SRI GCSR COLLEGE

GMR NAGAR, RAJAM

2 B.Sc. (4th Semester)

Department Of Computer Science

Sri GCSR College

GMR Nagar, Rajam, Vizianagara - 532127

Unit-I

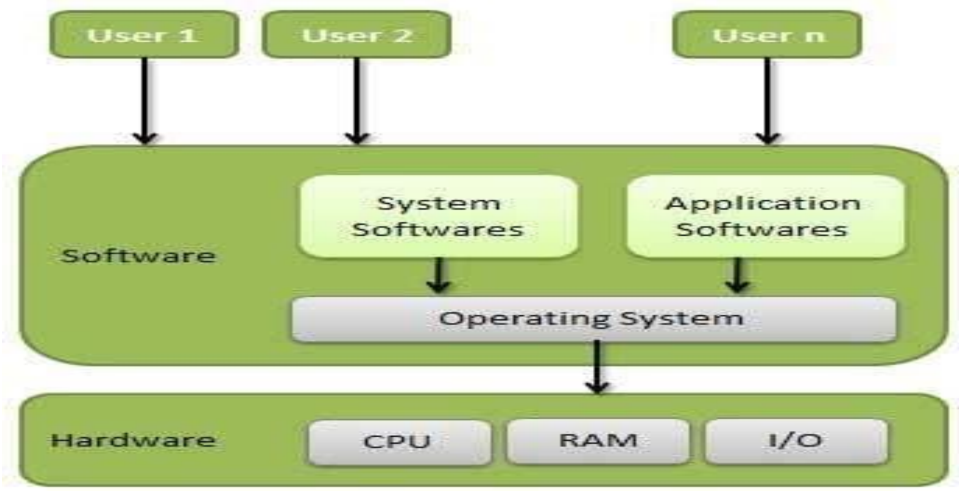
What is Operating System? History and Evolution of OS, Basic OS functions, Resource,Abstraction, Types of Operating Systems– Multiprogramming Systems, Batch Systems, TimeSharing Systems; Operating Systems for Personal Computers, Workstations and Hand-held Devices, Process Control & Real time Systems.

Q: WHAT IS OPERATING SYSTEM ?

Operating System can be defined as an interface between user and the hardware. It provides an environment to the user so that, the user can perform its task in convenient and efficient way.

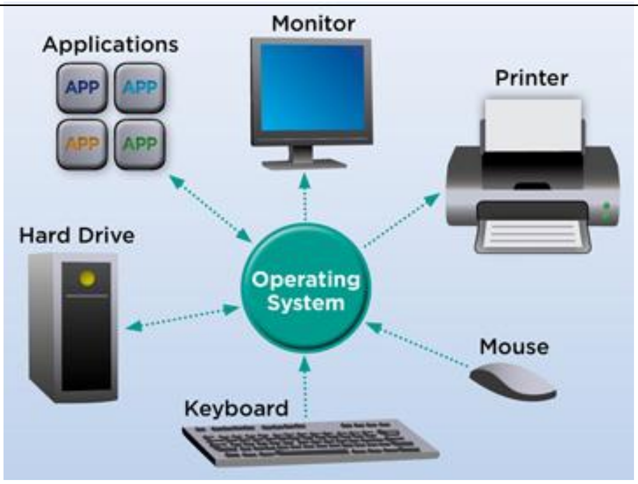
It is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks.

Some popular Operating Systems include Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.



Q: EXPLAIN THE Objectives of Operating System?

Operating System (OS) is one of the core software programs that runs on the hardware and makes it usable for the user to interact with the hardware so that they can send commands (input) and receive results (output). It provides a consistent environment for other software to execute commands.



The following are the basic key objectives of any operating system.

Interface between the user and the hardware:

An OS provides an interface between user and machine. This interface can be a graphical user interface (GUI) in which users click onscreen elements to interact with the OS or a command-line interface (CLI) in which users type commands at the command-line interface (CLI) to tell the OS to do things.

Coordinate hardware components:

An OS enables coordination of hardware components. Each hardware device speaks a different language, but the operating system can talk to them through the specific translational soft wares called device drivers. Every hardware component has different drivers for Operating systems. These drivers make the communication successful between the other soft wares and the hardware.

Provide environment for software to function:

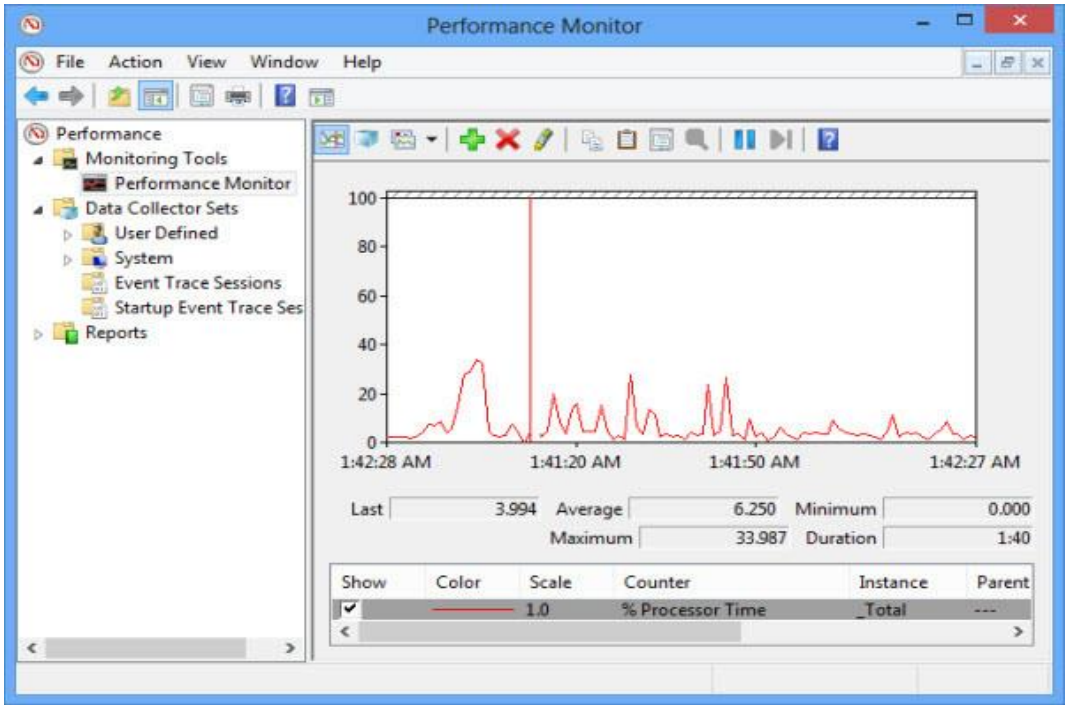
An OS provides an environment for software applications to function. Application software is specific software which is used to perform specific task. In GUI operating systems such as Windows and mac OS, applications run within a consistent, graphical desktop environment.

Provide structure for data management:

An OS displays structure/directories for data management. We can view file and folder listings and manipulate on those files and folders like (move, copy, rename, delete, and many others).

Monitor system health and functionality:

OS monitors the health of our system's hardware; giving us an idea of how well (or not) it's performing. We can see how busy our CPU is, or how quickly our hard drives retrieve data, or how much data our network card is sending etc. and it also monitors system activity for malware.



Last	3.994	Average	6.250	Minimum	0.000
		Maximum	33.987	Duration	1:40

Show	Color	Scale	Counter	Instance	Parent
✓	―	1.0	% Processor Time	_Total	---

Performance Monitor in windows

Q: EXPLAIN THE FUNCTIONS OF OPERATING SYSTEMS?

Following are some of important functions of an operating System.

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

Memory Management:

Current computer architectures arrange the computer's memory in a hierarchical manner, starting from the fastest registers, CPU cache, and random access memory and disk storage.

An operating system's memory manager coordinates the use of these various types of memory by tracking which one is available, which is to be allocated or de-allocated and how to move data between them.

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory.

An Operating System does the following activities for memory management −

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

Processor Management:

Process management is an operating system's way of dealing with running multiple processes.. Process management involves computing and distributing CPU time as well as other resources. Most operating systems allow a process to be assigned a priority which affects its allocation of CPU time.

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.

I/O Management:

Any Input / Output (I/O) devices present in the computer, such as keyboard, mouse, disk drives, printers, displays, etc require a significant amount of management. The Operating System allocates requests from applications to perform I/O to an appropriate device and provides convenient methods for using the device.

OS does the following activities for I/O 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.

Security:

There are numerous security threats to your computer, in particular various types of malware, which is short for malicious software. This includes computer viruses, which can interfere with the normal operations of your computer. Viruses can be very harmful and result in loss of data or system crashes.

Another basic security feature is to control access to your computer by setting up a password. Without the password someone else will not be able to get access to the software applications and files on your computer.

User Interface:

All operating systems need to provide an interface to communicate with the user. This could be a Command Line Interface or a Graphical User Interface.

A command line interface or CLI is a method of interacting with an operating system or software using a command line interpreter.

A **graphical user interface (GUI)** is a type of user interface which allows people to interact with a computer and computer-controlled devices which employ graphical icons, visual indicators or special graphical elements called widgets, along with text. Today, most modern operating systems contain GUIs.

Control over system performance − Recording delays between request for a service and response from the system.

Job accounting − Keeping track of time and resources used by various jobs and users.

Error detecting aids − Production of dumps, traces, error messages, and other debugging and error detecting aids.

Coordination between other software's and users − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

Q: Explain the history of OS?

The First Generation (1940's to early 1950's)

When electronic computers where first introduced in the 1940's they were created without any operating systems. All programming was done in absolute machine language, often by wiring up plugboards to control the machine's basic functions. During this generation computers were generally used to solve simple math calculations, **operating systems were not necessarily needed.**

The Second Generation (1955-1965)

The first operating system was introduced in the early 1950's, it was called **GMOS** and was created **by General Motors for IBM's** machine the 701. Operating systems in the 1950's were called single-stream **batch processing systems** because the data was submitted in groups. These new machines were called mainframes, and they were used by professional operators in large computer rooms. Since there was such as high price tag on these machines, only government agencies or large corporations were able to afford them.

The Third Generation (1965-1980)

By the late 1960's operating systems designers were able to develop the system of multiprogramming in which a computer program will be able to perform multiple jobs at the same time. The introduction of multiprogramming was a major part in the development of operating systems because it allowed a CPU to be busy nearly 100 percent of the time that it was in operation. Another major development during the third generation was the phenomenal growth of minicomputers,

Fourth Generation (1980-Now)

In the 4th Generation, the operating systems were used in personal computers also. This was made possible by **the Large-Scale Integration Circuits Development**. Some of the operating systems which were used at that time are, **UNIX and MS-DOS**.

In this generation the operating system is used for computer networks where users are aware of the existence of computers that are connected to one another.

Q: EXPLAIN OPERATING SYSTEM EVOLUTION?

Evolution of Operating System

The evolution of various types of operating systems can be briefly described as follows:

Serial Processing

Since 1950, the operating system started to be in use. Before 1950, there were no operating systems, and programmers had to directly communicate with the hardware.

The procedure is:

- First, type a program or punched card.
- Translate the punch card into a card reader.
- The translated card reader is submitted to the computer, and if any error has occurred, then with the help of the light, an error is indicated.
- The Programmer looks at the main memory and register to verify the reason behind the error.
- Then outputs are taken from the printers.

Drawback of Serial Processing

- The serial processing is tough for the users because, in serial processing, more time is required.
- The user cannot start executing another program if the previous program does not complete its execution. In serial processing, the programs are submitted to the computer one-by-one

BatchProcessing:

The Batch processing(similar kind of works) system was introduced in the second generation, where a job or a task that can be done in a series, and then executed sequentially. In this generation, the computer system is not equipped with an operating system, but several operating system functions exist like FileManagementSystem etc.

Multiprogramming

Multiprogramming means executing multiple programs at the same time with the help of a single processor. In this, multiple processes can exist in the main memory simultaneously. In multiprogramming, the operating system chooses one of the jobs from the main memory, and execute it.

Advantages of Multiprogramming

- In multiprogramming, the CPU will never sit ideal.
- Multiprogramming provides effective memory utilization.
- Throughput is an increase in multiprogramming

Time-Sharing System

In a time- sharing system, several users can share the system simultaneously. The reason behind its name 'time sharing' is in this system, the time of the processors is shared among the number of users simultaneously.

Parallel Processing System

In the parallel processing system, there are multiple processors, and, in this system, all the processors work concurrently. In this type of system, the job is divided into several sub-jobs, and then these sub-jobs are distributed among the processors that are present in the system. Parallel processing finishes the job in less time. This system is called a parallel processing system because, in this, multiple processors execute the job in a parallel manner.

Distributed System

Distributed systems are also known as loosely coupled systems. In a distributed system, two or more nodes are connected to each other, but the memory or a clock is not shared by the processors. With the help of communication lines like telephone lines or high-speed buses, the processors communicate with each other.

Q:Explain about Resource abstraction?

Resource abstraction is the process of “hiding the details of how the hardware operates, thereby making computer hardware relatively easy for an application programmer to use. Such an abstraction saves the programmer from needing to learn the details of both hardware interfaces

Q: What is the purpose of abstraction in operating system?

An abstraction is software that hides lower level details and provides a set of higher-level functions. An operating system transforms the physical world of devices, instructions, memory, and time into virtual world that is the result of abstractions built by the operating system.

Q:What is a resource in operating system?

Typical resources include the central processing unit (CPU), computer memory, file storage, input/output (I/O) devices, and network connections etc.Most programs, which complete a task and terminate, but operating system runs indefinitely and terminates only when the computer is turned off.

Q:Explain different types of operating systems?

TYPES OF OPERATING SYSTEMS:

Operating systems are there from the very first computer generation and they keep evolving with time. Some of the important types of operating systems which are most commonly used are

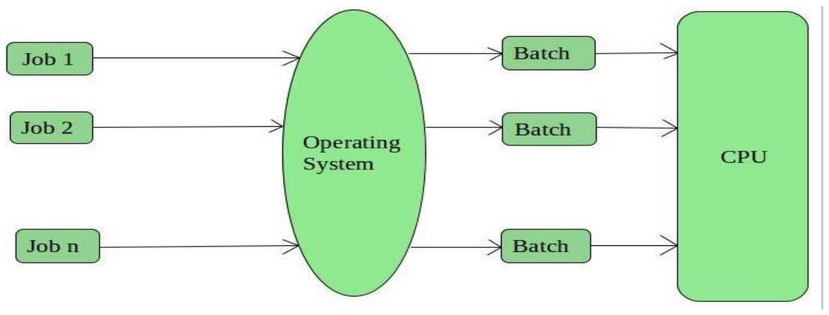
Batch operating system:

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 by the operator. 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.

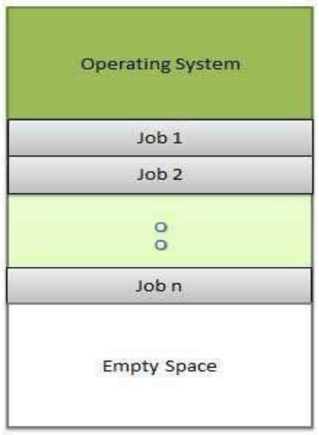


Multiprogramming

Sharing the processor, when two or more programs reside in memory at the same time, is referred as multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

The following figure shows the memory layout for a multiprogramming system.

An OS does the following activities related to multiprogramming.



- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the jobs in the memory.
- Multiprogramming operating systems monitor the state of all active programs and system resources using memory management programs to ensure that the CPU is never idle, unless there are no jobs to process.

Advantages:

- High and efficient CPU utilization.
- User feels that many programs are allotted CPU almost simultaneously.
- Disadvantages:
- CPU scheduling is required.
- To accommodate many jobs in memory, memory management is required.

Time-sharing operating system:

Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time.

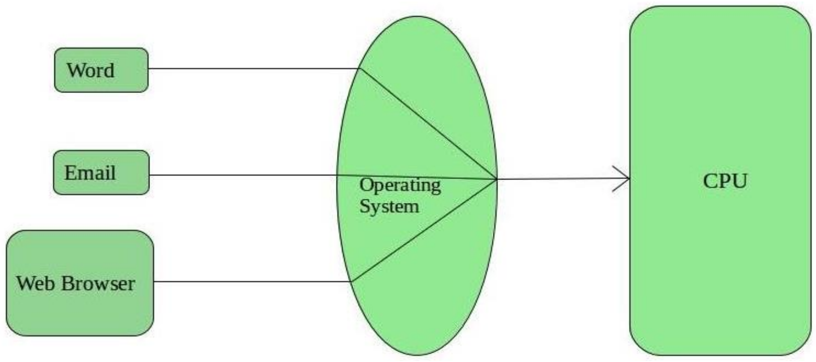
Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

The main difference between Batch Systems and Time-Sharing Systems is that in case of 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 seconds at most.

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.

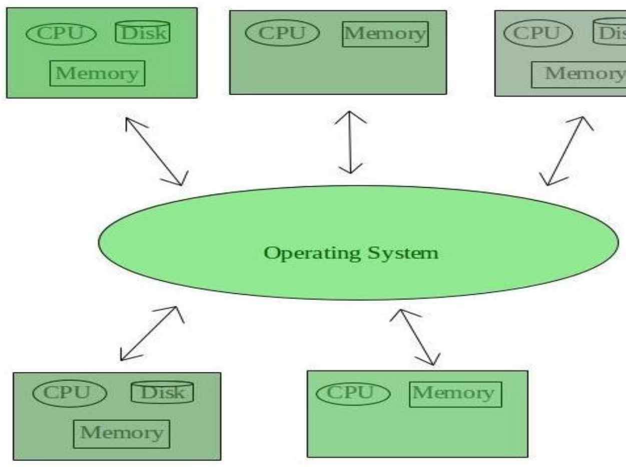
Disadvantages :

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

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

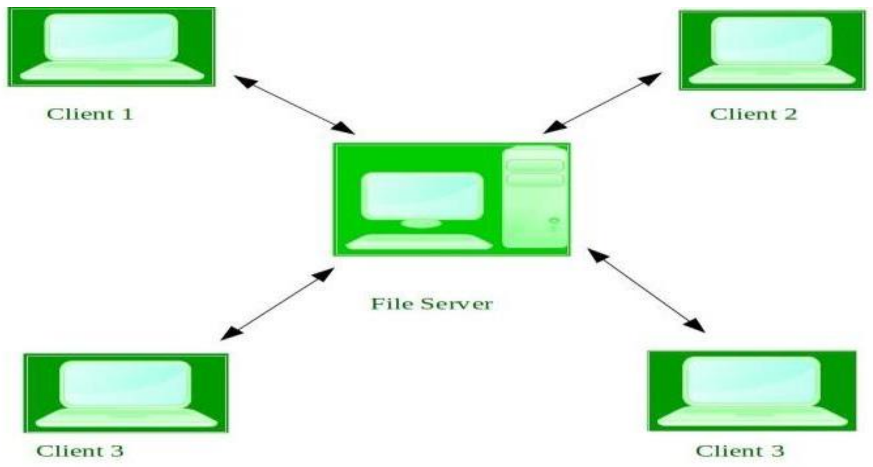
A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network,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

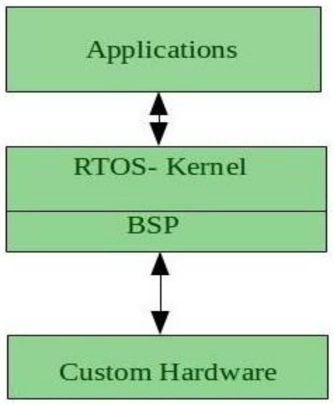
- The disadvantages of network operating systems are as follows:
- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.



