, the procedure is loaded and the link is established at the time of references. This linking is postponed until the execution time. With dynamic linking a "stub" is used in the image of each library referenced routine. A "stub" is a piece of code which is used to indicate how to locate the appropriate memory resident library routine or how to load library if the routine is not already present. When "stub" is executed it checks whether the routine is present is memory or not. If not it loads the routine in to the memory. This feature can be used to update libraries i.e.,

library is replaced by a new version and all the programs can make use of this library. More than one version of the library can be loaded in memory at a time and each program uses its version of the library. Only the programs that are compiled with the new version are affected by the changes incorporated in it. Other programs linked before new version is installed will continue using older libraries this type of system is called “shared library”.

Swapping

Swapping is a technique of temporarily removing inactive programs from the memory of the system. A process can be swapped temporarily out of the memory to a backing store and then brought back in to the memory for continuing the execution. This process is called swapping. Eg:-In a multi-programming environment with a round robin CPU scheduling whenever the time quantum expires then the process that has just finished is swapped out and a new process swaps in to the memory for execution.

Contiguous memory allocation:

One of the simplest method for memory allocation is to divide memory in to several fixed partition. Each partition contains exactly one process. The degree of multi-programming depends on the number of partitions. In multiple partition method, when a partition is free, process is selected from the input queue and is loaded in to free partition of memory. When process terminates, the memory partition becomes available for another process. Batch OS uses the fixed size partition scheme.The OS keeps a table indicating which part of the memory is free and is occupied. When the process enters the system it will be loaded in to the input queue. The OS keeps track of the memory requirement of each process and the amount of memory available and determines which process to allocate the memory. When a process requests, the OS searches for large hole for this process, hole is a large block of free memory available. If the hole is too large it is split in to two. One part is allocated to the requesting process and other is returned to the set of holes. The set of holes are searched to determinewhich hole is best to allocate. There are three strategies to select a free hole:

- First bit:-Allocates first hole that is big enough. This algorithm scans memory from the beginning and selects the first available block that is large enough to hold the process.
- Best bit:-It chooses the hole i.e., closest in size to the request. It allocates the smallest hole i.e., big enough to hold the process.
- Worst fit:-It allocates the largest hole to the process request. It searches for the largest hole in the entire list.

Fragmentation:

Memory fragmentation can be of two types: Internal Fragmentation External Fragmentation

In Internal Fragmentation there is wasted space internal to a portion due to the fact that block of data loaded is smaller than the partition. Eg:-If there is a block of 50kb and if the

process requests 40kb and if the block is allocated to the process then there will be 10kb of memory left.

External Fragmentation exists when there is enough memory space exists to satisfy the request, but it not contiguous i.e., storage is fragmented in to large number of small holes. External Fragmentation may be either minor or a major problem.

One solution for over-coming external fragmentation is compaction. The goal is to move all the free memory together to form a large block. Compaction is not possible always. If the relocation is static and is done at load time then compaction is not possible. Compaction is possible if the re-location is dynamic and done at execution time.

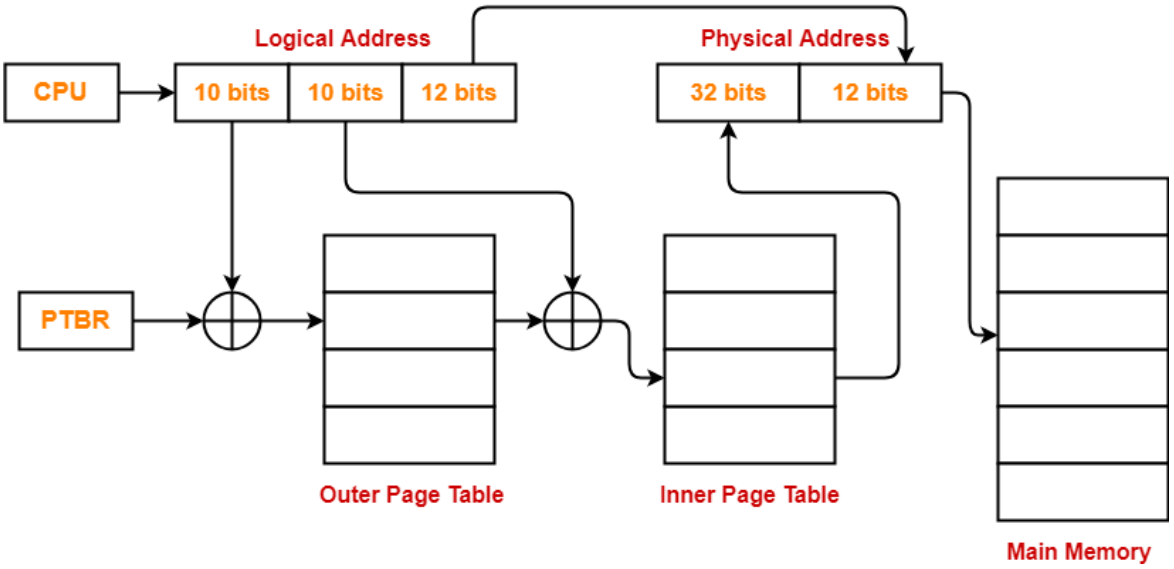
Another possible solution to the external fragmentation problem is to permit the logical address space of a process to be non-contiguous, thus allowing the process to be allocated physical memory whenever the latter is available.

Paging

As mentioned above, the memory management function called *paging* specifies storage locations to the CPU as additional memory, called virtual memory. The CPU cannot directly access storage disk, so the MMU emulates memory by mapping pages to frames that are in RAM.

Before we launch into a more detailed explanation of pages and frames, let's define some technical terms.

- **Page:** A fixed-length contiguous block of virtual memory residing on disk.
- **Frame:** A fixed-length contiguous block located in RAM; whose sizing is identical to pages.
- **Physical memory:** The computer's random access memory (RAM), typically contained in DIMM cards attached to the computer's motherboard.
- **Virtual memory:** Virtual memory is a portion of an HDD or SSD that is reserved to emulate RAM. The MMU serves up virtual memory from disk to the CPU to reduce the workload on physical memory.
- **Virtual address:** The CPU generates a virtual address for each active process. The MMU maps the virtual address to a physical location in RAM and passes the address to the bus. A virtual address space is the range of virtual addresses under CPU control.
- **Physical address:** The physical address is a location in RAM. The physical address space is the set of all physical addresses corresponding to the CPU's virtual addresses. A physical address space is the range of physical addresses under MMU control.