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.

A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail.



For example, scientific experiments, flight simulation, 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.

Parallel operating systems:

Parallel operating systems are a type of computer processing platform that breaks large tasks into smaller pieces that are done at the same time in different places and by different mechanisms.

Parallel operating systems are designed to make efficient use of computers with multiple processors, and many operating systems.

The general idea is that different threads of the program can be executed simultaneously, speeding up the execution of the program as a whole. Programs written for single-processor computers can often run on a multi-processor system, but can only execute on one processor at a time.

Q: EXPLAIN OPERATING-SYSTEM SERVICES?

OS provide environments in which programs run, and services for the users of the system, including:

User Interfaces – Means by which users can issue commands to the system. Depending on the system these may be a command-line interface (e.g. sh, csh, ksh, tcsh, etc.), a GUI interface (e.g. Windows, X-Windows, KDE, Gnome, etc.), or a batch command systems. The latter are generally older systems using punch cards of job-control language, JCL, but may still be used today for specialty systems designed for a single purpose.

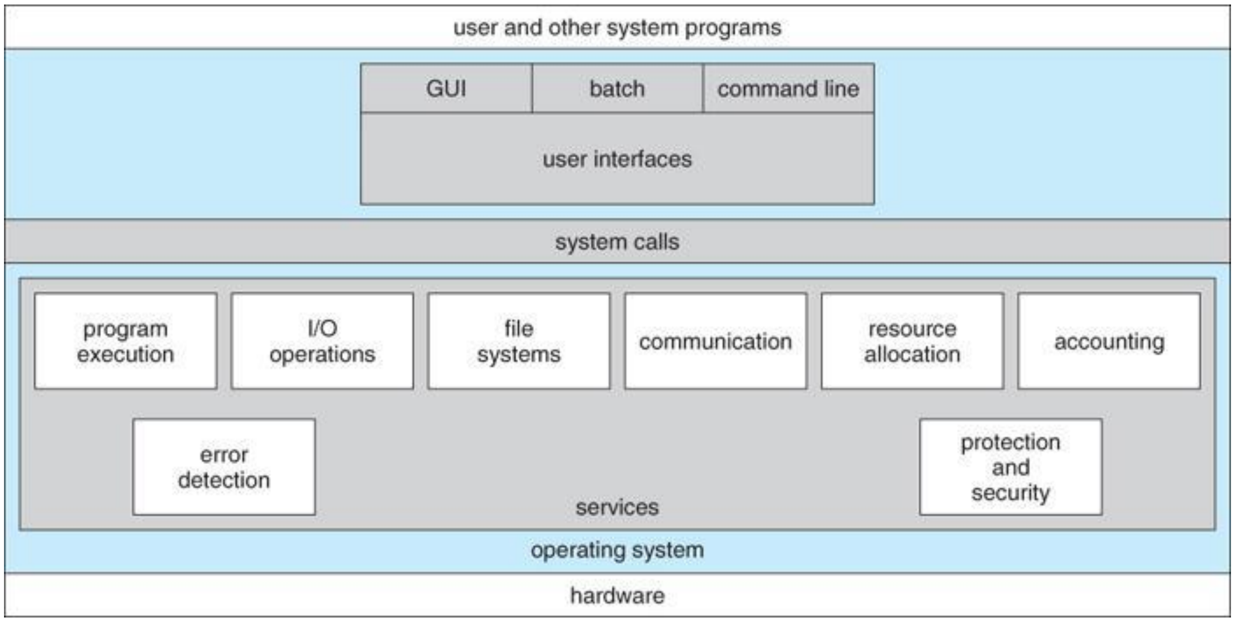
Program Execution – The OS must be able to load a program into RAM, run the program, and terminate the program, either normally or abnormally.

I/O Operations – The OS is responsible for transferring data to and from I/O devices, including keyboards, terminals, printers, and storage devices.

File-System Manipulation – In addition to raw data storage, the OS is also responsible for maintaining directory and subdirectory structures, mapping file names to specific blocks of data storage, and providing tools for navigating and utilizing the file system.

Communications – Inter-process communications, IPC, either between processes running on the same processor, or between processes running on separate processors or separate machines. May be implemented as either shared memory or message passing, (or some systems may offer both.

Error Detection – Both hardware and software errors must be detected and handled appropriately, with a minimum of harmful repercussions. Some systems may include complex error avoidance or recovery systems, including backups, RAID drives, and other redundant systems. Debugging and diagnostic tools aid users and administrators in tracing down the cause of problems.



Other systems aid in the efficient operation of the OS itself:

Resource Allocation – E.g. CPU cycles, main memory, storage space, and peripheral devices. Some resources are managed with generic systems and others with very carefully designed and specially tuned systems, customized for a particular resource and operating environment.

Accounting – Keeping track of system activity and resource usage, either for billing purposes or for statistical record keeping that can be used to optimize future performance.

Protection and Security – Preventing harm to the system and to resources, either through wayward internal processes or malicious outsiders. Authentication, ownership, and restricted access are obvious parts of this system. Highly secure systems may log all process activity down to excruciating detail, and security regulation dictate the storage of those records on permanent non-erasable medium for extended times in secure (off-site) facilities.

Q: Write Operating Systems for personal computers?

Personal Computer (PC)

A personal computer is a general-purpose, cost-effective computer that is designed to be used by a single end-user. Every PC is dependent on microprocessor technology, which allows PC makers to set the entire central processing unit (CPU) on a single chip

A PC can be a microcomputer, desktop computer, a laptop computer.

The three most common operating systems for personal computers are **Microsoft Windows, macOS, and Linux.**

Windows

Windows is a graphical operating system developed by Microsoft. It allows users to view and store files, run the software, play games, watch videos, and provides a way to connect to the internet. It was released for both home computing and professional works.

Microsoft introduced the first version as 1.0 on 10 November 1983. Later, it was released on many versions of Windows as well as the current version, Windows 10.

macOS

MacOS is the computer operating system (OS) for Apple desktops and laptops. It is a proprietary graphical OS that powers every Mac. OS es interact with a computer's hardware, allocating the resources necessary to complete tasks given to it.

Linux

Linux is an open-source operating system like other operating systems such as Microsoft Windows, Apple Mac OS, iOS, Google android, etc. An operating system is a software that enables the communication between computer hardware and software. It conveys input to get processed by the processor and brings output to the hardware to display it. This is the basic function of an operating system.

Q: Write Operating Systems for Handleld devices?

Hhandleld is any portable device that can be carried and held in any one palm’s. A handheld is primarly designed for communication.

The handheld devices are PDA(personal DigitalAssistants),Tablet PCS etc

The OS runs on these devices are called handleld Os

Most common os are Andriod,ios,blackberry.

Android Operating System

Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets

iOS: IOS is a mobile operating system developed and distributed by Apple Inc. It was originally released in 2007 for the iPhone, iPod Touch, and Apple TV. iOS is Apple's mobile version of the OS X operating system used in Apple computers.

BlackBerry OS is a proprietary mobile operating system developed by Research in Motion for its BlackBerry line of smartphones and handheld devices.

Symbian: OS is an operating system designed for mobile devices. Symbian was the leading smartphone platform up from 2003 up until 2010 (even 2011 for Europe). After that Google's Android OS took the lead

Q: Write Operating Systems for Real time Systems?

Real-time operating system (RTOS) is an operating system intended to serve real time application that process data as it comes in, mostly without buffer delay. The full form of RTOS is Real time operating system.

Examples of the real-time operating systems: Airline traffic control systems, Command Control Systems, Airlines reservation system, Heart Pacemaker, Network Multimedia Systems, Robot etc.

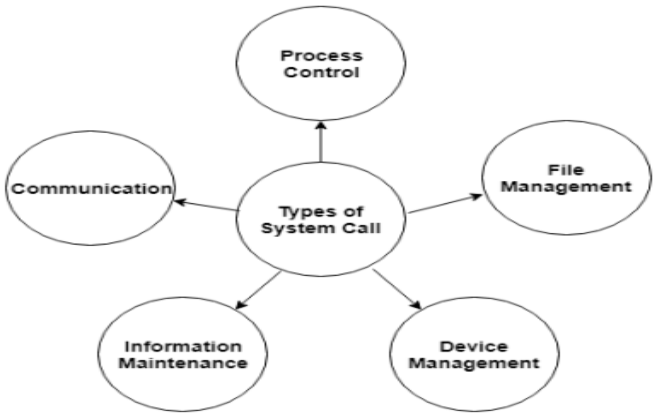
Q:Explain about System Call?

A system call is a method for a computer program to request a service from the kernel of the

on which it is running. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel..

Types of System Calls

There are mainly five types of system calls. These are explained in detail as follows −



Here are the types of system calls −

Process Control

These system calls deal with processes such as process creation, process termination etc.

File Management

These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

Device Management

These system calls are responsible for device manipulation such as reading from device buffers, writing into device buffers etc.

Information Maintenance

These system calls handle information and its transfer between the operating system and the user program.

Communication

These system calls are useful for interprocess communication. They also deal with creating and deleting a communication connection.

Q: Explain the advantages OS?

There are many advantages to operating systems. Some advantages are mentioned here for having a basic understanding.

1. OS Provides Graphical User Interface (GUI) in the form of menu, icons, and buttons.
2. OS manage the memory by memory management techniques. e.g, paging, swapping, Read More segmentation Read More etc.
3. OS manage the input and output. All I/O devices are managed by OS.
4. OS manage resource allocation. OS gives the resources to the process in such a way that all processes can use the resources in an efficient way. A resource can be a file or in the form of software or hardware etc.
5. OS convert a program into the process.
6. OS is responsible to synchronize the processes. Monitors and semaphores are very commonly used to synchronize the processes.
7. OS is responsible to manage the processes by making a process simple by threads.
8. OS manage the interrupts and handle the interrupts.
9. OS is responsible to schedule the process for the execution on CPU. Some common process scheduling techniques are first come-first served, round robin priority scheduling and shortest job first scheduling

UNIT-II

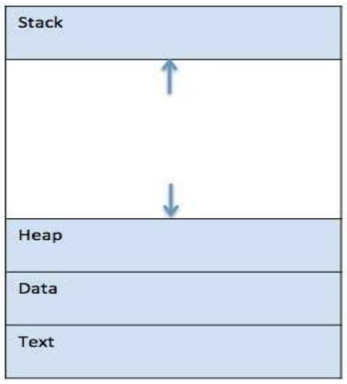
Processor and User Modes, Kernels, System Calls and System Programs, System View of the Process and Resources, ProcessAbstraction, ProcessHierarchy, Threads, Threading Issues, Thread Libraries; Process Scheduling, Non-Preemptive and Preemptive Scheduling Algorithms.

Q: Explain about PROCESS ?

A process is basically a program in execution. The execution of a process must progress in a sequential fashion. A process is defined as an entity which represents the basic unit of work to be implemented in the system.

To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections – stack, heap, text and data. The following image shows a simplified layout of a process inside main memory-



S.N.	Component & Description
1	**Stack**: The process Stack contains the temporary data such as method/function parameters, return address and local variables.
2	**Heap:** This is dynamically allocated memory to a process during its run time.
3	**Text:** This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
4	**Data**: This section contains the global and static variables.

Program:

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language.

For example, here is a simple program written in C programming language −

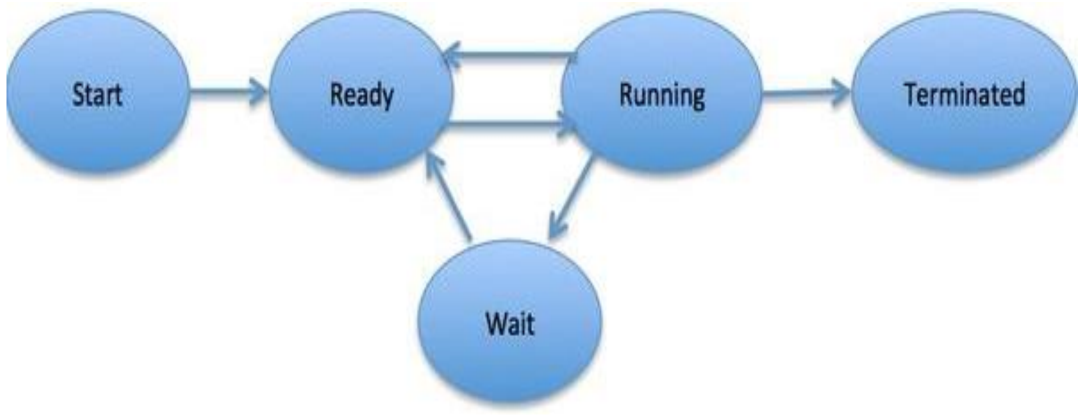
```
#include <stdio.h>
int main()
        {
printf("Hello, \n");
return 0;    }
```

A computer program is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a static instance computer program.

Q: Explain about process life cycle?

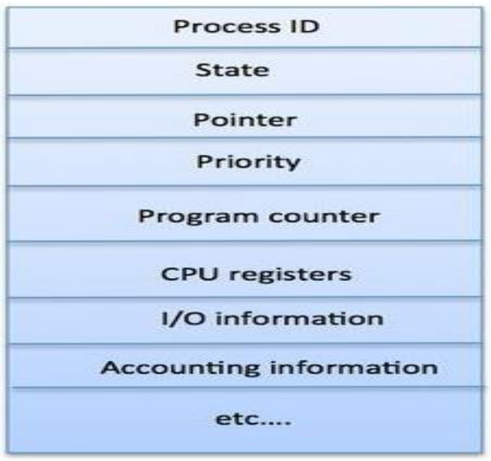
When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.
In general, a process can have one of the following five states at a time.

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 user input, or waiting for a file to become available.
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.



Q: Explain about PROCESS CONTROL BLOCK (PCB)?

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process as listed below in the table –



S.N.	Information & Description
1	Process State: The current state of the process i.e., whether it is ready, running, waiting, or whatever.
2	Process privileges: This is required to allow/disallow access to system resources.
3	Process ID: Unique identification for each of the process in the operating system.
4	Pointer: A pointer to parent process.
5	Program Counter: Program Counter is a pointer to the address of the next instruction to be executed for this process.
6	CPU registers: Various CPU registers where process need to be stored for execution for running state.
7	CPU Scheduling Information: Process priority and other scheduling information which is required to schedule the process.
8	Memory management information: This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
9	Accounting information: This includes the amount of CPU used for process execution, time limits, execution ID etc.
10	IO status information: This includes a list of I/O devices allocated to the process.

The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems. Here is a simplified diagram of a PCB −

The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.

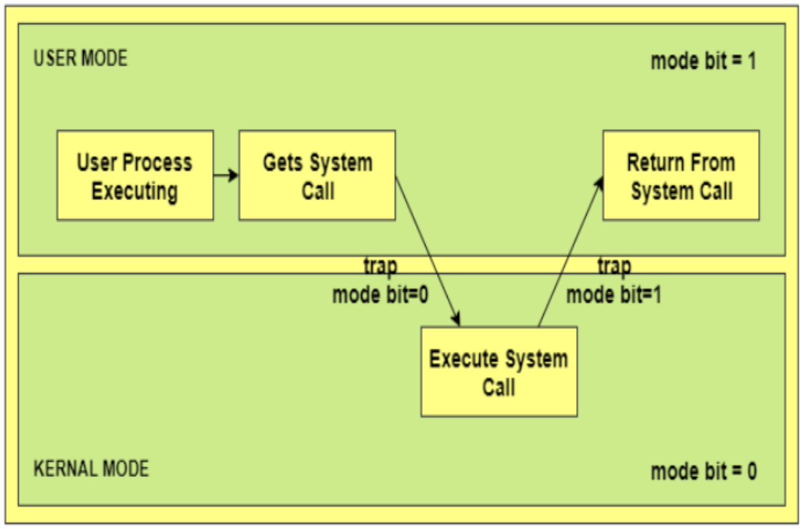
Q: Explain about Processor Modes?

There are two modes of operation in the operating systemThose sre

1. Kernal mode
2. User mode

Kernel mode: It is also known as system mode, is one of the central processing unit (CPU) operating modes. **While processes run in kernel mode, they have unrestricted access to the hardware.**

user mode: which is a non-privileged mode for user programs. **Therefore, when a process runs in user mode, it has limited access to the CPU and the memory.**



Q:Explain about System Calls?

System calls are special functions that manage OS routines that occur in kernel mode. system calls work as:

- To specify which particular service it needs from the OS, the user program places some values in registers or generates a stack frame with arguments.
- Then the user program executes the trap instruction.
- Trap instructions are part of the OS and they have memory protection; therefore, they cannot be modified by user programs. Also, they are not readable by user programs.
- The trap or system call handler instructions read the information of the requested service and then execute the request in kernel mode.
- When the system call is completed, the OS returns to user mode and exits the system call.

A system call can be used when we:

- need to access privileged information like process id (pid), or changing scheduler policy
- access an I/O device or a file
- need to change the execution context

Q: Explain about different Views of OS?

Operating System can be viewed from two viewpoints– User views & System views

1. User Views:-

- The user's view of the operating system depends on the type of user.
- If the user is using standalone system, then OS is designed for ease of use and high performances. Here resource utilization is not given importance.
- If the users are at different terminals connected to a mainframe or minicomputers, by sharing information and resources, then the OS is designed to maximize resource utilization. OS is designed such that the CPU time, memory and i/o are used efficiently and no single user takes more than the resource allotted to them.
- If the users are in workstations, connected to networks and servers, then the user have a system unit of their own and shares resources and files with other systems. Here the OS is designed for both ease of use and resource availability (files).
- Users of hand held systems, expects the OS to be designed for ease of use and performance per amount of battery life.
- Other systems like embedded systems used in home devies (like washing m/c) & automobiles do not have any user interaction.

2. System Views:-

Operating system can be viewed as a resource allocator and control program.

Resource allocator – The OS acts as a manager of hardware and software resources. CPU time, memory space, file-storage space, I/O devices, shared files etc. are the different resources required during execution of a program. There can be conflicting request for these resources by different programs running in same system. The OS assigns the resources to the requesting program depending on the priority.