The Paging Process

A page table stores the definition of each page. When an active process requests data, the MMU retrieves corresponding pages into frames located in physical memory for faster processing. The process is called paging.

The MMU uses page tables to translate virtual addresses to physical ones. Each table entry indicates where a page is located: in RAM or on disk as virtual memory. Tables may have a single or multi-level page table such as different tables for applications and segments.

However, constant table lookups can slow down the MMU. A memory cache called the Translation Lookaside Buffer (TLB) stores recent translations of virtual to physical addresses for rapid retrieval. Many systems have multiple TLBs, which may reside at different locations, including between the CPU and RAM, or between multiple page table levels.

Different frame sizes are available for data sets with larger or smaller pages and matching-sized frames. 4KB to 2MB are common sizes, and GB-sized frames are available in high-performance servers.

An issue called hidden fragmentation used to be a problem in older Windows deployments (95, 98, and Me). The problem was internal (or hidden) fragmentation. Unlike the serious external fragmentation of segmenting, internal fragmentation occurred if every frame is not the exact size of the page size. However, this is not an issue in modern Windows OS.

Segmentation

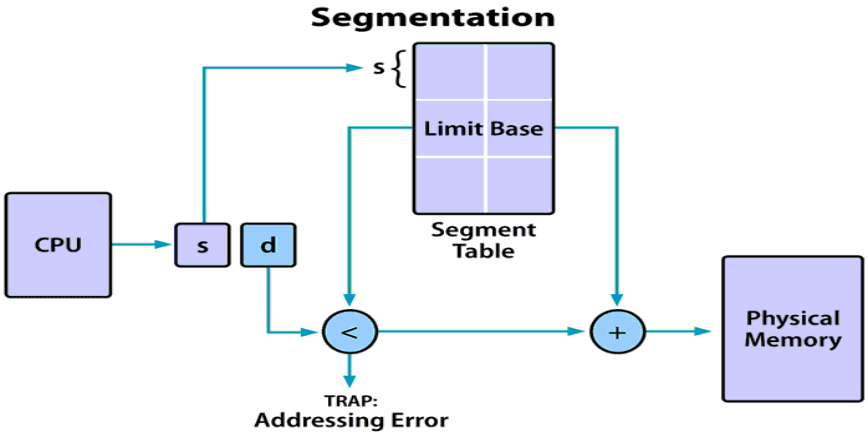
The process known as segmentation is a virtual process that creates address spaces of various sizes in a computer system, called segments. Each segment is a different virtual address space that directly corresponds to process objects.

When a process executes, segmentation assigns related data into segments for faster processing. The segmentation function maintains a segment table that includes physical addresses of the segment, size, and other data.

The Segmentation Process

Each segment stores the processes primary function, data structures, and utilities. The CPU keeps a segment map table for every process and memory blocks, along with segment identification and memory locations.

The CPU generates virtual addresses for running processes. Segmentation translates the CPU-generated virtual addresses into physical addresses that refer to a unique physical memory location. The translation is not strictly one-to-one: different virtual addresses can map to the same physical address.



Segmented Paging

Some modern computers use a function called segmented paging. Main memory is divided into variably-sized segments, which are then divided into smaller fixed-size pages on disk. Each segment contains a page table, and there are multiple page tables per process.

Each of the tables contains information on every segment page, while the segment table has information about every segment. Segment tables are mapped to page tables, and page tables are mapped to individual pages within a segment.

Advantages include less memory usage, more flexibility on page sizes, simplified memory allocation, and an additional level of data access security over paging. The process does not cause external fragmentation.

Paging and Segmentation: Advantages and Disadvantages

Paging Advantages

- On the programmer level, paging is a transparent function and does not require intervention.
- No external fragmentation.
- No internal fragmentation on updated OS's.
- Frames do not have to be contiguous.

Paging Disadvantages

- Paging causes internal fragmentation on older systems.
- Longer memory lookup times than segmentation; remedy with TLB memory caches.

Segmentation Advantages

- No internal fragmentation.
- Segment tables consume less space compared to page tables.
- Average segment sizes are larger than most page sizes, which allows segments to store more process data.
- Less processing overhead.
- Simpler to relocate segments than to relocate contiguous address spaces on disk.
- Segment tables are smaller than page tables, and takes up less memory.

Segmentation Disadvantages

- Uses legacy technology in x86-64 servers.
- Linux only supports segmentation in 80×86 microprocessors: states that paging simplifies memory management by using the same set of linear addresses.
- Porting Linux to different architectures is problematic because of limited segmentation support.
- Requires programmer intervention.
- Subject to serious external fragmentation.

Key Differences: Paging and Segmentation

Size:

- **Paging:** Fixed block size for pages and frames. Computer hardware determines page/frame sizes.
- **Segmentation:** Variable size segments are user-specified.

Fragmentation:

- **Paging:** Older systems were subject to internal fragmentation by not allocating entire pages to memory. Modern OS's no longer have this problem.
- **Segmentation:** Segmentation leads to external fragmentation.

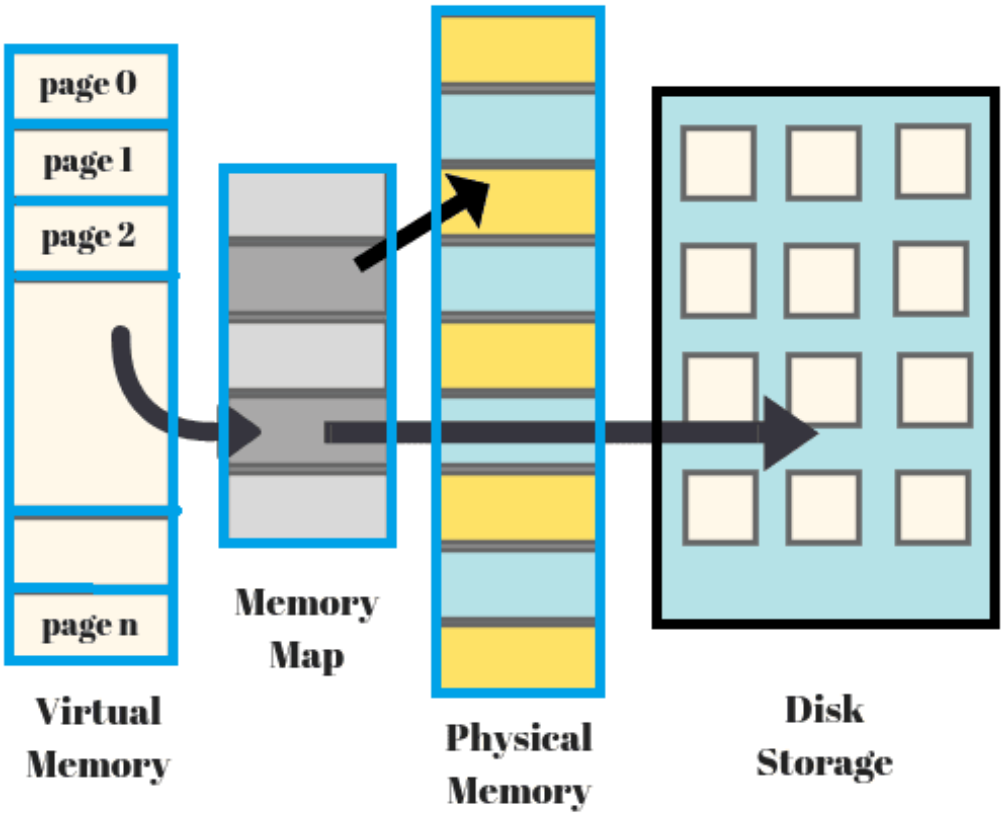
Tables:

- **Paging:** Page tables direct the MMU to page location and status. This is a slower process than segmentation tables, but TLB memory cache accelerates it.
- **Segmentation:** Segmentation tables contain segment ID and information, and are faster than direct paging table lookups.

Virtual Memory

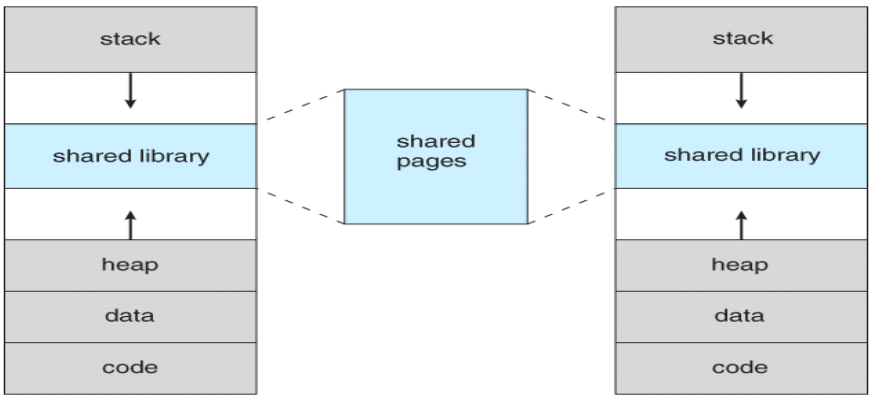
Virtual Memory is a Memory management technique that provides an "idealized abstraction of the storage resources that are actually available on a given machine"[which "creates the illusion to users of a very large (main) memory".

Figure below shows the general layout of virtual memory, which can be much larger than physical memory:



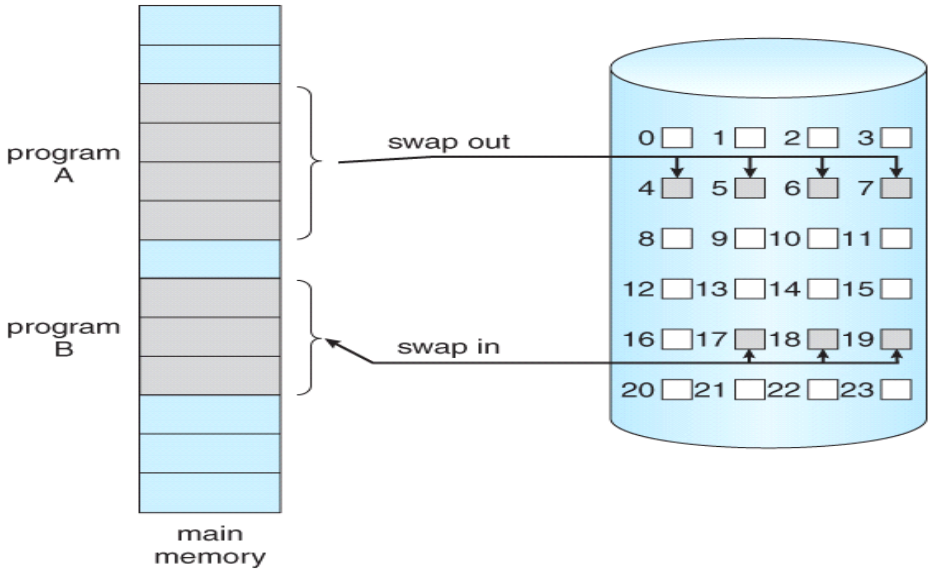
Virtual memory also allows the sharing of files and memory by multiple processes, with several benefits:

- System libraries can be shared by mapping them into the virtual address space of more than one process.
- Processes can also share virtual memory by mapping the same block of memory to more than one process.
- Process pages can be shared during a fork( ) system call, eliminating the need to copy all of the pages of the original (parent) process.



Demand Paging

The basic idea behind demand paging is that when a process is swapped in, its pages are not swapped in all at once. Rather they are swapped in only when the process needs them. (on demand) This is termed a lazy swapper, although a pager is a more accurate term.



Page Replacement

- In order to make the most use of virtual memory, we load several processes into memory at the same time. Since we only load the pages that are actually needed by each process at any given time, there is room to load many more processes than if we had to load in the entire process.
    - However memory is also needed for other purposes (such as I/O buffering), and what happens if some process suddenly decides it needs more pages and there aren't any free frames available? There are several possible solutions to consider:

1. Adjust the memory used by I/O buffering, etc., to free up some frames for user processes. The decision of how to allocate memory for I/O versus user processes is a complex one, yielding different policies on different systems. (Some allocate a fixed amount for I/O, and others let the I/O system contend for memory along with everything else.)