Control Program – The OS is a control program and manage the execution of user program to prevent errors and improper use of the computer

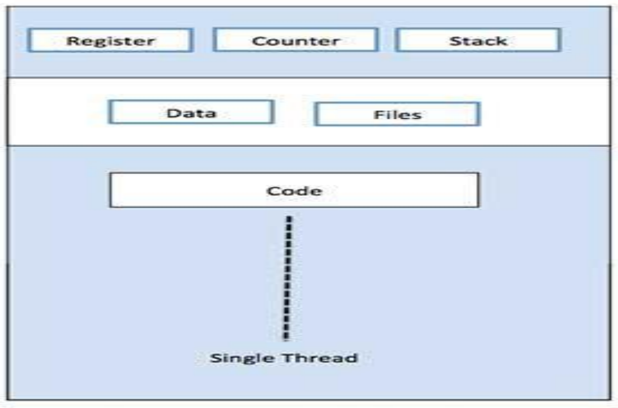
Q: WHAT IS THREAD AND EXPLAIN THEIR TYPES?

A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

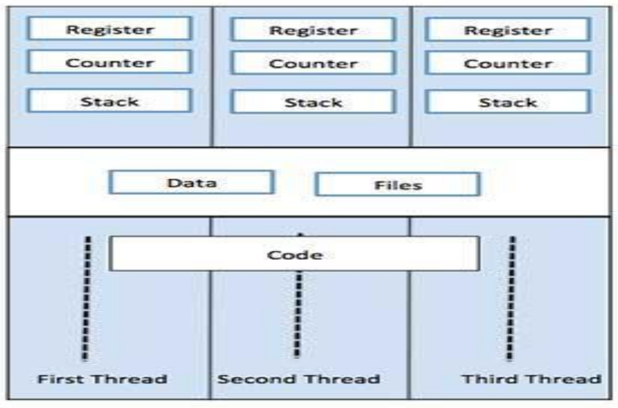
A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. The following figure shows the working of a single-threaded and a multithreaded process.



Single Process P with single thread



Single Process P with three threads

Difference between Process and Thread:

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.

4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

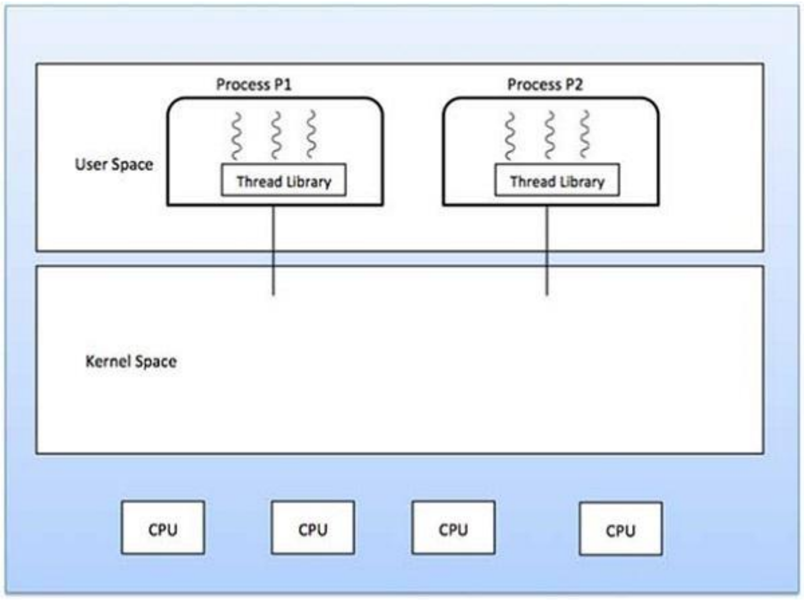
Types of Thread:

Threads are implemented in following two ways −

- **User Level Threads** − User managed threads.
- **Kernel Level Threads** − Operating System managed threads acting on kernel, an operating system core.

User Level Threads

In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.



Advantages:

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.

Kernel Level Threads:

In this case, thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

Difference between User-Level & Kernel-Level Thread

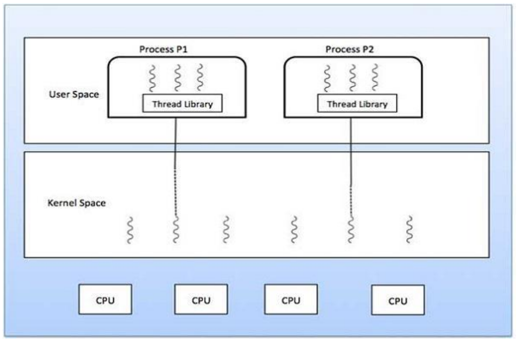
S.N.	**User-Level Threads**	**Kernel-Level Thread**
1	User-level threads are faster to create and manage.	Kernel-level threads are slower to create and manage.
2	Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads.
3	User-level thread is generic and can run on any operating system.	Kernel-level thread is specific to the operating system.
4	Multi-threaded applications cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.

Multithreading Models:

Some operating system provides a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

- Many to many relationship.
- Many to one relationship.
- One to one relationship.

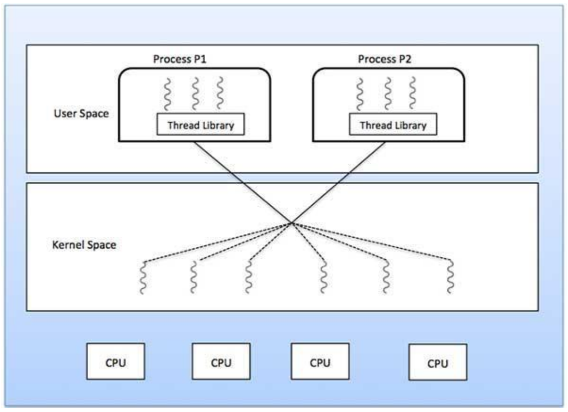
Many to Many Model



The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads.

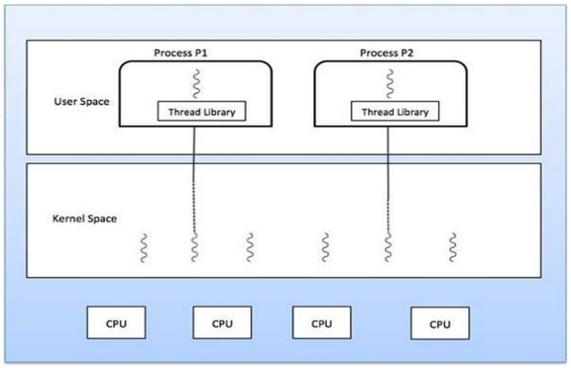
In this model, developers can create as many user threads as necessary and the corresponding Kernel threads can run in parallel on a multiprocessor machine. This model provides the best accuracy on concurrency and when a thread performs a blocking system call, the kernel can schedule another thread for execution.

Many to One Model



Many-to-one model maps many user level threads to one Kernel-level thread. Thread management is done in user space by the thread library. When thread makes a blocking system call, the entire process will be blocked. Only one thread can access the Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.

One to One Model



There is one-to-one relationship of user-level thread to the kernel-level thread. This model provides more concurrency than the many-to-one model. It also allows another thread to run when a thread makes a blocking system call. It supports multiple threads to execute in parallel on microprocessors.

Q: Explain about Thread Libraries?

- A thread library provides the programmer API for creating and managing threads.
- There are two primary ways of implementing a thread library.
- The first approach is to provide a library entirely in user space with no kernel support. All code and data structures for the library exist in user space. This means that invoking a function in the library results in a local function call in user space and not a system call.
- The second approach is to implement a kernel-level library supported directly by the operating system. In this case, code and data structures for the library exist in kernel space. Invoking a function in the API for the library typically results in a system call to the kernel.

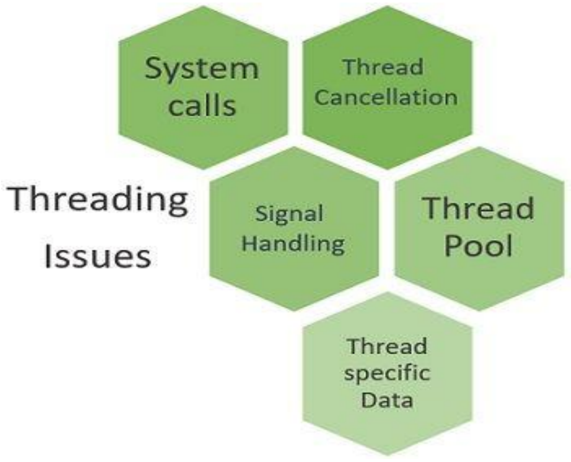
Three main thread libraries are in use today:

- POSIX Pthreads,
- Win32, and
- Java. Pthreads,

1. the threads extension of the POSIX standard, may be provided as either a user- or kernel-level library.
2. The Win32 thread library is a kernel-level library available on Windows systems.
3. The Java thread API allows thread creation and management directly in Java programs. However, because in most instances the JVM is running on top of a host operating system, the Java thread API is typically implemented using a thread library available on the host system.

Q:Explain about thread issues?

Some of the issues consider in designing multithreaded programs are



The fork() and exec() system calls

The fork() is used to create a duplicate process. The meaning of the fork() and exec() system calls change in a multithreaded program.

If one thread in a program which calls fork(), does the new process duplicate all threads, or is the new process single-threaded?

If a thread calls the exec() system call, the program specified in the parameter to exec() will replace the entire process which includes all threads?

Signal Handling

Generally, signal is used in UNIX systems to notify a process that a particular event has occurred. A signal received either synchronously or asynchronously, based on the source of and the reason for the event being signalled.

All signals, whether synchronous or asynchronous, follow the same pattern as given below −

- A signal is generated by the occurrence of a particular event.
- The signal is delivered to a process.
- Once delivered, the signal must be handled.

Cancellation

Thread cancellation is the task of terminating a thread before it has completed.

For example − If multiple database threads are concurrently searching through a database and one thread returns the result the remaining threads might be cancelled.

A target thread is a thread that is to be cancelled, cancellation of target thread may occur in two different scenarios −

- Asynchronous cancellation − One thread immediately terminates the target thread.
- Deferred cancellation − The target thread periodically checks whether it should terminate, allowing it an opportunity to terminate itself in an ordinary fashion.

Thread polls

Multithreading in a web server, whenever the server receives a request it creates a separate thread to service the request.

Some of the problems that arise in creating a thread are as follows −

- The amount of time required to create the thread prior to serving the request together with the fact that this thread will be discarded once it has completed its work.
- If all concurrent requests are allowed to be serviced in a new thread, there is no bound on the number of threads concurrently active in the system.
- Unlimited thread could exhaust system resources like CPU time or memory.

A thread pool is to create a number of threads at process start-up and place them into a pool, where they sit and wait for work.

Q: Explain about PROCESS SCHEDULING?

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

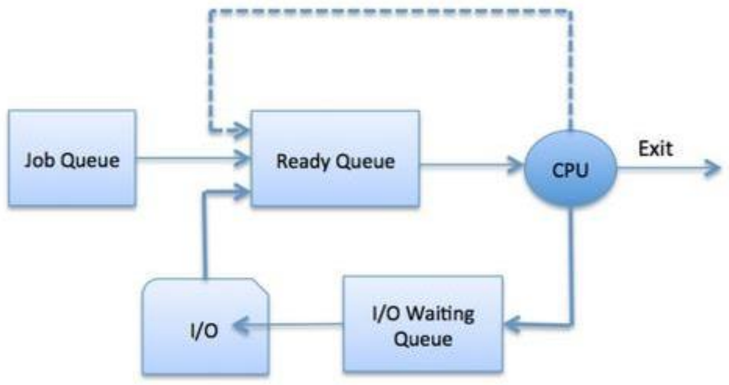
In an operating system (OS), a process scheduler performs the important activity of scheduling a process between the **ready queue and waiting queue** and allocating them to the **CPU**. The OS assigns priority to each process and maintains these queues. The scheduler selects the process from the queue and loads it into memory for execution.

Multi-programming environment provides maximum CPU utilization by executing multiple processes simultaneously.

PROCESS SCHEDULING QUEUES:

The Operating System maintains the following important process scheduling queues −

A. **Job queue** − This queue keeps all the processes in the system.
B. **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.
C. **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.

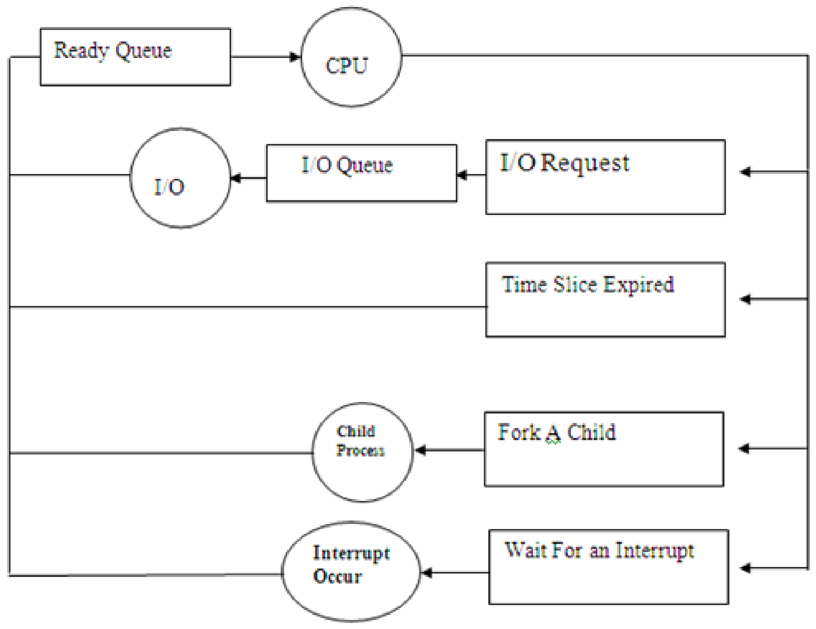


When process enters in the system, they are put into a **job queue**. The queue consists of all processes in the system.

The processes that are stored in main memory and ready to execute are kept on a list called **ready queue.** Since system has many processes, the disk may be busy with some other processes for input/output operation.

The list of processes waiting for a particular input/output device is called a **device queue**. Each device has its own queue that contains all processes which are waiting for that device. Enqueue process is initially put in ready queue. It waits in ready queue until it is selected for execution.

Once the process is assigned to the CPU, then it executes. During execution of the process one of several events occurs in the process and issue an input/output request and then places in an input output queue. The process could create a new sub-process and waits for termination. A process can be removed from the CPU as a result of interrupt and put back in ready queue.



Queuing Diagram representing Process Scheduling

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.

Two-State Process Model

Two-state process model refers to running and non-running states which are described below:

S.N.	State & Description
1	**Running** When a new process is created, it enters into the system as in the running state.
2	**Not Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Q: Explain about processor schedulers?

Schedulers are special system software which handle 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 –**

1. Long-Term Scheduler
2. Short-Term Scheduler
3. 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.

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

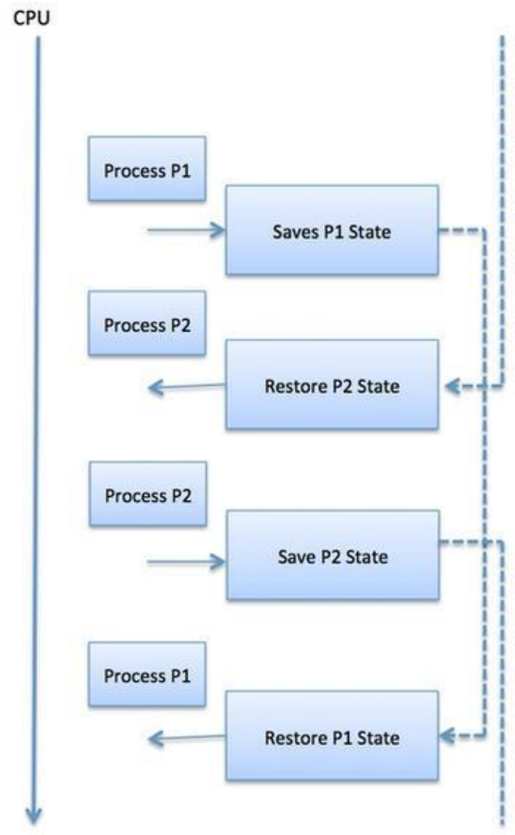
Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

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 Scheduler:

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.

Q: Explain about 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

Q: Explain cpu scheduler and types of process scheduling?

Whenever the CPU becomes idle, it is the job of the CPU Scheduler (the short-term scheduler) to select another process from the ready queue to run next.

The storage structure for the ready queue and the algorithm used to select the next process are not necessarily a FIFO queue. There are several alternatives to choose from, as well as numerous adjustable parameters for each algorithm.

There are two types of process scheduling: **preemptive scheduling**
non-preemptive scheduling.

A. **Preemptive Scheduling:**

The scheduling in which a running process can be interrupted if a high priority process enters the queue and is allocated to the CPU is called **preemptive scheduling**. In this case, the current process switches from the running queue to ready queue, and the high priority process utilizes the CPU cycle.

Let's take an example where four processes, P0, P1, P2, and P3, have different arrival times in the queue. Based on their priority or burst time, these processes are interrupted and allocated the CPU.

A process P2 arrives at time 0 and allocated the CPU. Process P3 arrives at time 1, before P2 finishes execution. The time remaining for P2 is 5 milliseconds, which is larger than a time required for P3 (4 milliseconds). So CPU is allocated to process P3. Process P1 arrives at time 2. P3 continues to execute because the remaining time for P3 (3 milliseconds) is less than the time required by processes P1 (4 milliseconds) and P2 (5 milliseconds).

Process P0 arrives at time 3. Now P3 continues to run because the remaining time for P3 (2 milliseconds) is equal to the time required by P0 (2 milliseconds). After P3 finishes, the CPU is allocated to P0 as it has smaller burst time than other processes. Later, the CPU is allocated to P1 and then to P2.

B. **Non-Preemptive Scheduling**

The scheduling in which a running process cannot be interrupted by any other process is called **non-preemptive scheduling**. Any other process which enters the queue has to wait until the current process finishes its CPU cycle. Let's take the same example and apply non-preemptive scheduling to it.

Process P2 arrives at time 0 and is allocated the CPU until it finishes execution. While P2 is executing, processes P0, P1, P3 arrive into the ready queue. But all other processes have to wait until process P2 finishes its execution. After P2 finishes its CPU cycle, based on the arrival time, process P3 is allocated the CPU. After P3 finishes, P1 executes and then P0.

Q: Explain about Dispatcher?

Another component involved in CPU scheduling is dispatcher. The dispatcher is the module that gives control to the CPU to the process selected by short term scheduler. The functions performed by dispatcher are as follows:

1. Switching context
2. Switching to user mode
3. Jumping to the proper location in the user program to restart that program.