2. Put the process requesting more pages into a wait queue until some free frames become available.

3. Swap some process out of memory completely, freeing up its page frames.69

4. Find some page in memory that isn't being used right now, and swap that page only out to disk, freeing up a frame that can be allocated to the process requesting it. This is known as page replacement, and is the most common solution.

Page Replacement Algorithms

The page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).

Types of Page Replacement Algorithms

There are various page replacement algorithms. Each algorithm has a different method by which the pages can be replaced.

- **Optimal Page Replacement algorithm →** this algorithms replaces the page which will not be referred for so long in future. Although it can not be practically implementable but it can be used as a benchmark. Other algorithms are compared to this in terms of optimality.
- **Least recent used (LRU) page replacement algorithm →** this algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.
- **FIFO →** in this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault.

Unit 5
I/O Management Functions (Principles and Brief Concept)

Device management in operating system implies the management of the I/O devices such as a keyboard, magnetic tape, disk, printer, microphone, USB ports, scanner, and camcorder etc.as well as the supporting units like control channels. The basics of I/O devices can fall into 3 categories:

Block device: it stores information in fixed-size block, each one with its own address.Fr example, disks.

Character device: it's delivers or accepts a stream of characters. The individual characters are not addressable. For example printers, keyboards etc.

Network device: For transmitting data packets.

The main functions of device management in the operating system

An operating system or the OS manages communication with the devices through their respective drivers. The operating system component provides a uniform interface to access devices of varied physical attributes. For device management in operating system:

- Keep tracks of all devices and the program which is responsible to perform this is called I/O controller.
- Monitoring the status of each device such as storage drivers, printers and other peripheral devices.
- Enforcing preset policies and taking a decision which process gets the device when and for how long.
- Allocates and Deallocates the device in an efficient way.De-allocating them at two levels: at the process level when I/O command has been executed and the device is temporarily released, and at the job level, when the job is finished and the device is permanently released.
- Optimizes the performance of individual devices.

Types of devices

The OS peripheral devices can be categorized into 3: Dedicated, Shared, and Virtual. The differences among them are the functions of the characteristics of the devices as well as how they are managed by the Device Manager.

Dedicated devices

Such type of devices in the **device management in operating system** are dedicated or assigned to only one job at a time until that job releases them. Devices like printers, tape drivers, plotters etc. demand such allocation scheme since it would be awkward if several

users share them at the same point of time. The disadvantages of such kind f devices s the inefficiency resulting from the allocation of the device to a single user for the entire duration of job execution even though the device is not put to use 100% of the time.

Shared devices

These devices can be allocated o several processes. Disk-DASD can be shared among several processes at the same time by interleaving their requests. The interleaving is carefully controlled by the Device Manager and all issues must be resolved on the basis of predetermined policies.

Virtual Devices

These devices are the combination of the first two types and they are dedicated devices which are transformed into shared devices. For example, a printer converted into a shareable device via spooling program which re-routes all the print requests to a disk. A print job is not sent straight to the printer, instead, it goes to the disk(spool)until it is fully prepared with all the necessary sequences and formatting, then it goes to the printers. This technique can transform one printer into several virtual printers which leads to better performance and use.

Storage Devices

Storage devices are the computer hardware used to remember/store data.

There are many types of storage devices, each with their own benefits and drawbacks.

Below are explanations about different storage devices.

Jump to a section on this page:

- Solid State Drive
- Random Access Memory (RAM)
- CD, DVD and Blu-Ray Discs
- DVD-RAM
- ROM
- USB Flash Memory

Device Drivers:

A device driver is specialized software. It is specifically written to manage communication with an identified class of devices. For instance, a device driver is specially written for a printer with known characteristics. Different make of devices may differ in some respect, and therefore, shall require different drivers. More specifically, the devices of different makes may differ in speed, the sizes of buffer and the interface characteristics, etc. Nevertheless device drivers present a uniform interface to the OS. This is so even while managing to communicate with the devices which have different characteristics.

Spooling

Simultaneous peripheral operation online, acronym for this is Spooling. A spool is a kind of buffer that holds the jobs for a device till the device is ready to accept the job. Spooling considers disk as a huge buffer that can store as many jobs for the device till the output devices are ready to accept them.

In spooling, I/O of one job is overlapped with the computation of another job. For example, a spooler at a time may read input of one job, and at the same time, it may also print the output of another job.

Spooling can also process data at the remote sites. The spooler only has to notify when a process gets completed at the remote site so that spooler can spool next process to the remote side device.

Spooling increases the performance of the system by increasing the working rate of the devices. It naturally leads to multiprogramming.

Buffering

Before discussing buffering, let us discuss, what is the buffer? The buffer is an area in the main memory that is used to store or hold the data temporarily that is being transmitted either between two devices or between a device or an application. In simple words, buffer temporarily stores data that is being transmitted from one place to another. The act of storing data temporarily in the buffer is called buffering.

There are three reasons behind buffering of data, first is it helps in matching speed between two devices, between which the data is transmitted. For example, a hard disk has to store the file received from the modem.

Now, as we know the transmission speed of a modem is slow, as compared to the hard disk. So bytes coming from the modem is accumulated in the buffer space, and when all the bytes of a file has arrived at the buffer, the entire data is written to the hard disk in a single operation.

Secondly, it helps the devices with different **data transfer size** to get adapted to each other. It helps devices to manipulate data before sending or receiving. In computer networking, the large message is fragmented into the small fragments and sent over the network. At the receiving end, the fragments are accumulated in the buffer and reassembled to form the complete large message.

The **third** use of buffering is that it also supports **copy semantics**. With copy semantics, the version of data in the buffer is guaranteed to be the version of data at the time of system call irrespective of any subsequent change to data in the buffer. Buffering increases the performance of the device. It overlaps the i/o of one job with the computation of the same job.