The dispatcher should be as fast as possible given that it is involved during every process switch. The time it takes for dispatcher to stop one process and start another running is known as Dispatch Latency.

Q: Explain about scheduling criteria?

Different CPU scheduling algorithms has different properties. Selection decision depends on the properties of various algorithms. There are several different criteria to consider when trying to select the "best" scheduling algorithm for a particular situation and environment, including:

The criteria are as follows

1. **CPU Utilization:**

   We want to keep the CPU as busy as possible. It may range from 0 to 100%. In real time system, it suits range from 40% (for a lightly loadedsystem) to 90% (for heavily loaded system).

2. **Throughput:**

   If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes completed per time unit for throughput. For time processes, these may be a one process for one minute. For shorter transactions, throughput might be 10 processes per minute.

3. **Turnaround Time:**

    From the submission time of a process to its completion time is ***turnaround time***. It is the sum of total time period spends waiting to get into memory, waiting in the ready queue, executing in the CPU and doing input/output operations(Wall clock time).

4. **Waiting Time:**

    The CPU scheduling algorithm does not affect the amount of the time during which process execute or does input/output. It affects only the amount of time that a process spends waiting in the ready queue. ***Waiting time*** is the sum of period spends waiting in the ready queue.

5. **Response Time**

    In an interactive system, a process can produce some output early and can continue, computing new results. Previous results are being displayed the user. Thus another measure is the time from the submission to the request until the first response is produced. It is called ***Response time***. The turnaround time is normally limited by speed of output device.

In general one wants to optimize the average value of a criteria (Maximize CPU utilization and throughput, and minimize all the others.).

Q: EXPLAIN DIFFERENT TYPES OF SCHEDULING ALGORITHMS?

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.

There are six popular process scheduling algorithms

1. First-Come, First-Served (FCFS) Scheduling
2. Shortest-Job-Next (SJN) Scheduling
3. Priority Scheduling
4. Shortest Remaining Time
5. Round Robin(RR) Scheduling
6. Multiple-Level Queues Scheduling

These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

1. **First Come First Serve (FCFS)**

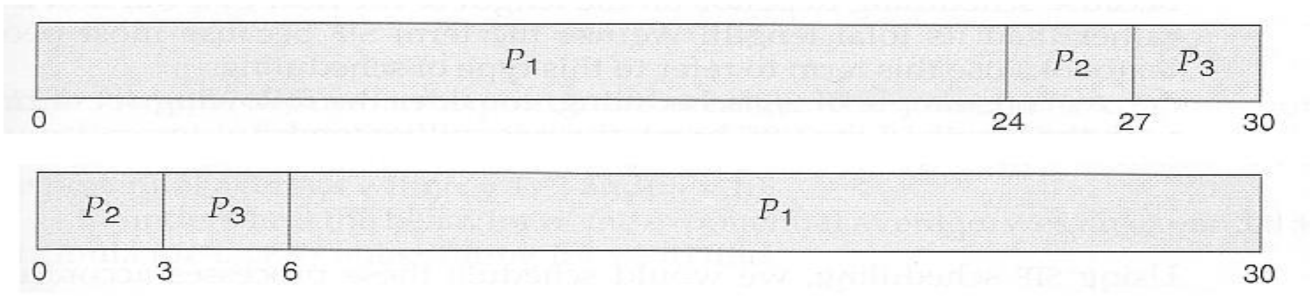
Jobs are executed on first come, first serve basis. It is a non-preemptive, pre-emptive scheduling algorithm. Its implementation is based on FIFO queue.

FCFS is very simple - Just a FIFO queue, like customers waiting in line at the bank or the post office or at a copying machine. Unfortunately, however, FCFS can yield some very long average wait times, particularly if the first process to get there takes a long time.

For example, consider the following three processes:

Process	Burst Time
P1	24
P2	3
P3	3

Gantt Chart:



In the first Gantt chart below, process P1 arrives first. The average waiting time for the three processes is (0 + 24 + 27) / 3 = 17.0 ms.

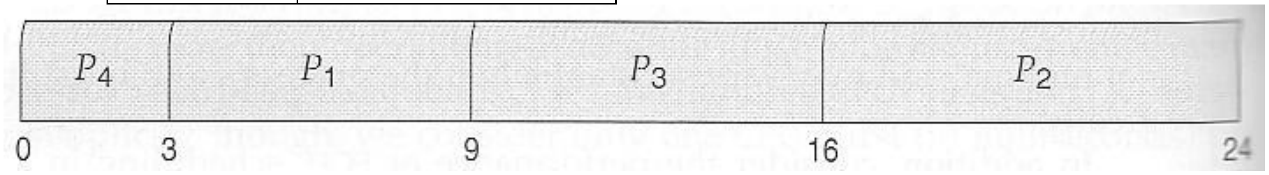
In the second Gantt chart below, the same three processes have an average wait time of (0 + 3 + 6) / 3 = 3.0 ms. The total run time for the three bursts is the same, but in the second case two of the three finish much quicker, and the other process is only delayed by a short amount.

2. **Shortest-Job-First Scheduling, SJF**

This is also known as **shortest job Next**, or SJN. This is a non-preemptive, pre-emptive scheduling algorithm. Best approach to minimize waiting time. The idea behind the SJF algorithm is to pick the quickest fastest little job that needs to be done, get it out of the way first, and then pick the next smallest fastest job to do next. Technically this algorithm picks a process based on the next shortest **CPU burst**, not the overall process time.

For example, the Gantt chart below is based upon the following CPU burst times, (and the assumption that all jobs arrive at the same time.)

Process	Burst Time
P1	6
P2	8
P3	7
P4	3



In the case above the average wait time is (0 + 3 + 9 + 16) / 4 = 7.0 ms, (as opposed to 10.25 ms for FCFS for the same processes.)

SJF can be proven to be the fastest scheduling algorithm,

3. Priority Based Scheduling

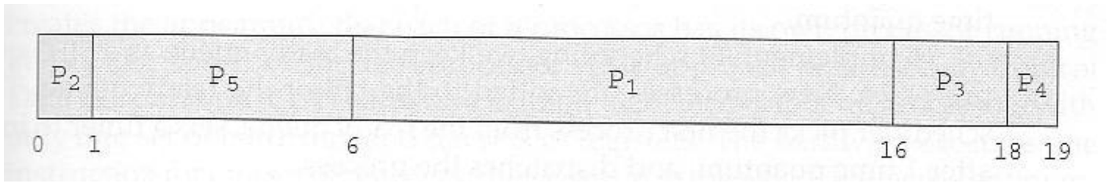
Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with highest priority is to be executed first and so on. Processes with same priority are executed on first come first served basis.

Priorities are implemented using integers within a fixed range, but there is no agreed-upon convention as to whether "high" priorities use large numbers or small numbers. Here we use low number for high priorities, with 0 being the highest possible priority.

Priority can be decided based on memory requirements, time requirements or any other resource requirement.

For example, the following Gantt chart is based upon these process burst times and priorities, and yields an average waiting time of 8.2 ms:

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2



Priorities can be assigned either internally or externally. Internal priorities are assigned by the OS using criteria such as average burst time, ratio of CPU to I/O activity, system resource use, and other factors available to the kernel. External priorities are assigned by users, based on the importance of the job, fees paid, politics, etc.

Priority scheduling can suffer from a major problem known as **indefinite blocking**, or **starvation**, in which a low-priority task can wait forever because there are always some other jobs around that have higher priority.

4. **Round Robin Scheduling:**

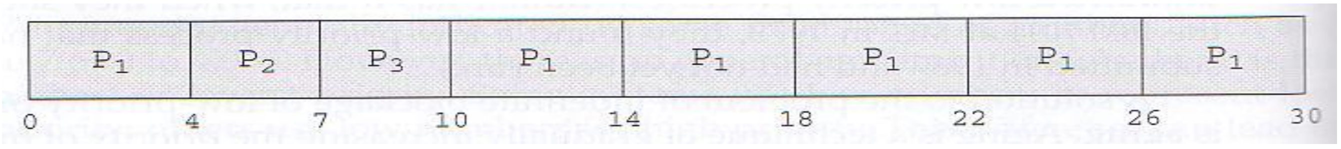
Round Robin is the preemptive process scheduling algorithm. Each process is provided a fix time to execute, it is called a **quantum**. Once a process is executed for a given time period, it is preempted and other process executes for a given time period. Context switching is used to save states of preempted processes.

When a process is given the CPU, a timer is set for whatever value has been set for a time quantum. If the process finishes its burst before the time quantum timer expires, then it is swapped out of the CPU just like the normal FCFS algorithm. If the timer goes off first, then the process is swapped out of the CPU and moved to the back end of the ready queue. The ready queue is maintained as a circular queue, so when all processes have had a turn, then the scheduler gives the first process another turn, and so on.

RR scheduling can give the effect of all processors sharing the CPU equally, although the average wait time can be longer than with other scheduling algorithms.

In the following example the average wait time is 5.66 ms.

Process	**Burst Time**
P1	24
P2	3
P3	3



The performance of RR is sensitive to the time quantum selected. If the quantum is large enough, then RR reduces to the FCFS algorithm; If it is very small, then each process gets 1/nth of the processor time and share the CPU equally. Most modern systems use time quantum between 10 and 100 milliseconds, and context switch times on the order of 10 microseconds.

UNIT III

Deadlock, Deadlock Characterization, Necessary andSufficient Conditions forDeadlock, Deadlock Handling Approaches: Deadlock Prevention, Deadlock Avoidance and Deadlock Detection and Recovery.

Process Management:

Concurrent and Dependent Processes, Critical Section, Semaphores, Methods for Interprocess Communication; Process Synchronization, Classical Process Synchronization Problems: Producer-Consumer, Reader-Writer.

Q: Explain about DEADLOCK?

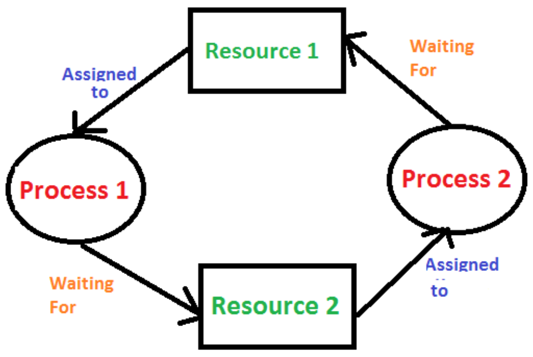
In a multiprogramming environment, several processes may compete for a finite number of resources. A process requests resources; if the resources are not available at that time, the process enters a waiting state. Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting processes. This situation is called a deadlock. we will see that deadlocks can occur with many other types of resources available in a computer system.

A deadlock is a situation in which two processes sharing the same set of resources are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function. Deadlocks situation occurs when two or more processes attempt to access a resource, which is locked by another process and therefore, cannot be shared. Due to this locking of resource, each process has to wait for resource that is locked by another process and as a result none of the transaction can finish.

Thus, in deadlock, processes never finish executing and system resources are tied up, preventing other jobs from starting.

A deadlock can be defined formally as

"***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”.



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.

Basically in the Normal mode of Operation utilization of resources by a process is in the following sequence:

1. **Request:** Firstly, the process requests the resource. In a case, if the request cannot be granted immediately(e.g: resource is being used by any other process), then the requesting process must wait until it can acquire the resource.
2. **Use:** The Process can operate on the resource (e.g: if the resource is a printer then in that case process can print on the printer).
3. **Release:** The Process releases the resource

Q: Explain about DEADLOCK CHARACTERIZATION?

In a deadlock, processes never finish executing, and system resources are tied up, preventing other jobs from starting. Before we discuss the various methods for dealing with the deadlock problem, we look more closely at features that characterize deadlocks.

1 Necessary and sufficient Conditions for deadlock

There are four conditions that are necessary to achieve deadlock:

a) **Mutual Exclusion -** At least one resource must be held in a non-sharable mode; If any other process requests this resource, then that process must wait for the resource to be released.

b) **Hold and Wait -** A process must be simultaneously holding at least one resource and waiting for at least one resource that is currently being held by some other process.

c) **No preemption -** Once a process is holding a resource (i.e. once its request has been granted), then that resource cannot be taken away from that process until the process voluntarily releases it.

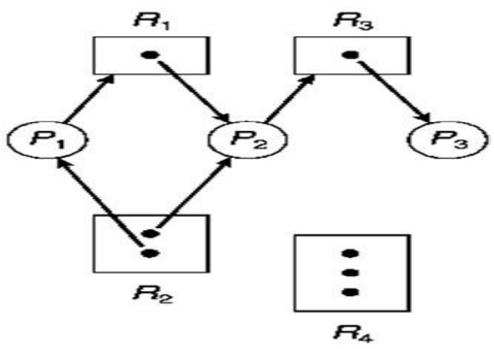
d) **Circular Wait -** A set of processes { P0, P1, P2, . . ., PN } must exist such that every P[i] is waiting for P[(i + 1) % (N + 1)]. (Note that this condition implies the hold-and-wait condition, but it is easier to deal with the conditions if the four are considered separately.)

2) 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. (However note also that request edges point to the category box, whereas assignment edges emanate from a particular instance dot within the box.)

For example:



Resource allocation graph

- If a resource-allocation graph contains no cycles, then the system is not deadlocked. (When looking for cycles, remember that these are directed graphs.)
- 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. Consider, for example, Figures 7.3 and 7.4 below:

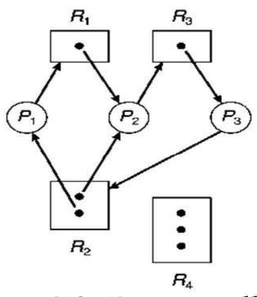
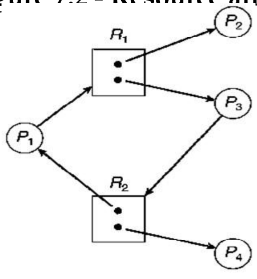


Figure 7.2 - Resource allocation



Resource allocation graph with a cycle but no deadlock

Q: Explain about DEADLOCK HANDLING APPROACHES?

Generally speaking, there are three ways of handling deadlocks:

Deadlock prevention or avoidance - Do not allow the system to get into a deadlocked state.

Deadlock detection and recovery - Abort a process or preempt some resources when deadlocks are detected.

Ignore the problem all together - If deadlocks only occur once a year or so, it may be better to simply let them happen and reboot as necessary than to incur the constant overhead and system performance penalties associated with deadlock prevention or detection. This is the approach that both Windows and UNIX take.

Q: Explain about DEADLOCK PREVENTION?

Deadlocks can be prevented by preventing at least one of the four required conditions:

1)Mutual Exclusion

- Shared resources such as read-only files do not lead to deadlocks.
- Unfortunately some resources, such as printers and tape drives, require exclusive access by a single process.

2) Hold and Wait

To prevent this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others. **There are several possibilities for this:**

a) Require that all processes request all resources at one time. This can be wasteful of system resources if a process needs one resource early in its execution and doesn't need some other resource until much later.

b) Require that processes holding resources must release them before requesting new resources, and then re-acquire the released resources along with the new ones in a single new request. This can be a problem if a process has partially completed an operation using a resource and then fails to get it re-allocated after releasing it.

Either of the methods described above can lead to starvation if a process requires one or more popular resources.

3) No Preemption

- Preemption of process resource allocations can prevent this condition of deadlocks, when it is possible.
- **One approach** is that if a process is forced to wait when requesting a new resource, then all other resources previously held by this process are implicitly released, (preempted), forcing this

process to re-acquire the old resources along with the new resources in a single request, similar to the previous discussion.

- **Another approach** is that when a resource is requested and not available, then the system looks to see what other processes currently have those resources and are themselves blocked waiting for some other resource. If such a process is found, then some of their resources may get preempted and added to the list of resources for which the process is waiting.
- Either of these approaches may be applicable for resources whose states are easily saved and restored, such as registers and memory, but are generally not applicable to other devices such as printers and tape drives.

4 Circular Wait

- One way to avoid circular wait is to number all resources, and to require that processes request resources only in strictly increasing (or decreasing) order.
- In other words, in order to request resource Rj,a process must first release all Ri such that i>= j.
- One big challenge in this scheme is determining the relative ordering of the different resources

Q: Explain about DEADLOCK AVOIDANCE?

The general idea behind deadlock avoidance is to prevent deadlocks from ever happening, by preventing at least one of the aforementioned conditions.

- This requires more information about each process, AND tends to lead to low device utilization. (I.e. it is a conservative approach.)
- When a scheduler sees that starting a process or granting resource requests may lead to future deadlocks, then that process is just not started or the request is not granted.
- A resource allocation state is defined by the number of available and allocated resources, and the maximum requirements of all processes in the system.

Safe and Unsafe States

The resource allocation state of a system can be defined by the instances of available and allocated resources, and the maximum instance of the resources demanded by the processes.

A state of a system recorded at some random time is shown below.

Resources Assigned

Process	Type 1	Type 2	Type 3	Type 4

A	3	0	2	2
B	0	0	1	1
C	1	1	1	0
D	2	1	4	0

Resources still needed

Process	Type 1	Type 2	Type 3	Type 4
A	1	1	0	0
B	0	1	1	2
C	1	2	1	0
D	2	1	1	2

E = (7 6 8 4)

P = (6 2 8 3)

A = (1 4 0 1)

Above tables and vector E, P and A describes the resource allocation state of a system.

There are 4 processes and 4 types of the resources in a system.

Table 1 shows the instances of each resource assigned to each process.

Table 2 shows the instances of the resources, each process still needs.

Vector E is the representation of total instances of each resource in the system.

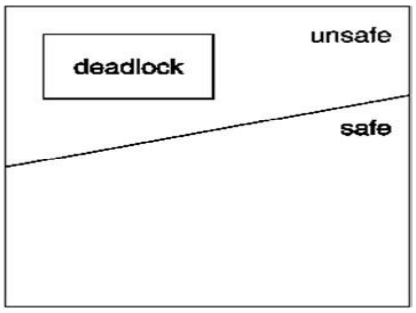
Vector P represents the instances of resources that have been assigned to processes.

Vector A represents the number of resources that are not in use.

A state of the system is called **safe** if the system can allocate all the resources requested by all the processes without entering into deadlock.

If the system cannot fulfill the request of all processes then the state of the system is called **unsafe**.

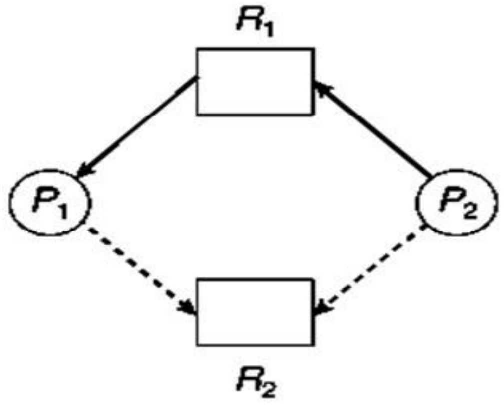
The key of Deadlock avoidance approach is when the request is made for resources then the request must only be approved in the case if the resulting state is also a safe state.



Safe, unsafe, and deadlocked state spaces.

2 Resource-Allocation Graph Algorithm

- If resource categories have only single instances of their resources, then deadlock states can be detected by cycles in the resource-allocation graphs.
- In this case, unsafe states can be recognized and avoided by augmenting the resource-allocation graph with claim edges, noted by dashed lines, which point from a process to a resource that it may request in the future.
- In order for this technique to work, all claim edges must be added to the graph for any particular process before that process is allowed to request any resources. (Alternatively, processes may only make requests for resources for which they have already established claim edges, and claim edges cannot be added to any process that is currently holding resources.)
- When a process makes a request, the claim edge Pi->Rj is converted to a request edge. Similarly when a resource is released, the assignment reverts back to a claim edge.
- This approach works by denying requests that would produce cycles in the resource-allocation graph, taking claim edges into effect.
- Consider for example what happens when process P2 requests resource R2:



Resource allocation graph for deadlock avoidance

• The resulting resource-allocation graph would have a cycle in it, and so the request cannot be granted.

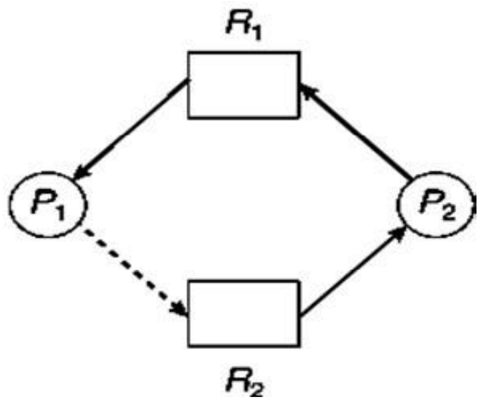


Figure 7.8 - An unsafe state in a resource allocation graph

Deadlock Avoidance

Deadlock avoidance can be done with Banker's Algorithm.

Banker's 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.

Inputs to Banker's Algorithm:

1. Max need of resources by each process.
2. Currently allocated resources by each process.
3. Max free available resources in the system.

The request will only be granted under the below condition:

1. If the request made by the process is less than equal to max need to that process.
2. If the request made by the process is less than equal to the freely available resource in the system.

Example:

```
Total resources in system:
A B C D
6 5 7 6
Available system resources are:
A B C D
3 1 1 2
Processes (currently allocated resources):
   A B C D
P1  1 2 2 1
P2  1 0 3 3
```

P3 1 2 1 0

Processes (maximum resources):

	A	B	C	D
P1	3	3	2	2
P2	1	2	3	4
P3	1	3	5	0

Need = maximum resources - currently allocated resources.

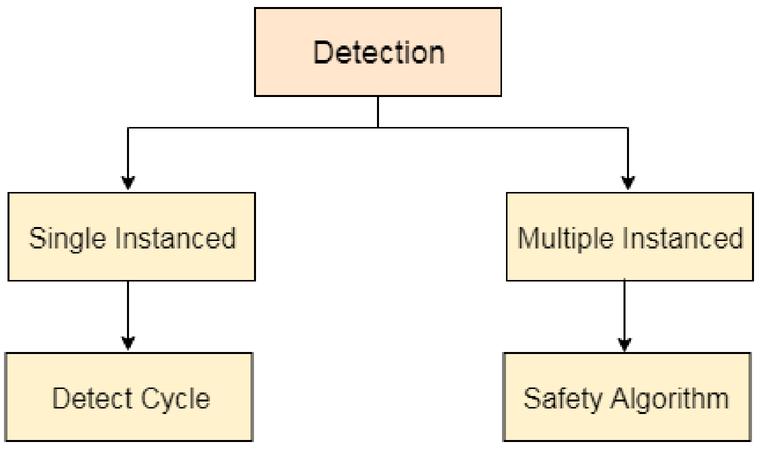
Processes (need resources):

	A	B	C	D
P1	2	1	0	1
P2	0	2	0	1
P3	0	1	4	0

Q: Explain about Deadlock Detection and Recovery?

In this approach, The OS doesn't apply any mechanism to avoid or prevent the deadlocks. Therefore the system considers that the deadlock will definitely occur. In order to get rid of deadlocks, The OS periodically checks the system for any deadlock. In case, it finds any of the deadlock then the OS will recover the system using some recovery techniques.

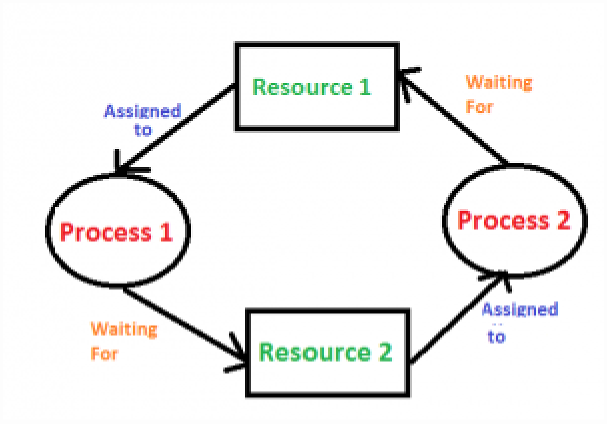
The main task of the OS is detecting the deadlocks. The OS can detect the deadlocks with the help of Resource allocation graph.



Deadlock Detection

1. **If resources have single instance:**

   In this case for Deadlock detection we can run an algorithm to check for cycle in the Resource Allocation Graph. Presence of cycle in the graph is the sufficient condition for deadlock.



   In the above diagram, resource 1 and resource 2 have single instances.

   There is a cycle R1 → P1 → R2 → P2. So, Deadlock is Confirmed.

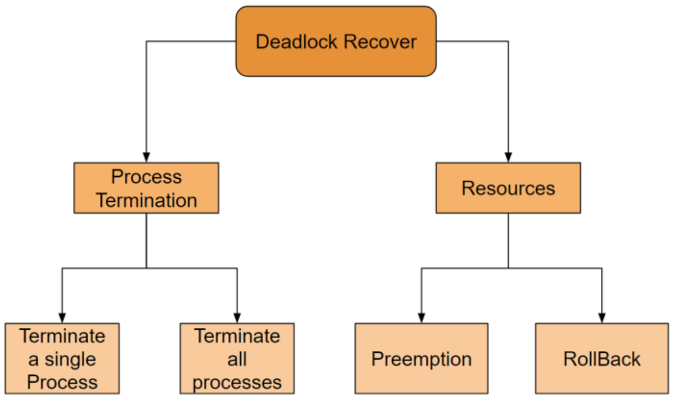
2. **If there are multiple instances of resources:**

   in multiple instanced resource type graph, detecting a cycle is not just enough. We have to apply the safety algorithm on the system by converting the resource allocation graph into the allocation matrix and request matrix. The system may or may not be in deadlock varies according to different situations.

Q: Explain about Deadlock Recovery

In order to recover the system from deadlocks, either OS considers resources or processes.

Methods involved in the recovery of the Deadlock:



Process termination

Kill a Process

- In this process, the process responsible for the Deadlock is terminated. However, selecting which process to kill can be difficult. As a result, the operating system primarily kills the process that no longer works.

Kill all Processes

- It is not appropriate to kill all the processes. However, when a critical situation arises, this approach may be suitable.
- Putting an end to all processes reduces the system's efficiency, so we must start all the processes again.

Resource Preemption

Preemption

- Here, we transfer resources from one process to the process that needs them to finish its execution, and once that process is done, the resource is released.
- The selection of resources is complex, and grabbing the resources is also challenging. Several states are necessary for the system to reach the Deadlock.

RollBack

- A rollback of the system to the earlier safe state is possible using the operating system. This requires the implementation of checkpoints in every state.
- When we identify Deadlock, we must roll back all allocations and return to the previous safe state.

Q: What is Process Synchronization?

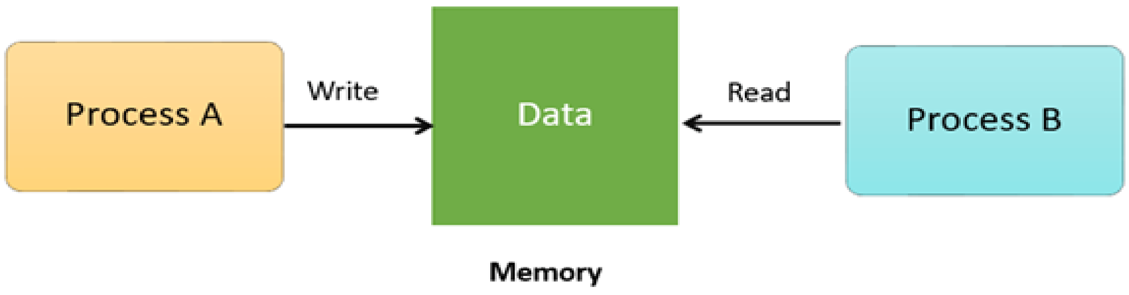
Process Synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.

- It is specially needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or data at the same time.
- This can lead to the inconsistency of shared data. So the change made by one process not necessarily reflected when other processes accessed the same shared data. To avoid this type of inconsistency of data, the processes need to be synchronized with each other.

How Process Synchronization Works?

Process Synchronization was introduced to handle problems that arise while multiple process executions.

For Example, process A changing the data in a memory location while another process B is trying to read the data from the **same** memory location. There is a high probability that data read by the second process will be erroneous.



Q:Explain about process types?

Process is categorized into two types on the basis of synchronization and these are given below:

- Independent Process
- Cooperative Process

Independent Processes

Two processes are said to be independent if the execution of one process does not affect the execution of another process.

Cooperative Processes

Two processes are said to be cooperative if the execution of one process affects the execution of another process. These processes need to be synchronized so that the order of execution can be guaranteed.

There are several reasons why cooperating processes are allowed:

- Information Sharing - There may be several processes which need access to the same file for example. (e.g. pipelines.)
- Computation speedup - Often a solution to a problem can be solved faster if the problem can be broken down into sub-tasks to be solved simultaneously (particularly when multiple processors are involved.)
- Modularity - The most efficient architecture may be to break a system down into cooperating modules. (E.g. databases with a client-server architecture.)
- Convenience - Even a single user may be multi-tasking, such as editing, compiling, printing, and running the same code in different windows.

Q: Explain different Sections of a Program?

There are four essential elements of the critical section:

- **Entry Section:** It is part of the process which decides the entry of a particular process.
- **Critical Section:** This part allows one process to enter and modify the shared variable.
- **Exit Section:** Exit section allows the other process that are waiting in the Entry Section, to enter into the Critical Sections. It also checks that a process that finished its execution should be removed through this Section.
- **Remainder Section:** All other parts of the Code, which is not in Critical, Entry, and Exit Section, are known as the Remainder Section.

```
do {
    entry section
        critical section
    exit section
        remainder section
} while (TRUE);
```

Q: Explain about race condition?

Race Condition

At the time when more than one process is either executing the same code or accessing the same memory or any shared variable; In that condition, there is a possibility that the output or the value of the shared variable is wrong so for that purpose all the processes are doing the race to say that my output is correct. This condition is commonly known as **a race condition.**

Q: What is Critical Section Problem?

- If multiple processes access the critical section concurrently, then results produced might be inconsistent.
- This problem is called as **critical section problem**.

The section consists of shared data resources that required to be accessed by other processes.

- The entry to the critical section is handled by the **wait()** function, and it is represented **as P().**
- The exit from a critical section is controlled by the **signal()** function, represented as **V().**

In the critical section, only a single process can be executed. Other processes, waiting to execute their critical section, need to wait until the current process completes its execution.

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.

Methods to solve the critical section problem

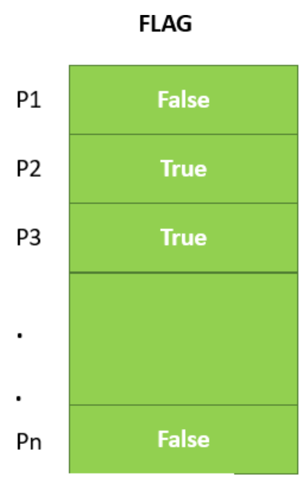
1. **Peterson Solution**
2. **Synchronization Hardware**
3. **Mutex Locks**
4. **Semaphore Solution**

1) 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.

Example



```
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
```

- Assume there are N processes (P1, P2, ... PN) and every process at some point of time requires to enter the Critical Section
- A FLAG[] array of size N is maintained which is by default false. So, whenever a process requires to enter the critical section, it has to set its flag as true. For example, If Pi wants to enter it will set FLAG[i]=TRUE.
- Another variable called TURN indicates the process number which is currently wating to enter into the CS.
- The process which enters into the critical section while exiting would change the TURN to another number from the list of ready processes.
- Example: turn is 2 then P2 enters the Critical section and while exiting turn=3 and therefore P3 breaks out of wait loop.

2) Synchronization Hardware

Some times the problems of the Critical Section are also resolved by hardware. Some operating system offers a lock functionality where a Process acquires a lock when entering the Critical section and releases the lock after leaving it.

So when another process is trying to enter the critical section, it will not be able to enter as it is locked. It can only do so if it is free by acquiring the lock itself.

3) 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.

4) Semaphore Solution

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

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

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

Q:Explain about Semaphores?

In 1965, Dijkstra proposed a new and very significant technique for managing concurrent processes by using the value of a simple integer variable to synchronize the progress of interacting processes. This integer variable is called a **semaphore**. So it is basically a synchronizing tool and is accessed only through two low standard atomic operations, **wait** and **signal** designated by P(S) and V(S) respectively.

```
P(S): if S >= 1 then S := S - 1
   else <block and enqueue the process>;

V(S): if <some process is blocked on the queue>
    then <unblock a process>
   else S := S + 1;
```

.

Properties of Semaphores

1. It's simple and always have a non-negative integer value.
2. Works with many processes.
3. Can have many different critical sections with different semaphores.
4. Each critical section has unique access semaphores.
5. Can permit multiple processes into the critical section at once, if desirable.

Types of Semaphores

Semaphores are mainly of two types in Operating system:

Binary Semaphore:

- It is a special form of semaphore used for implementing mutual exclusion, hence it is often called a **Mutex**. A binary semaphore is initialized to 1 and only takes the values 0 and 1 during the execution of a program.
- In Binary Semaphore, the wait operation works only if the value of semaphore = 1, and the signal operation succeeds when the semaphore= 0.

Counting Semaphores:

- These are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access,
- The value of counting semaphore may be positive or negative.
- Positive value indicates the number of processes that can be present in the critical section at the same time.
- Negative value indicates the number of processes that are blocked in the waiting list.
- where the semaphore count is the number of available resources
- . If the resources are added, semaphore count automatically incremented and if the resources are removed, the count is decremented.

Benefits of using Semaphores are as given below:

- With the help of semaphores, there is a flexible management of resources.
- Semaphores are machine-independent and they should be run in the machine-independent code of the microkernel.
- Semaphores do not allow multiple processes to enter in the critical section.
- They allow more than one thread to access the critical section.
- As semaphores follow the mutual exclusion principle strictly and these are much more efficient than some other methods of synchronization.

Disadvantages of Semaphores

- One of the biggest limitations is that semaphores may lead to priority inversion; where low priority processes may access the critical section first and high priority processes may access the critical section later.
- To avoid deadlocks in the semaphore, the Wait and Signal operations are required to be executed in the correct order.
- Their use is not enforced but is by convention only.

- With improper use, a process may block indefinitely. Such a situation is called **Deadlock**.

Q:Explain about classical problems of synchronization?

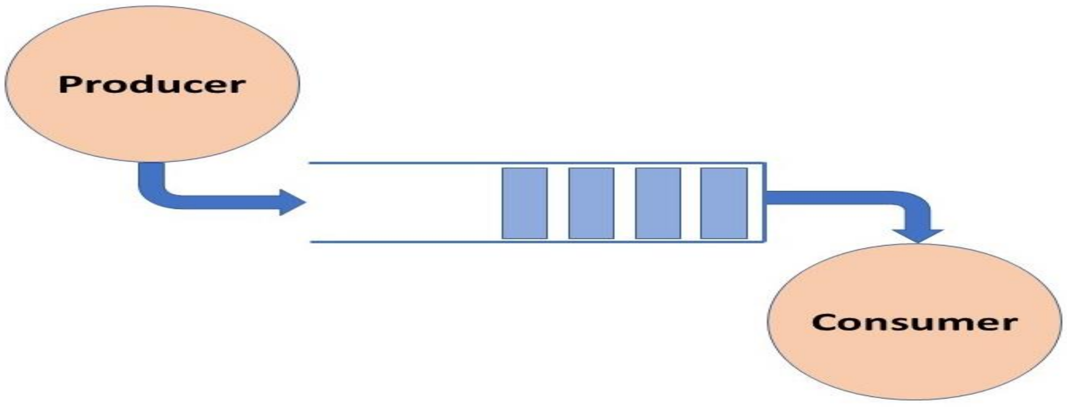
The classical problems of synchronization are as follows:

1. Bound-Buffer problem
2. Dining Philosophers problem
3. Readers and writers problem

What is Bound-Buffer problem?

Also known as the **Producer-Consumer problem**. In this problem, there is a buffer of n slots, and each buffer is capable of storing one unit of data. There are two processes that are operating on the buffer – Producer and Consumer. The producer tries to insert data and the consumer tries to remove data.

If the processes are run simultaneously they will not yield the expected output.



The solution to this problem is creating two semaphores, one full and the other empty to keep a track of the concurrent processes.

Q:What is Readers Writer Problem?

Readers writer problem is another example of a classic synchronization problem.

The Problem Statement

There is a shared resource which should be accessed by multiple processes. There are two types of processes in this context. They are **reader** and **writer**. Any number of **readers** can read from the shared resource simultaneously, but only one **writer** can write to the shared resource. When a **writer** is writing data to the resource, no other process can access the resource. A **writer** cannot write to the resource if there are non zero number of readers accessing the resource at that time.

The Solution

From the above problem statement, it is evident that readers have higher priority than writer. If a writer wants to write to the resource, it must wait until there are no readers currently accessing that resource.

Here, we use one **mutex** m and a **semaphore** w. An integer variable read_count is used to maintain the number of readers currently accessing the resource. The variable read_count is initialized to 0. A value of 1 is given initially to m and w.

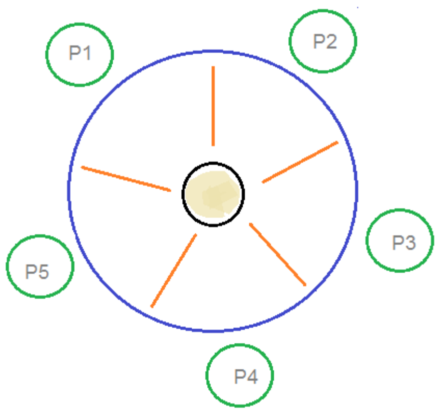
Instead of having the process to acquire lock on the shared resource, we use the mutex m to make the process to acquire and release lock whenever it is updating the read_count variable.