Key Differences between Spooling and Buffering

- The key difference between spooling and buffering is that Spooling can handle the I/O of one job along with the computation of an another job at the same time while buffering handles I/O of one job along with its computation.
- Spooling is an acronym for Simultaneous Peripheral Operation online. However buffering is not an acronym.
- Spooling is more efficient than buffering, as it can overlap processing two jobs at a time.
- The buffer is a limited area in main memory while Spool uses the disk as a huge buffer.

Comparison Chart

BASIS FOR COMPARISON	**SPOOLING**	**BUFFERING**
Basic	Spooling overlap the I/O of one job with the computation of another job.	Buffer overlaps the I/O of one job with the computation of the same job.
Full form	Simultaneous peripheral operation online	No full form.
Efficient	Spooling is more efficient tha buffering.	Buffering is less efficeint than spooling.
Size	Spooling considers disk as a huge spool or buffer.	Buffer is a limited area in main memory.

Unit 6
File Management

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.

File System provide efficient access to the disk by allowing data to be stored, located and retrieved in a convenient way. A file System must be able to store the file, locate the file and retrieve the file.

Most of the Operating Systems use layering approach for every task including file systems. Every layer of the file system is responsible for some activities.

Operations on the File

There are various operations which can be implemented on a file. We will see all of them in detail.

1. Create

Creation of the file is the most important operation on the file. Different types of files are created by different methods for example text editors are used to create a text file, word processors are used to create a word file and Image editors are used to create the image files.

2. Write

Writing the file is different from creating the file. The OS maintains a write pointer for every file which points to the position in the file from which, the data needs to be written.

3. Read

Every file is opened in three different modes : Read, Write and append. A Read pointer is maintained by the OS, pointing to the position up to which, the data has been read.

4. Re-position

Re-positioning is simply moving the file pointers forward or backward depending upon the user's requirement. It is also called as seeking.

5. Delete

Deleting the file will not only delete all the data stored inside the file, It also deletes all the attributes of the file. The space which is allocated to the file will now become available and can be allocated to the other files.

6. Truncate

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

Physical and Logical File Systems

1. Physical files:

Physical files contain the actual data that is stored on a system, and a description of how data is to be presented to or received from a program. They contain only one record format, and one or more members. Records in database files can be described using either a field level description or record level description.

A field-level description describes the fields in the record to the system. Database files that are created with field level descriptions are referred to as externally described files.

2. Logical files:

Logical files do not contain data. They contain a description of records that are found in one or more physical files. A logical file is a view or representation of one or more physical files. Logical files that contain more than one format are referred to as multi-format logical files.

Physical versus Logical Files:

- **Physical File –**
  A collection of bytes stored on a disk or tape.
- **Logical File –**
  A "Channel" (like a telephone line) that hides the details of the file's location and physical format to the program.

When a program wants to use a particular file, "data", the operating system must find the physical file called "data" and make logical name by assigning a logical file to it. This logical file has a logical name which is what is used inside the program.

Physical File	**Logical File**
It occupies the portion of memory. It contains the original data.	It does not occupy memory space. It does not contain data.
A physical file contains one record format.	It can contain upto 32 record formats.
It can exist without logical file.	It cannot exist without physical file.
If there is a logical file for physical file, the physical file cannot be deleted until and unless we delete the logical file.	If there is a logical file for a physical file, the logical file can be deleted without deleting the physical file.

Logical Storage Views – viewed by users are a collection of files organized within directories and storage volumes.

• Logical file structure is independent of its physical implementation.

• Logical file structure "ignores".

Physical storage allocations – records can be stored in separate file locations. Data access methods and Data encoding methods.

Physical Storage Views – a collection of physical storage locations organized as a linear address space.

• File is subdivided into records.

• Record usually contains information about a single customer, thing such as a product in inventory, or an event.

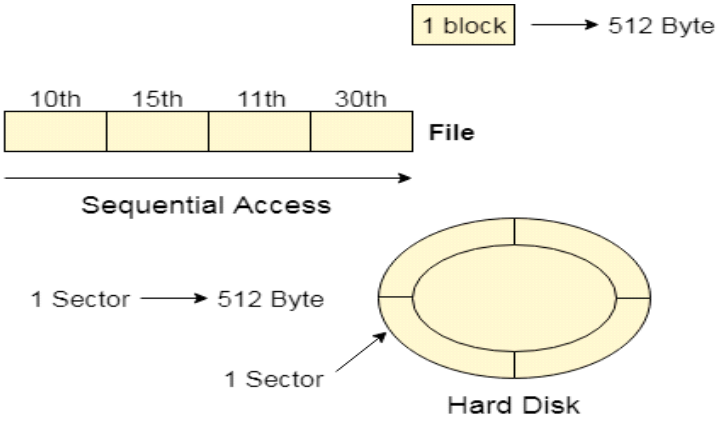
• Records are divided into fields.

• Fields are individual units of data.

File Access Methods

There are three methods to access files. These are as:

Sequential Access



Most of the operating systems access the file sequentially. In other words, we can say that most of the files need to be accessed sequentially by the operating system.

In sequential access, the OS read the file word by word. A pointer is maintained which initially points to the base address of the file. If the user wants to read first word of the file then the pointer provides that word to the user and increases its value by 1 word. This process continues till the end of the file.

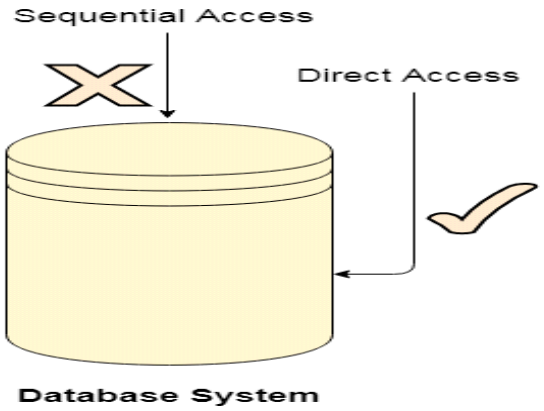
Modern word systems do provide the concept of direct access and indexed access but the most used method is sequential access due to the fact that most of the files such as text files, audio files, video files, etc need to be sequentially accessed.

Direct Access

The Direct Access is mostly required in the case of database systems. In most of the cases, we need filtered information from the database. The sequential access can be very slow and inefficient in such cases.