Q: What is Dining Philosophers Problem?

The dining philosophers problem is another classic synchronization problem which is used to evaluate situations where there is a need of allocating multiple resources to multiple processes.

Problem Statement

Consider there are five philosophers sitting around a circular dining table. The dining table has five chopsticks and a bowl of rice in the middle as shown in the below figure.



Dining Philosophers Problem

At any instant, a philosopher is either eating or thinking. When a philosopher wants to eat, he uses two chopsticks - one from their left and one from their right. When a philosopher wants to think, he keeps down both chopsticks at their original place.

Here's the Solution

From the problem statement, it is clear that a philosopher can think for an indefinite amount of time. But when a philosopher starts eating, he has to stop at some point of time. The philosopher is in an endless cycle of thinking and eating.

An array of five semaphores, stick[5], for each of the five chopsticks

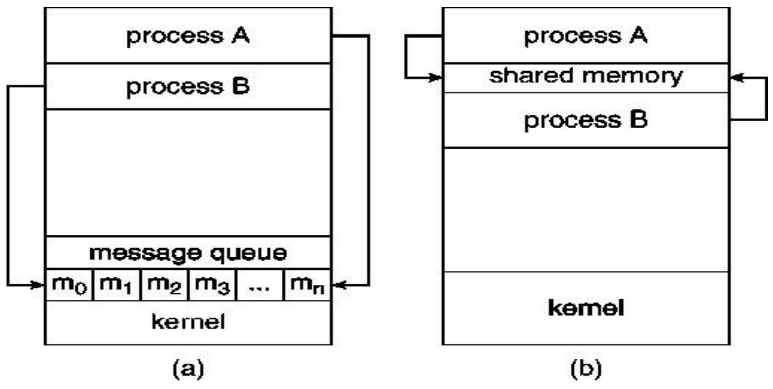
Q:What is Inter Process Communication?

In general, Inter Process Communication is a type of mechanism usually provided by the operating system (or OS). The main aim or goal of this mechanism is to provide communications in between several processes. the intercommunication allows a process letting another process know that some event has occurred.

Why we need interprocess communication?

There are numerous reasons to use inter-process communication for sharing the data. Here are some of the most important reasons that are given below:

- It helps to speedup modularity
- Computational
- Privilege separation
- Convenience
- Helps operating system to communicate with each other and synchronize their actions as well.



Communications models: (a) Message passing. (b) Shared memory.

Role of Synchronization in Inter Process Communication

It is one of the essential parts of inter process communication. Typically, this is provided by interprocess communication control mechanisms, but sometimes it can also be controlled by communication processes.

These are the following methods that used to provide the synchronization:

1. **Mutual Exclusion**
2. **Semaphore**
3. **Barrier**
4. **Spinlock**

Mutual Exclusion:-

It is generally required that only one process thread can enter the critical section at a time. This also helps in synchronization and creates a stable state to avoid the race condition.

Semaphore:-

Semaphore is a type of variable that usually controls the access to the shared resources by several processes. Semaphore is further divided into two types which are as follows:

1. Binary Semaphore
2. Counting Semaphore

Barrier:-

A barrier typically not allows an individual process to proceed unless all the processes does not reach it. It is used by many parallel languages, and collective routines impose barriers.

Spinlock:-

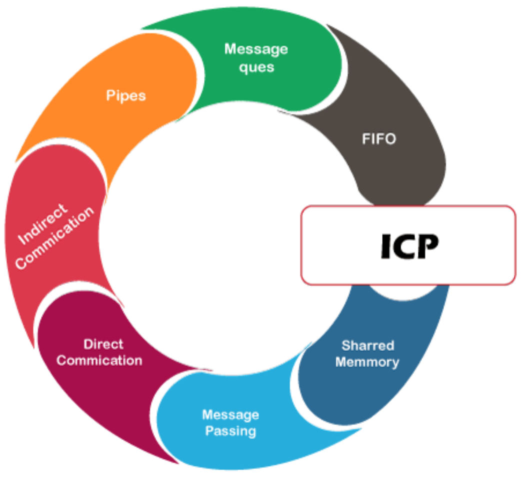
Spinlock is a type of lock as its name implies. The processes are trying to acquire the spinlock waits or stays in a loop while checking that the lock is available or not. It is known as busy waiting because even though the process active, the process does not perform any functional operation (or task).

Approaches to Interprocess Communication

We will now discuss some different approaches to inter-process communication which are as follows:

These are a few different approaches for Inter- Process Communication:

1. **Pipes**
2. **Shared Memory**
3. **Message Queue**
4. **Direct Communication**
5. **Indirect communication**
6. **Message Passing**
7. **FIFO**



Pipe:-

The pipe is a type of data channel that is unidirectional in nature. It means that the data in this type of data channel can be moved in only a single direction at a time. Still, one can use two-channel of this type, so that he can able to send and receive data in two processes. Typically, it uses the standard methods for input and output. These pipes are used in all types of POSIX systems and in different versions of window operating systems as well.

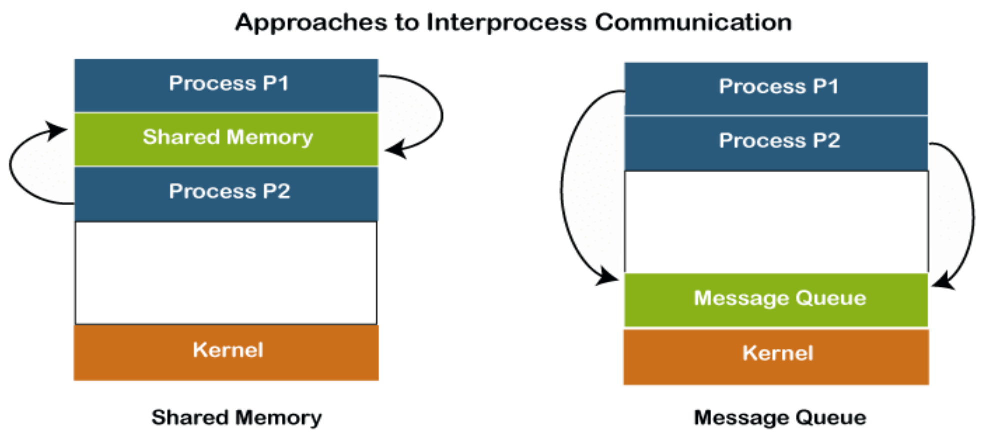
Shared Memory:-

It can be referred to as a type of memory that can be used or accessed by multiple processes simultaneously. It is primarily used so that the processes can communicate with each other. Therefore the shared memory is used by almost all POSIX and Windows operating systems as well.

Message Queue:-

In general, several different messages are allowed to read and write the data to the message queue. In the message queue, the messages are stored or stay in the queue unless their recipients retrieve them. In short, we can also say that the message queue is very helpful in inter-process communication and used by all operating systems.

To understand the concept of Message queue and Shared memory in more detail, let's take a look at its diagram given below:



Message Passing:-

It is a type of mechanism that allows processes to synchronize and communicate with each other. However, by using the message passing, the processes can communicate with each other without restoring the hared variables.

Usually, the inter-process communication mechanism provides two operations that are as follows:

- send (message)
- received (message)

Direct Communication:-

In this type of communication process, usually, a link is created or established between two communicating processes. However, in every pair of communicating processes, only one link can exist.

UNIT- 4

Memory Management: Physical and Virtual Address Space; Memory Allocation Strategies– Fixed and -Variable Partitions, Paging, Segmentation, Virtual Memory

Q:Explain about Memory Management?

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution.

Memory management keeps track of

- Each and every memory location of process
- It checks how much memory is to be allocated to processes.
- It decides which process will get memory at what time.
- It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

Process Address Space:

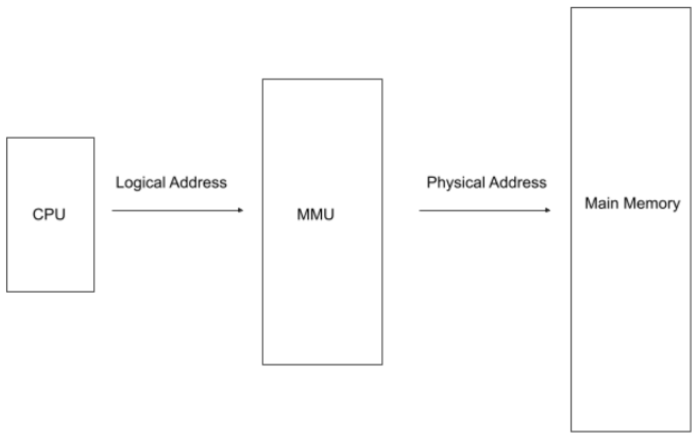
- We store the data in the memory at different locations with addresses to access the data again whenever required in the future.
- Address is used to uniquely identify the location of something inside the CPU memory.
- The addresses identify a location in the memory where the actual code resides in the system in the operating system.
- There are two types of addresses used for memory in the operating system, i.e., the ***physical address*** and ***logical address***

Logical Address

- It is a virtual address generated by the CPU while a program is running.
- It is referred to as a virtual address because it does not exist physically.
- Using this address, the CPU access the actual address or physical address inside the memory, and data is fetched from there.
- The hardware device called Memory Management Unit (MMU) is used for mapping this logical address to the physical address.
- The set of all logical addresses generated by the CPU for a program is called the logical address space.

Physical Address

- Physical Address is the actual address of the data inside the memory.
- The logical address is a virtual address and the program needs physical memory for its execution.
- The user never deals with the Physical Address. The user program generates the logical address and is mapped to the physical address by the Memory Management Unit(MMU).
- The set of all physical addresses corresponding to the logical addresses in the logical address space is called the physical address space.



Differences between Logical Address and Physical Address :

logical Address	Physical Address
It is a virtual address.	It is the actual location in the memory.
It is visible to the user.	It is not visible to the user.
It is generated by the CPU.	It is computed by the MMU.
It is used by the user to access the physical address inside the memory.	It is not accessible directly by the user.
The set of all logical addresses generated by the CPU is called the logical address space.	The set of all physical addresses corresponding to the logical addresses in the logical address space is called the physical address space.

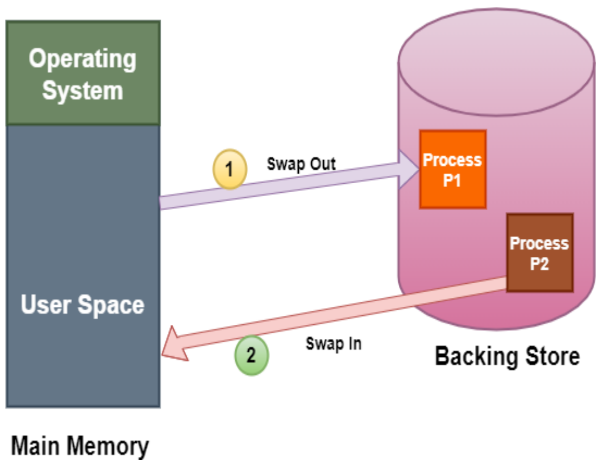
Q: EXPLAIN ABOUT SWAPPING?

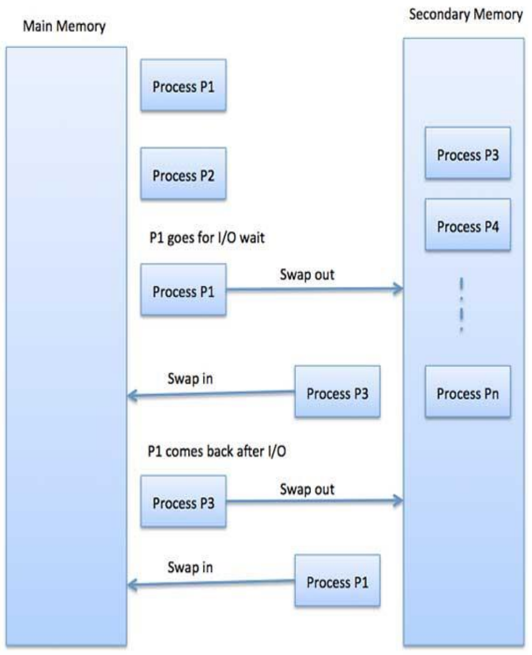
Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.

Suspending a process ensures that it is not runnable while it is swapped out. At some later time, the system swaps back the process from the secondary storage to main memory. When a process is busy swapping pages in and out then this situation is called **thrashing**.

Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction**.

The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.





Swap In and Swap Out in OS

- The procedure by which any process gets removed from the **hard disk** and placed in the **main memory or RAM** commonly known as **Swap In.**
- On the other hand, **Swap Out** is the method of removing a process from the **main memory or RAM** and then adding it to the **Hard Disk.**

Example:

Let us assume that the user process is of size 2048KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take

2048KB / 1024KB per second

= 2 seconds

= 2000 milliseconds

Now considering in and out time, it will take complete 4000 milliseconds plus other overhead where the process competes to regain main memory.

Advantages of Swapping

- The swapping technique mainly helps the CPU to manage multiple processes within a single main memory.
- This technique helps to create and use virtual memory.
- With the help of this technique, the CPU can perform several tasks simultaneously. Thus, processes need not wait too long before their execution.
- This technique is economical.
- This technique can be easily applied to priority-based scheduling in order to improve its performance.

Disadvantages of Swapping

- There may occur inefficiency in the case if a resource or a variable is commonly used by those processes that are participating in the swapping process.
- If the algorithm used for swapping is not good then the overall method can increase the number of page faults and thus decline the overall performance of processing.
- If the computer system loses power at the time of high swapping activity then the user might lose all the information related to the program.

Q: MEMORY ALLOCATION STRATEGIES?

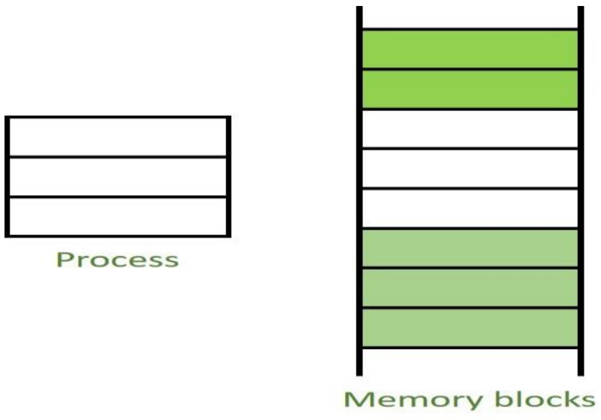
The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods.

1. Contiguous Allocation
2. Non Contiguous Allocation

The main idea behind these methods is to provide: Efficient disk space utilization.

Fast access to the file blocks

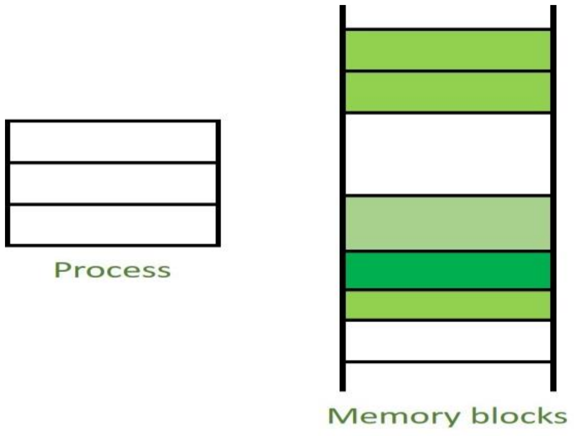
1. Contiguous Memory Allocation : Contiguous memory allocation is basically a method in which a single contiguous section/part of memory is allocated to a process or file needing it. Because of this all the available memory space resides at the same place together, which means that the freely/unused available memory partitions are not distributed in a random fashion here and there across the whole memory space.



Contiguous Memory Allocation

The main memory is a combination of two main portions- one for the operating system and other for the user program. We can implement/achieve contiguous memory allocation by dividing the memory partitions into fixed size partitions.

2. Non-Contiguous Memory Allocation : Non-Contiguous memory allocation is basically a method which allocates the memory space present in different locations to the process as per it's requirements. As all the available memory space is in a distributed pattern so the freely available memory space is also scattered here and there. This technique of memory allocation helps to reduce the wastage of memory, which eventually gives rise to Internal and external fragmentation.



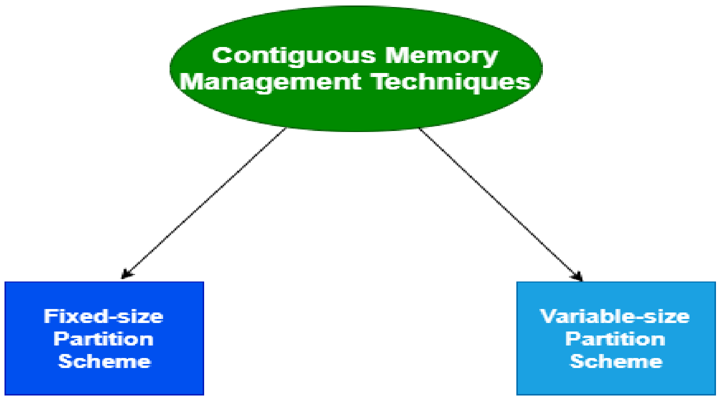
Noncontiguous Memory Allocation

Q: Difference between Contiguous and Non-contiguous Memory Allocation :

S.NO.	Contiguous Memory Allocation	Non-Contiguous Memory Allocation
1.	Contiguous memory allocation allocates consecutive blocks of memory to a file/process.	Non-Contiguous memory allocation allocates separate blocks of memory to a file/process.
2.	Faster in Execution.	Slower in Execution.
3.	It is easier for the OS to control.	It is difficult for the OS to control.
4.	Overhead is minimum as not much address translations are there while executing a process.	More Overheads are there as there are more address translations.
5.	Both Internal fragmentation and external fragmentation occurs in Contiguous memory allocation method.	Only External fragmentation occurs in Non-Contiguous memory allocation method.
6.	It includes single partition allocation and multi-partition allocation.	It includes paging and segmentation.
7.	Wastage of memory is there.	No memory wastage is there.
8.	In contiguous memory allocation, swapped-in processes are arranged in the originally allocated space.	In non-contiguous memory allocation, swapped-in processes can be arranged in any place in the memory.
9.	It is of two types: 1. Fixed(or static) partitioning 2. Dynamic partitioning	It is of five types: 1. Paging 2. Multilevel Paging 3. Inverted Paging 4. Segmentation 5. Segmentated Paging
10.	It could be visualized and implemented using Arrays.	It could be implemented using Linked Lists.

Q: Explain about contiguous memory management strategies?

The memory can be divided either in the **fixed-sized partition or in the variable-sized partition** in order to allocate contiguous space to user processes.

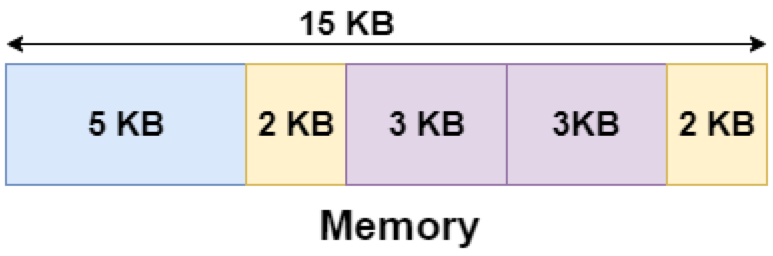

Contiguous Memory Management Techniques
Fixed-size Partition Scheme
Variable-size Partition Scheme

Fixed-size Partition Scheme

- This technique is also known as **Static partitioning**.
- In this scheme, the system divides the memory into fixed-size partitions. The partitions may or may not be the same size.
- The size of each partition is fixed as indicated by the name of the technique and it cannot be changed.
- In this partition scheme, each partition may contain exactly one process.
- There is a problem that this technique will limit the degree of multiprogramming because the number of partitions will basically decide the number of processes.
- Whenever any process terminates then the partition becomes available for another process.

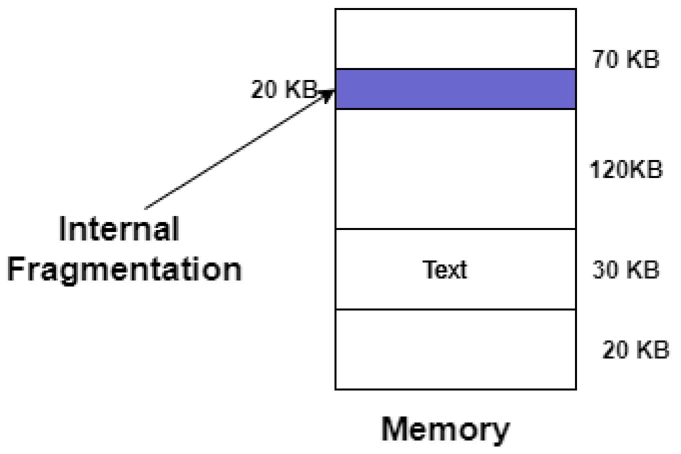
Example

Let's take an example of fixed size partitioning scheme, we will divide a memory size of 15 KB into fixed-size partitions:



It is important to note that these partitions are allocated to the processes as they arrive and the partition that is allocated to the arrived process basically depends on the algorithm followed.

If there is some wastage inside the partition then it is termed **Internal Fragmentation**.



Advantages of Fixed-size Partition Scheme

- This scheme is simple and is easy to implement
- It supports multiprogramming as multiple processes can be stored inside the main memory.
- Management is easy using this scheme

Disadvantages of Fixed-size Partition Scheme

1. Internal Fragmentation

Suppose the size of the process is lesser than the size of the partition in that case some size of the partition gets wasted and remains unused. This wastage inside the memory is generally termed as Internal fragmentation

As we have shown in the above diagram the 70 KB partition is used to load a process of 50 KB so the remaining 20 KB got wasted.

2. Limitation on the size of the process

If in a case size of a process is more than that of a maximum-sized partition then that process cannot be loaded into the memory. Due to this, a condition is imposed on the size of the process and it is: the size of the process cannot be larger than the size of the largest partition.

3. External Fragmentation

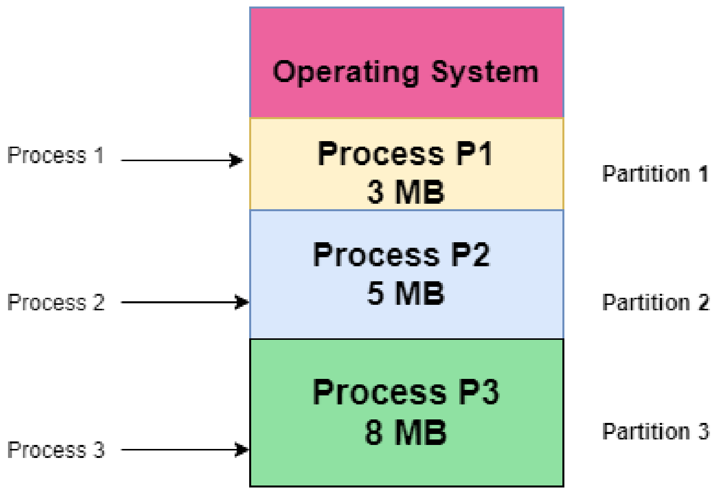
It is another drawback of the fixed-size partition scheme as total unused space by various partitions cannot be used in order to load the processes even though there is the availability of space but it is not in the contiguous fashion.

4. Degree of multiprogramming is less

In this partition scheme, as the size of the partition cannot change according to the size of the process. Thus the degree of multiprogramming is very less and is fixed.

Variable-size Partition Scheme

- This scheme is also known as **Dynamic partitioning** and is came into existence to overcome the internal fragmentation that is caused by **Static partitioning**. In this partitioning, scheme allocation is done dynamically.
- The size of the partition is not declared initially.
- Whenever any process arrives, a partition of size equal to the size of the process is created and then allocated to the process. Thus the size of each partition is equal to the size of the process.
- As partition size varies according to the need of the process so in this partition scheme there is no **internal fragmentation.**



Size of Partition =Size of Process

Advantages of Variable-size Partition Scheme

1. **No Internal Fragmentation** As in this partition scheme space in the main memory is allocated strictly according to the requirement of the process thus there is no chance of internal fragmentation. Also, there will be no unused space left in the partition.
2. **Degree of Multiprogramming is Dynamic** As there is no internal fragmentation in this partition scheme due to which there is no unused space in the memory. Thus more processes can be loaded into the memory at the same time.
3. **No Limitation on the Size of Process** In this partition scheme as the partition is allocated to the process dynamically thus the size of the process cannot be restricted because the partition size is decided according to the process size.

Disadvantages of Variable-size Partition Scheme

1. **External Fragmentation** As there is no internal fragmentation which is an advantage of using this partition scheme does not mean there will no external fragmentation.

2. **Difficult Implementation** The implementation of this partition scheme is difficult as compared to the Fixed Partitioning scheme as it involves the allocation of memory at run-time rather than during the system configuration.

Q: Explain Partition Allocation algorithms?

As available memory blocks comprised of a set of holes of various sizes that scattered throughout the memory. Whenever a process arrives and needs memory, the system searches the set for a hole that is large enough for the arrived process.

Whenever the process terminates, it releases the block of the memory and the released block is then placed back into the set of holes. If the new hole is adjacent to other holes, these adjacent holes are then merged together to form a large single hole

Partition Allocation (Memory Allocation) Strategies

In the figure given below, there are strategies that are used to select a hole from the set of available holes.

Algorithms for partition allocation
First Fit Algorithm
Best Fit Algorithm
Worst Fit Algorithm

1. First Fit Algorithm-

- This algorithm starts scanning the partitions serially from the starting.
- When an empty partition that is big enough to store the process is found, it is allocated to the process.
- Obviously, the partition size has to be greater than or at least equal to the process size.

2. Best Fit Algorithm-

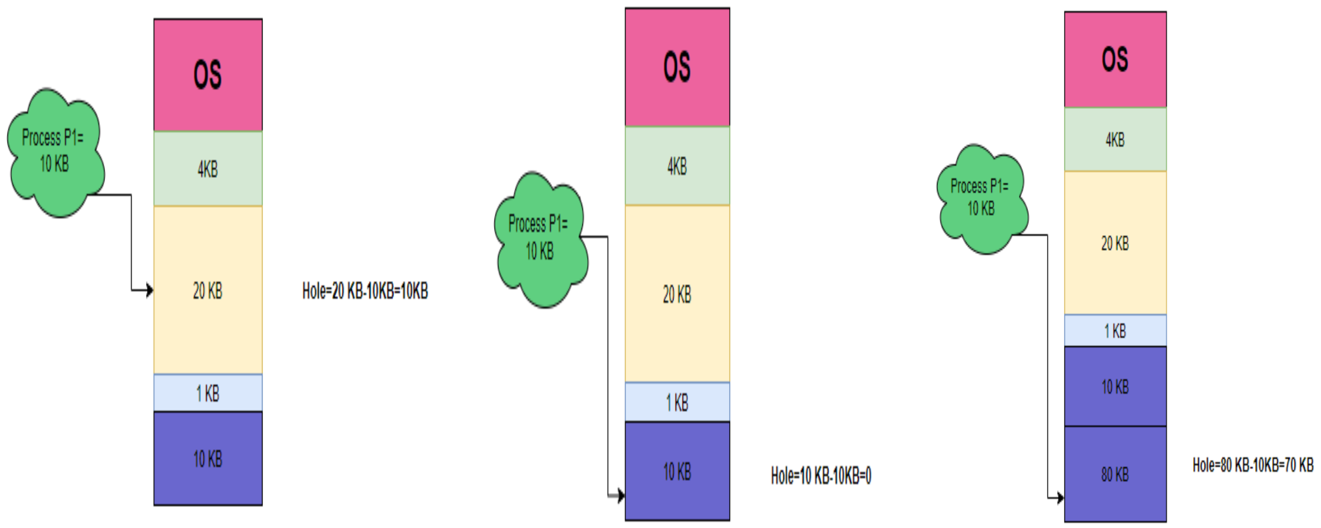
- This algorithm first scans all the empty partitions.
- It then allocates the smallest size partition to the process.

3. Worst Fit Algorithm-

- This algorithm first scans all the empty partitions.
- It then allocates the largest size partition to the process

Let us take the example given below:

Process P1 of size 10KB has arrived and then the first hole that is enough to meet the requirements of size 20KB is chosen and allocated to the process by using different algorithems



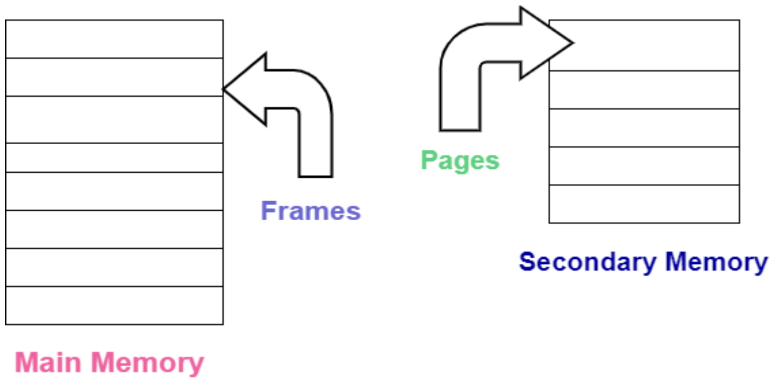
Q: Explain about Paging?

- In Operating Systems, Paging is a storage mechanism used to retrieve processes from the secondary storage into the main memory in the form of pages.
- The main idea behind the paging is to divide each process in the form of **pages**. The main memory will also be divided in the form of **frames**.
- One page of the process is to be stored in one of the frames of the memory. The pages can be stored at the different locations of the memory but the priority is always to find the contiguous frames or holes.
- Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage.
- Paging permits the **physical address space** of a process to be **non-contiguous.** It is a **fixed-size partitioning scheme**. In the Paging technique, the secondary memory and main memory are divided into equal fixed-size partitions.
- Paging helps to **avoid external fragmentation** and the **need for compaction**.

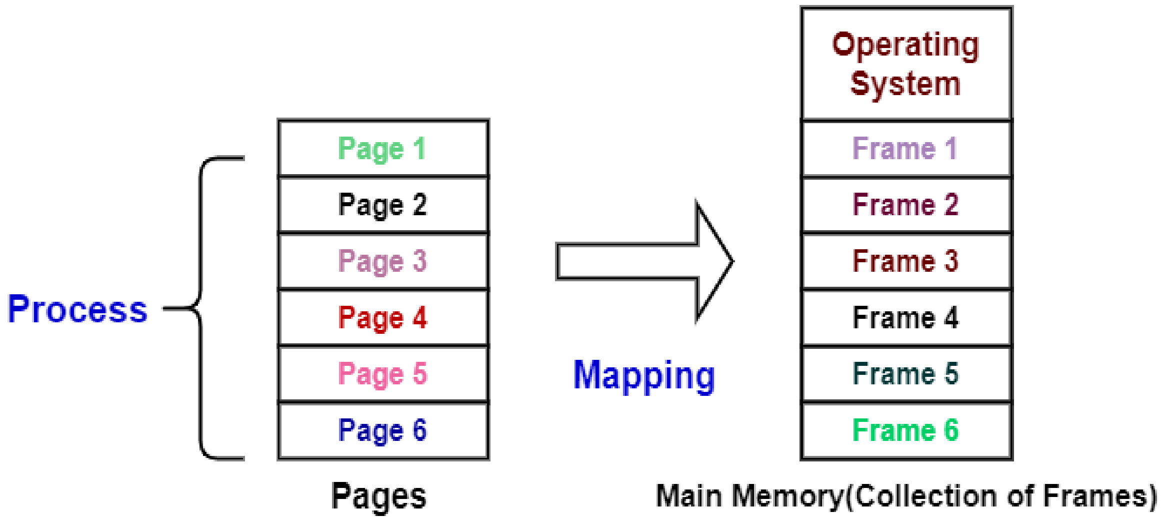
Basic Method of Paging

The paging technique divides the physical memory(main memory) into fixed-size blocks that are known as **Frames** and also divide the **logical memory(secondary memory) into blocks of the same size** that are known as **Pages.**

The Frame has the same size as that of a Page. A frame is basically a place where a (logical) page can be (physically) placed.



Each process is mainly divided into parts where the size of each part is the same as the page size.



Translation of Logical Address into Physical Address

- The CPU always generates a logical address.
- In order to access the main memory always a physical address is needed.

The **logical address generated by CPU always consists of two parts:**

1. Page Number(p)
2. Page Offset (d)

where,

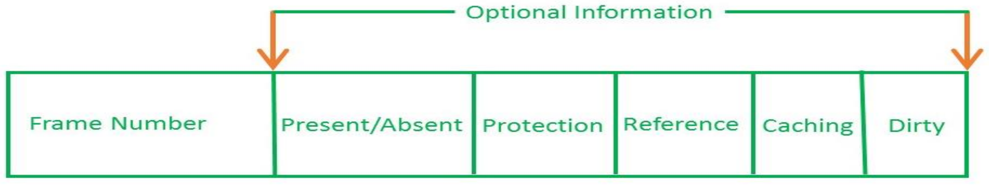
Page Number is used to specify the specific page of the process from which the CPU wants to read the data. and it is also used as an index to the page table.

and **Page offset** is mainly used to specify the specific word on the page that the CPU wants to read.

Structure Of A Page Table:

Page table has a table entries where each page table entry stores a frame number and optional status (like protection) bits. Many of status bits used in the virtual memory system. The most **important** thing in PTE is **frame Number**.

Page table entry has the following information –



PAGE TABLE ENTRY

a) **Frame Number :** It gives the frame number in which the current page you are looking for is present. The number of bits required depends on the number of frames.

Number of bits for frame = Size of physical memory/frame size

b) **Present/Absent bit :** Present or absent bit says whether a particular page you are looking for is present or absent. In case if it is not present, that is called Page Fault. It is set to 0 if the corresponding page is not in memory. Used to control page fault by the operating system to support virtual memory. Sometimes this bit is also known as **valid/invalid** bits.

c) **Protection bit :** Protection bit says that what kind of protection you want on that page. So, these bit for the protection of the page frame (read, write etc).

d) **Referenced bit :** Referenced bit will say whether this page has been referred in the last clock cycle or not. It is set to 1 by hardware when the page is accessed.

e) **Caching enabled/disabled :** Sometimes we need the fresh data. Let us say the user is typing some information from the keyboard and your program should run according to the input. CPU should be able to see it as first as possible. That is the reason we want to disable caching. So, this bit **enables or disable** caching of the page.

f) **Modified bit –** Modified bit says whether the page has been modified or not. Modified means sometimes you might try to write something on to the page. If a page is modified, then whenever you should replace that page with some other page, then the modified information should be kept on the hard disk or it has to be written back or it has to be saved back. It is set to

1 by hardware on write-access to page which is used to avoid writing when swapped out. Sometimes this modified bit is also called as the **Dirty bit**.

Address Translation:

Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

Frame address is called **physical address** and represented by a **frame number** and the **offset**.

Physical Address = Frame number + page offset

A data structure called **page map table** is used to keep track of the relation between a pages of a process to a frame in physical memory.

So, **The physical address consists of two parts:**

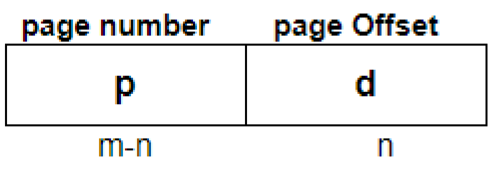
1. Page offset(d)
2. Frame Number(f)

where,

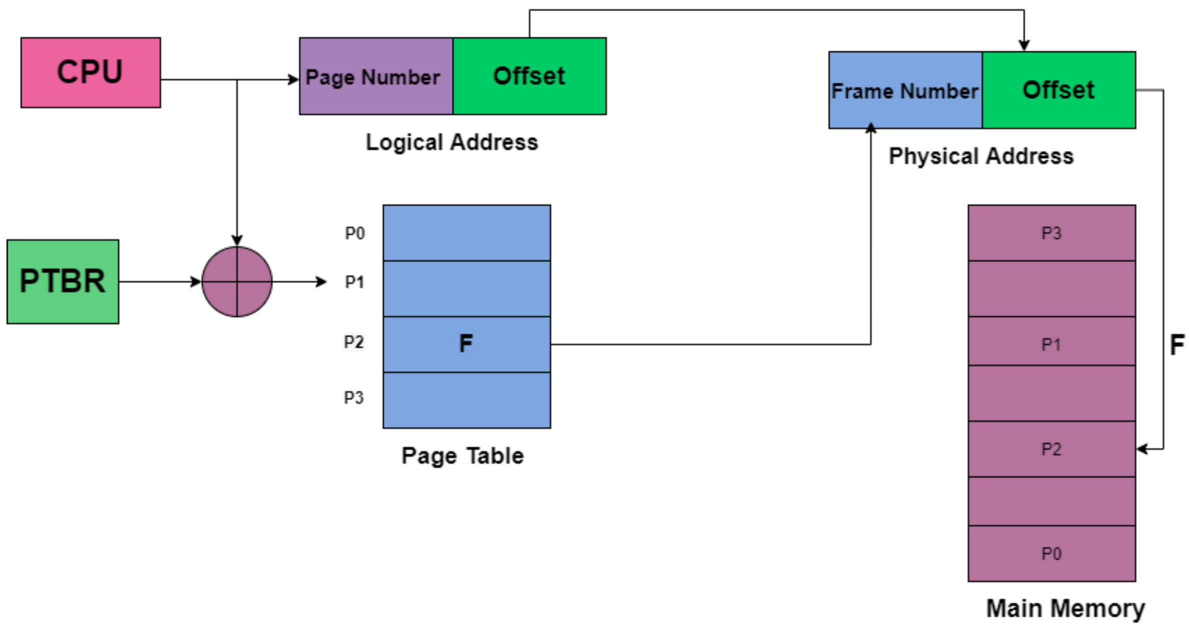
The **Frame number** is used to indicate the specific frame where the required page is stored.

and **Page Offset** indicates the specific word that has to be read from that page.