Suppose every block of the storage stores 4 records and we know that the record we needed is stored in 10th block. In that case, the sequential access will not be implemented because it will traverse all the blocks in order to access the needed record.

Direct access will give the required result despite of the fact that the operating system has to perform some complex tasks such as determining the desired block number. However, that is generally implemented in database applications.



Indexed Access

If a file can be sorted on any of the filed then an index can be assigned to a group of certain records. However, A particular record can be accessed by its index. The index is nothing but the address of a record in the file.

In index accessing, searching in a large database became very quick and easy but we need to have some extra space in the memory to store the index value.

Free Space Management

A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk. There are mainly two approaches by using which, the free blocks in the disk are managed.

1. Bit Vector

In this approach, the free space list is implemented as a bit map vector. It contains the number of bits where each bit represents each block.

If the block is empty then the bit is 1 otherwise it is 0. Initially all the blocks are empty therefore each bit in the bit map vector contains 1.

LAs the space allocation proceeds, the file system starts allocating blocks to the files and setting the respective bit to 0.

2. Linked List

It is another approach for free space management. This approach suggests linking together all the free blocks and keeping a pointer in the cache which points to the first free block.

Therefore, all the free blocks on the disks will be linked together with a pointer. Whenever a block gets allocated, its previous free block will be linked to its next free block.

Allocation Methods

There are various methods which can be used to allocate disk space to the files. Selection of an appropriate allocation method will significantly affect the performance and efficiency of the system. Allocation method provides a way in which the disk will be utilized and the files will be accessed.

There are following methods which can be used for allocation.

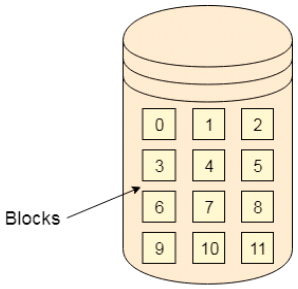
- Contiguous Allocation.

- Linked Allocation
- Indexed Allocation

Contiguous Allocation

If the blocks are allocated to the file in such a way that all the logical blocks of the file get the contiguous physical block in the hard disk then such allocation scheme is known as contiguous allocation.

In the image shown below, there are three files in the directory. The starting block and the length of each file are mentioned in the table. We can check in the table that the contiguous blocks are assigned to each file as per its need.



File Name	Start	Length	Allocated Blocks
abc.text	0	3	0,1,2
video.mp4	4	2	4,5
jtp.docx	9	3	9,10,11

Directory

Contiguous Allocation

Advantages

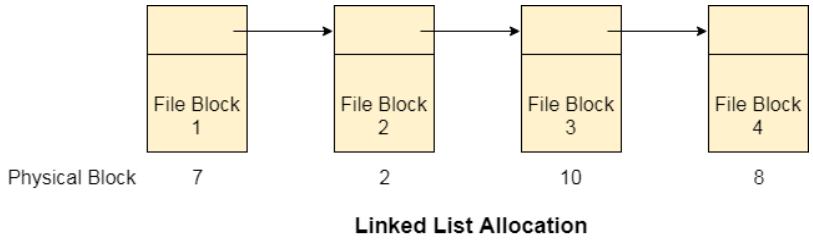
- It is simple to implement.
- We will get Excellent read performance.
- Supports Random Access into files.

Disadvantages

- The disk will become fragmented.
- It may be difficult to have a file grow.

Linked Allocation

Linked List allocation solves all problems of contiguous allocation. In linked list allocation, each file is considered as the linked list of disk blocks. However, the disks blocks allocated to a particular file need not to be contiguous on the disk. Each disk block allocated to a file contains a pointer which points to the next disk block allocated to the same file.



Linked List Allocation

Advantages

- There is no external fragmentation with linked allocation.
- Any free block can be utilized in order to satisfy the file block requests.
- File can continue to grow as long as the free blocks are available.
- Directory entry will only contain the starting block address.

Disadvantages

- Random Access is not provided.
- Pointers require some space in the disk blocks.
- Any of the pointers in the linked list must not be broken otherwise the file will get corrupted.
- Need to traverse each block.

Limitation of Indexed Allocation

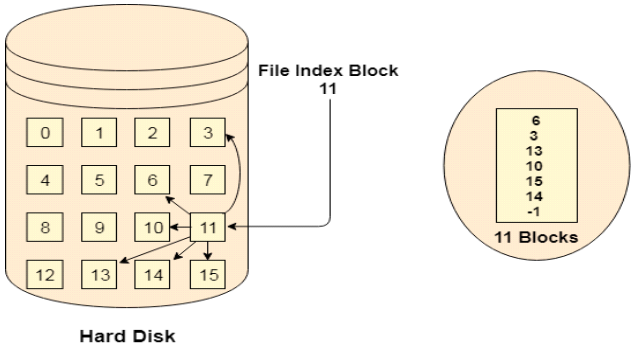
Limitation in the existing technology causes the evolution of a new technology. Till now, we have seen various allocation methods; each of them was carrying several advantages and disadvantages.

File allocation table tries to solve as many problems as possible but leads to a drawback. The more the number of blocks, the more will be the size of FAT.

Therefore, we need to allocate more space to a file allocation table. Since, file allocation table needs to be cached therefore it is impossible to have as many space in cache. Here we need a new technology which can solve such problems.

Indexed Allocation Scheme

Instead of maintaining a file allocation table of all the disk pointers, Indexed allocation scheme stores all the disk pointers in one of the blocks called as indexed block. Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file. Directory entry will only contain the index block address.



Advantages

- Supports direct access
- A bad data block causes the lost of only that block.

Disadvantages

- A bad index block could cause the lost of entire file.
- Size of a file depends upon the number of pointers, a index block can hold.
- Having an index block for a small file is totally wastage.
- More pointer overhead

Unit 7

Linux Operating System

History of Linux:

The History of Linux began in 1991 with the beginning of a personal project by a Finland student Linus Torvalds to create a new free operating system kernel. Since then, the resulting Linux kernel has been marked by constant growth throughout history.

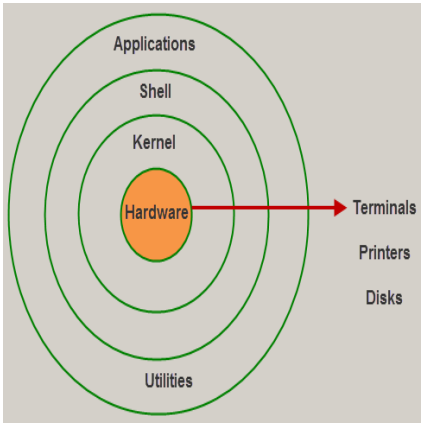
- In the year 1991, Linux was introduced by a Finland student Linus Torvalds.
- Hewlett Packard UNIX(HP-UX) 8.0 was released.
- In the year 1992, Hewlett Packard 9.0 was released.
- In the year 1993, NetBSD 0.8 and FreeBSD 1.0 released.
- In the year 1994, Red Hat Linux was introduced, Caldera was founded by Bryan Sparks and Ransom Love and NetBSD1.0 Released.
- In the year 1995, FreeBSD 2.0 and HP UX 10.0 were released.
- In the year 1996, K Desktop Environment was developed by Matthias Ettrich.
- In the year 1997, HP-UX 11.0 was released.
- In the year 1998, the fifth generation of SGI Unix i.e IRIX 6.5, Sun Solaris 7 operating system, and Free BSD 3.0 was released.
- In the year 2000, the agreement of Caldera Systems with the SCO server software division and the professional services division was announced.
- In the year 2001, Linus Torvalds released the Linux 2.4 version source code.
- In the year 2001, Microsoft filed a trademark suit against Lindows.com
- In the year 2004, Lindows name was changed to Linspire.
- In the year 2004, the first release of Ubuntu was released.
- In the year 2005, The project, openSUSE began a free distribution from Novell's community.
- In the year 2006, Oracle released its own distribution of Red Hat.
- In the year 2007, Dell started distributing laptops with Ubuntu pre-installed in it.
- In the year 2011, the Linux kernel 3.0 version was released.
- In the year 2013, Google Linux-based Android claimed 75% of the smartphone market share, in terms of the number of phones shipped.
- In the year 2014, Ubuntu claimed 22,000,000 users.
-

The Linux Distributions are listed below.

- Debian Linux.
- Arch Linux
- Gentoo Linux.
- Kali Linux Distribution
- Ubuntu Linux.
- Fedora Linux Distribution.
- Linux Mint Desktop.
- OpenSUSE
- RHEL Linux Distribution.
- CentOS Linux Distribution.

Linux System Architecture

The Linux Operating System's architecture primarily has these components: the Kernel, Hardware layer, System library, Shell, and System utility.



Architecture of Linux

1). The kernel is the core part of the operating system, which is responsible for all the major activities of the LINUX operating system. This operating system consists of different modules and interacts directly with the underlying hardware. The kernel offers the required abstraction to hide application programs or low-level hardware details to the system. The types of Kernels are as follows:

- Monolithic Kernel
- Microkernels
- Exo kernels
- Hybrid kernels

2). System libraries are special functions, that are used to implement the functionality of the operating system and do not require code access rights of kernel modules.

3). System Utility programs are liable to do individual and specialized-level tasks.

4).The hardware layer of the LINUX operating system consists of peripheral devices such as RAM, HDD, CPU.

5). The shell is an interface between the user and the kernel, and it affords services of the kernel. It takes commands from the user and executes the kernel's functions. The Shell is present in different types of operating systems, which are classified into two types: **command-line shells and graphical shells**.

The command-line shells provide a command-line interface, while the graphical line shells provide a graphical user interface. Though both shells perform operations, the graphical user interface shells perform slower than the command line interface shells. Types of shells are classified into four:

- Korn shell
- Bourne shell
- C shell
- POSIX shell

Features

The main **features of the Linux operating system** are

Portable: Linux operating system can work on different types of hardware as well as Linux kernel supports the installation of any kind of hardware platform.

Open Source: The source code of the LINUX operating system is freely available and, to enhance the ability of the LINUX operating system, many teams work in collaboration.

Multiuser: Linux operating system is a multiuser system, which means, multiple users can access the system resources like RAM, Memory, or Application programs at the same time.

Multiprogramming: Linux operating system is a multiprogramming system, which means multiple applications can run at the same time.

Hierarchical File System: Linux operating system affords a standard file structure in which system files or user files are arranged.

Shell: Linux operating system offers a special interpreter program, that can be used to execute commands of the OS. It can be used to do several types of operations like call application programs, and so on.

Security: Linux operating system offers user security systems using authentication features like encryption of data or password protection or controlled access to particular files.

How Does Linux be Different from other OS

There are several features of the Linux OS that demonstrate that it is superior as compared to other OS. But, some other OS can be more helpful than Linux. The main major advantages of the Linux system include the following and that will decide why it is superior as compared to other operating systems.

- Open Source
- Heavily Documented for beginners
- Security
- Multiple Desktop Support
- Multitasking
  Free
- Installation
- Lightweight
- Compatibility
- Stability
- Networking
- Performance
- Privacy
- Flexibility
- Community Support
- Software Updates
- Suitable for programmers

- Distributions/ Distros
- Graphical User Interface
- Live CD/USB

Difference between Linux and Windows Operating System

The difference between Linux and Windows OS include the following.

Linux Operating System	**Windows Operating System**
Linux is an open-source OS	Windows is not an open-source OS
The file name of Linux is case sensitive	The file name of Windows is case insensitive
It is free	It is commercial
In this OS, a monolithic kernel is used	In this OS, a microkernel is used
Linux is more efficient as compared to windows.	Windows is less efficient
To separate the directories, a forward slash is used	To separate the directories, the backslash is used
It is more secured	It is not secured as compared to Linux
Linux is extensively used to hack the systems	Windows do not offer much effectiveness in hacking.
Linux uses a hierarchical file system.	Windows uses several data drives namely C: D: E for the purpose of storing the files as well as folders.
The considered files in Linux are CD-ROMs, hard drives, & printers	The considered devices in windows are Hard drives, printers, CD-ROMs.
The user account types in Linux are 3 types like Regular, Root & Service Account	The user account types in Windows are four types like Administrator, Standard, Child, & Guest
The superuser like Root user of Linux includes all administrative human rights.	The administrator user of Windows includes all administrative human rights of computers.
The naming convention of Linux files is case-sensitive. So, two different files in this OS are sample & SAMPLE.	In Windows OS, you cannot have two files with the similar name within the same folder

For each user, his home directory is created like home or username.	In windows OS, the default home directory is My Documents

Linux Commands:

cat

This command outputs the contents of a text file. You can use it to read brief files or to concatenate files together.

To append file1 onto the end of file2, enter:

```
cat file1 >> file2
```

To view the contents of a file named myfile, enter:

```
cat myfile
```

Because cat displays text without pausing, its output may quickly scroll off your screen. Use the less command or an editor for reading longer text files.

cd

This command changes your current directory location. By default, your Unix login session begins in your home directory.

To switch to a subdirectory (of the current directory) named myfiles, enter:

```
cd myfiles
```

To switch to a directory named /home/dvader/empire_docs, enter:

```
cd /home/dvader/empire_docs
```

To move to the parent directory of the current directory, enter:

```
cd ..
```

To move to the root directory, enter:

```
cd /
```

To return to your home directory, enter:

```
cd
```

chmod

This command changes the permission information associated with a file. Every file (including directories, which Unix treats as files) on a Unix system is stored with records indicating who has permission to read, write, or execute the file, abbreviated as r, w, and x. These permissions are broken down for three categories of user: first, the owner of the file; second, a group with which both the user and the file may be associated; and third, all other users. These categories are abbreviated as u for owner (or user), g for group, and o for other.

To allow yourself to execute a file that you own named myfile, enter:

```
chmod u+x myfile
```

To allow anyone who has access to the directory in which myfile is stored to read or execute myfile, enter:

```
chmod o+rx myfile
```

You can view the permission settings of a file using the ls command, described below.

Be careful with the chmod command. If you tamper with the directory permissions of your home directory, for example, you could lock yourself out or allow others unrestricted access to your account and its contents.

cp

This command copies a file, preserving the original and creating an identical copy. If you already have a file with the new name, cp will overwrite and destroy the duplicate. For this reason, it's safest to always add -i after the cp command, to force the system to ask for your approval before it destroys any files. The general syntax for cp is:

```
cp -i oldfile newfile
```

To copy a file named meeting1 in the directory /home/dvader/notes to your current directory, enter:

```
cp -i /home/dvader/notes/meeting1 .
```

The . (period) indicates the current directory as destination, and the -i ensures that if there is another file named meeting1 in the current directory, you will not overwrite it by accident.

To copy a file named oldfile in the current directory to the new name newfile in the mystuff subdirectory of your home directory, enter:

```
cp -i oldfile ~/mystuff/newfile
```

date

The date command displays the current day, date, time, and year.

To see this information, enter:

```
date
```

kill

Use this command as a last resort to destroy any jobs or programs that you suspended and are unable to restart. Use the jobs command to see a list of suspended jobs. To kill suspended job number three, for example, enter:

```
kill %3
```

Now check the jobs command again. If the job has not been cancelled, harsher measures may be necessary. Enter:

```
kill -9 %3
```

less and more

Both less and more display the contents of a file one screen at a time, waiting for you to press the Spacebar between screens. This lets you read text without it scrolling quickly off your screen. The less utility is generally more flexible and powerful than more, but more is available on all Unix systems while less may not be.

To read the contents of a file named textfile in the current directory, enter:

```
less textfile
```

The less utility is often used for reading the output of other commands. For example, to read the output of the ls command one screen at a time, enter:

```
ls -la | less
```

In both examples, you could substitute more for less with similar results. To exit either less or more, press q. To exit less after viewing the file, press q.

ls

This command will list the files stored in a directory. To see a brief, multi-column list of the files in the current directory, enter:

```
ls
```

To also see "dot" files (configuration files that begin with a period, such as .login), enter:

```
ls -a
```

To see the file permissions, owners, and sizes of all files, enter:

```
ls -la
```

If the listing is long and scrolls off your screen before you can read it, combine ls with the less utility, for example:

```
ls -la | less
```

mkdir

This command will make a new subdirectory.

To create a subdirectory named mystuff in the current directory, enter:

```
mkdir mystuff
```

To create a subdirectory named morestuff in the existing directory named /tmp, enter:

```
mkdir /tmp/morestuff
```

mv

This command will move a file. You can use mv not only to change the directory location of a file, but also to rename files. Unlike the cp command, mv will not preserve the original file.

As with the cp command, you should always use -i to make sure you do not overwrite an existing file.

To rename a file named oldname in the current directory to the new name newname, enter:

```
mv -i oldname newname
```

To move a file named hw1 from a subdirectory named newhw to another subdirectory named oldhw (both subdirectories of the current directory), enter:

```
mv -i newhw/hw1 oldhw
```

If, in this last operation, you also wanted to give the file a new name, such as firsthw, you would enter:

```
mv -i newhw/hw1 oldhw/firsthw
```

pwd

This command reports the current directory path. Enter the command by itself:

```
 pwd
```

rm

This command will remove (destroy) a file. You should enter this command with the -i option, so that you'll be asked to confirm each file deletion. To remove a file named junk, enter:

```
rm -i junk
```

Note: Using rm will remove a file permanently, so be sure you really want to delete a file before you use rm.

To remove a non-empty subdirectory, rm accepts the -r option. On most systems this will prompt you to confirm the removal of each file. This behavior can be prevented by adding the -f option. To remove an entire subdirectory named oldstuff and all of its contents, enter:

```
rm -rf oldstuff
```

Note: Using this command will cause rm to descend into each subdirectory within the specified subdirectory and remove all files without prompting you. Use this command with caution, as it is very easy to accidently delete important files. As a precaution, use the ls command to list the files within the subdirectory you wish to remove. To browse through a subdirectory named oldstuff, enter:

```
ls -R oldstuff | less
```

rmdir

This command will remove a subdirectory. To remove a subdirectory named oldstuff, enter:

```
rmdir oldstuff
```

vi

This command starts the vi text editor. To edit a file named myfile in the current directory, enter:

```
vi myfile
```

w and who

The w and who commands are similar programs that list all users logged into the computer. If you use w, you also get a list of what they are doing. If you use who, you also get the IP numbers or computer names of the terminals they are using.