The logical address is as follows:



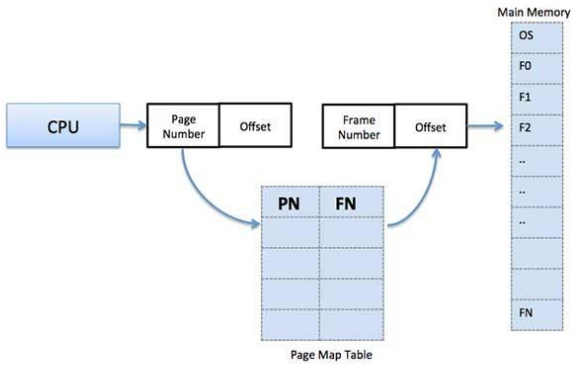
where **p** indicates the index into the page table, and **d** indicates the displacement within the page.



The **PTBR** in the above diagram means page table base register and it basically holds the base address for the page table of the current process.

When the system allocates a frame to any page, it translates this logical address into a physical address and creates entry into the page table to be used throughout execution of the program.

When a process is to be executed, its corresponding pages are loaded into any available memory frames.



Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture. When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.

This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

Advantages and Disadvantages of Paging:

a) Paging reduces external fragmentation, but still suffers from internal fragmentation.
b) Paging is simple to implement and assumed as an efficient memory management technique.
c) Due to equal size of the pages and frames, swapping becomes very easy.
d) Page table requires extra memory space, so may not be good for a system having small RAM.

Q: EXPLAIN ABOUT SEGMENTATION?

Segmentation is another way of dividing the addressable memory. It is another scheme of memory management and it generally supports the user view of memory.

The Logical address space is basically the collection of segments. Each segment has a name and a length.

Basically, a process is divided into segments. Like paging, segmentation divides or segments the memory. But there is a difference and that is while the **paging** divides the memory into a **fixed size** and on the other hand, segmentation divides the **memory into variable segments** these are then loaded into logical memory space.

A Program is basically a collection of segments. And a segment is a logical unit such as:

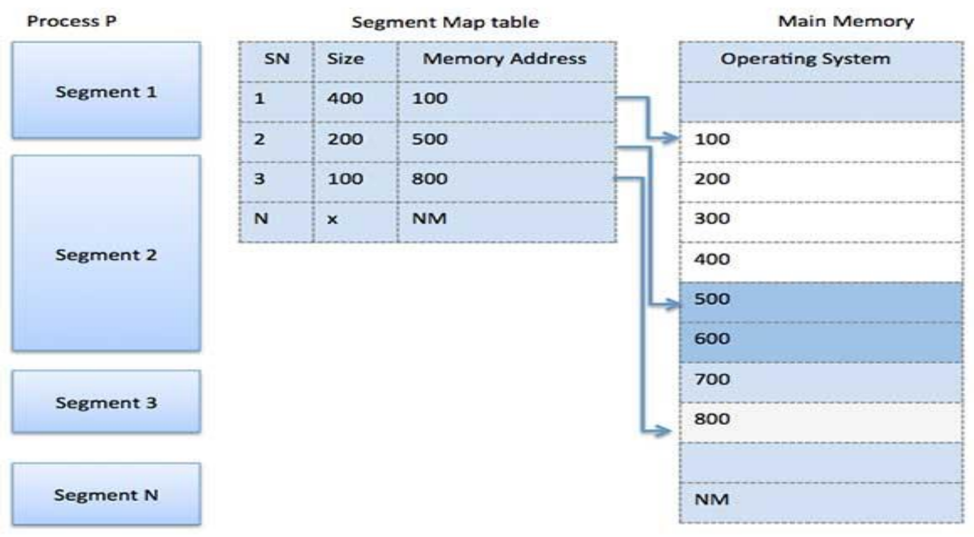
- main program
- procedure
- function
- method
- object
- local variable and global variables.
- symbol table
- common block
- stack
- arrays

Types of Segmentation:**Given below are the types of Segmentation:**

- **Virtual Memory Segmentation** With this type of segmentation, each process is segmented into n divisions and the most important thing is they are not segmented all at once.
- **Simple Segmentation** With the help of this type, each process is segmented into n divisions and they are all together segmented at once exactly but at the runtime and can be non-contiguous (that is they may be scattered in the memory).

Some characteristics of the segmentation technique are as follows:

- The Segmentation partitioning scheme is **variable-size.**
- Partitions of the secondary memory are commonly known as **segments**.
- Partition size mainly depends upon the length of modules.
- Thus with the help of this technique, secondary memory and main memory are divided into unequal-sized partitions.



Q:Explain the differences between Paging and Segmentation.

Sr. No.	Key	Paging	Segmentation
1	**Memory Size**	In Paging, a process address space is broken into fixed sized blocks called pages.	In Segmentation, a process address space is broken in varying sized blocks called sections.
2	**Accountability**	Operating System divides the memory into pages.	Compiler is responsible to calculate the segment size, the virtual address and actual address.
3	**Size**	Page size is determined by available memory.	Section size is determined by the user.
4	**Speed**	Paging technique is faster in terms of memory access.	Segmentation is slower than paging.
5	**Fragmentation**	Paging can cause internal fragmentation as some pages may go underutilized.	Segmentation can cause external fragmentation as some memory block may not be used at all.
6	**Logical Address**	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.

Sr. No.	Key	Paging	Segmentation
7	**Table**	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.
8	**Data Storage**	Page table stores the page data.	Segmentation table stores the segmentation data.

Q: Explain about virtual memory?

- Virtual Memory is a space where large programs can store themselves in form of pages while their execution and only the required pages or portions of processes are loaded into the main memory.
- This technique is useful as a large virtual memory is provided for user programs when a very small physical memory is there. Thus Virtual memory is a technique that allows the execution of processes that are not in the physical memory completely.
- Virtual Memory mainly gives the illusion of more physical memory than there really is with the help of Demand Paging.

In real scenarios, most processes never need all their pages at once, for the following reasons

- Error handling code is not needed unless that specific error occurs, some of which are quite rare.
- Arrays are often over-sized for worst-case scenarios, and only a small fraction of the arrays are actually used in practice.
- Certain features of certain programs are rarely used.

Virtual memory can be implemented with the help of:-

1. Demand Paging
2. Demand Segmentation

Benefits of having Virtual Memory

1. Large programs can be written, as the virtual space available is huge compared to physical memory.
2. Less I/O required leads to faster and easy swapping of processes.
3. More physical memory available, as programs are stored on virtual memory, so they occupy very little space on actual physical memory.

4. Therefore, the Logical address space can be much larger than that of physical address space.
5. Virtual memory allows address spaces to be shared by several processes.
6. During the process of creation, virtual memory allows: **copy-on-write** and **Memory-mapped files**

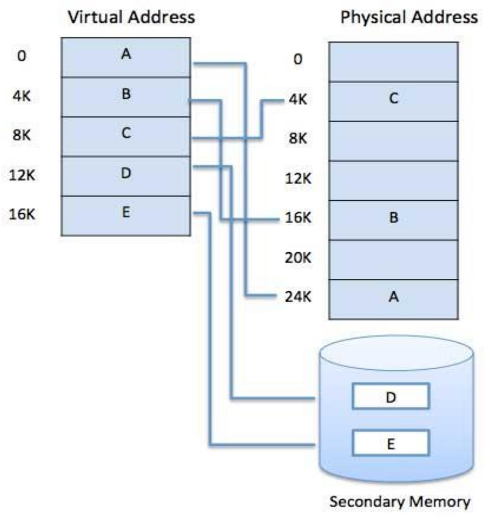
Q: Explain about Demand Paging

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). Initially, only those pages are loaded which will be required by the process immediately.

The pages that are not moved into the memory, are marked as invalid in the page table. For an invalid entry, the rest of the table is empty. In the case of pages that are loaded in the memory, they are marked as valid along with the information about where to find the swapped-out page.

Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

It is a technique that is implemented using both hardware and software. It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory.



1. All memory references within a process are logical addresses that are dynamically translated into physical addresses at run time. This means that a process can be swapped in and out of main memory such that it occupies different places in main memory at different times during the course of execution.
2. A process may be broken into number of pieces and these pieces need not be continuously located in the

main memory during execution. The combination of dynamic run-time addres translation and use of page or segment table permits this.

If these characteristics are present then, it is not necessary that all the pages or segments are present in the main memory during execution. This means that the required pages need to be loaded into memory whenever required. Virtual memory is implemented using Demand Paging or Demand Segmentation.

Page Replacement

In Demand Paging, only certain pages of a process are loaded initially into the memory. This allows us to get more processes into memory at the same time. but what happens when a process requests for more pages and no free memory is available to bring them in.

Following steps can be taken to deal with this problem :

1. Put the process in the wait queue, until any other process finishes its execution thereby freeing frames.
2. Or, remove some other process completely from the memory to free frames.
3. Or, find some pages that are not being used right now, move them to the disk to get free frames. This technique is called **Page replacement** and is most commonly used. We have some great algorithms to carry on page replacement efficiently.

Thrashing

A process that is spending more time paging than executing is said to be **thrashing**. In other words, it means, that the process doesn't have enough frames to hold all the pages for its execution, so it is swapping pages in and out very frequently to keep executing. Sometimes, the pages which will be required in the near future have to be swapped out.

Initially, when the CPU utilization is low, the process scheduling mechanism, to increase the level of multiprogramming loads multiple processes into the memory at the same time, allocating a limited amount of frames to each process.

As the memory fills up, the process starts to spend a lot of time for the required pages to be swapped in, again leading to low CPU utilization because most of the processes are waiting for pages. Hence the scheduler loads more processes to increase CPU utilization, as this continues at a point in time the complete system comes to a stop.

Advantages of Virtual Memory

- Virtual Memory allows you to run more applications at a time.
- With the help of virtual memory, you can easily fit many large programs into smaller programs.
- With the help of Virtual memory, a multiprogramming environment can be easily implemented.
- As more processes should be maintained in the main memory which leads to the effective utilization of the CPU.
- Data should be read from the disk at the time when required.
- Common data can be shared easily between memory.
- With the help of virtual memory, speed is gained when only a particular segment of the program is required for the execution of the program.
- The process may even become larger than all of the physical memory.

Disadvantages of Virtual Memory

- Virtual memory reduces the stability of the system.
- The performance of Virtual memory is not as good as that of RAM.
- If a system is using virtual memory then applications may run slower.
- Virtual memory negatively affects the overall performance of a system.
- Virtual memory occupies the storage space, which might be otherwise used for long-term data storage.
- This memory takes more time in order to switch between applications.

Q: Explain different Page Replacement Algorithms?

In operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults.

Page Fault – A page fault is a type of interrupt, raised by the hardware when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page **p**, then any immediately following references to page **p** will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.

There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

A. First In First Out (FIFO) –

This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal. Oldest page in main memory is the one which will be selected for replacement. Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

Example-1:

Consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots

1	**5**	**5**
3	**3**	**6**
0	**0**	**0**

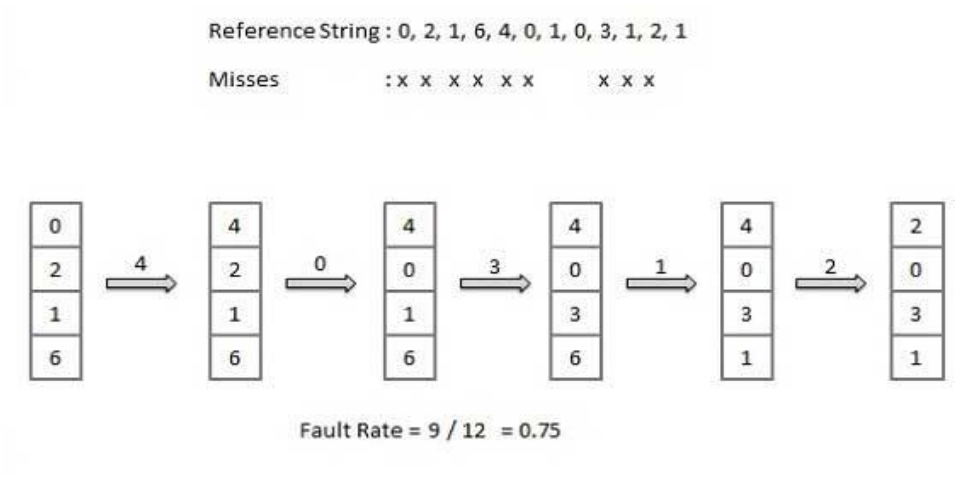
Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —>**3 Page Faults.**

When 3 comes, it is already in memory so —>**0 Page Faults.**

Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1.-->**1 Page Fault.**

Finally 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**

Example-2:



Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses :x x x x x x x x x

Fault Rate = 9 / 12 = 0.75

Example 3:

Let's have a reference string: a, b, c, d, c, a, d, b, e, b, a, b, c, d and the size of the frame be In the above example the pages are replaced by removing the first page in the queue. This resulted in causing a total of 9 page faults using FIFO algorithm.

Time Req		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Page frames		a	b	c	d	c	a	d	b	e	b	a	b	c	d
0	a	a	a	a	a	a	a	a	a	e	e	e	e	e	d
1	b		b	b	b	b	b	b	b	b	b	a	a	a	a
2	c			c	c	c	c	c	c	c	c	c	b	b	b
3	d				d	d	d	d	d	d	d	d	d	c	c
FAULTS		x	x	x	x					x		x	x	x	x

B. Optimal Page replacement-

An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN. Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

Ex: Let us consider page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 and 4 page slots.

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**

0 is already there so —>**0 Page fault.**

when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**

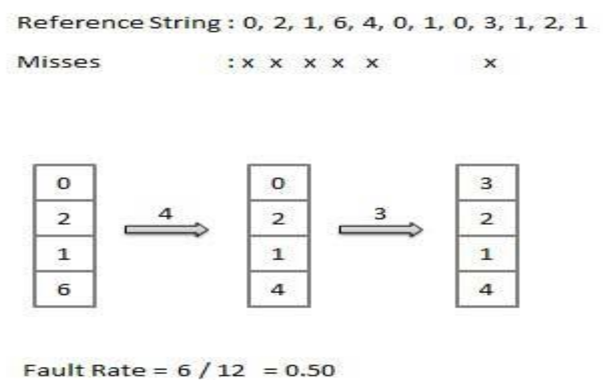
0 is already there so —> **0 Page fault.**

4 will takes place of 1 —> **1 Page Fault.**

Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

7	**7**	**3**	**3**	**3**
0	**0**	**0**	**0**	**0**
1	**1**	**1**	**1**	**4**
2	**2**	**2**	**2**	**2**

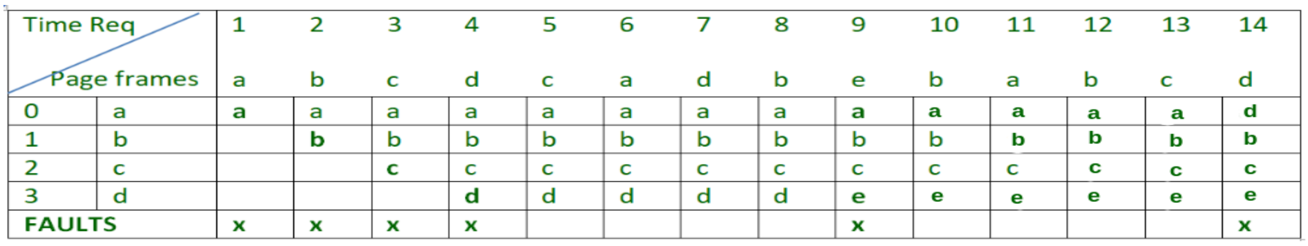
Example-2:


Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x

Fault Rate = 6 / 12 = 0.50

Example-3: Let's have a reference string: a, b, c, d, c, a, d, b, e, b, a, b, c, d and the size of the frame be



Time Req / Page frames		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		a	b	c	d	c	a	d	b	e	b	a	b	c	d
0	a	a	a	a	a	a	a	a	a	a	a	a	a	a	d
1	b		b	b	b	b	b	b	b	b	b	b	b	b	b
2	c			c	c	c	c	c	c	c	c	c	c	c	c
3	d				d	d	d	d	d	e	e	e	e	e	e
FAULTS		x	x	x	x					x					x

There are 6 page faults using optimal algorithm.

Optimal page replacement is perfect, but not possible in practice as operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

C. Least Recently Used:

Page which has not been used for the longest time in main memory is the one which will be selected for replacement. Easy to implement, keep a list, replace pages by looking back into time. In this algorithm page will be replaced which is least recently used.

Example-1: Let say the page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2. Initially we have 4 page slots empty.

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**

0 is already their so —>**0 Page fault.**

When 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**

0 is already in memory so —> **0 Page fault**.

4 will takes place of 1 —> **1 Page Fault**

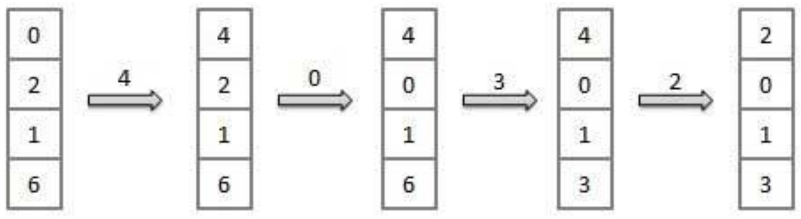
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

7	**7**	**3**	**3**	**3**
0	**0**	**0**	**0**	**0**
1	**1**	**1**	**1**	**4**
2	**2**	**2**	**2**	**2**

Example-2:

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses :x x x x x x x x



Fault Rate = 8 / 12 = 0.67

Example: Let's have a reference string: a, b, c, d, c, a, d, b, e, b, a, b, c, d and the size of the frame be 4.

Time Req		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Page frames		a	b	c	d	c	a	d	b	e	b	a	b	c	d
0	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a
1	b		b	b	b	b	b	b	b	b	b	b	b	b	b
2	c			c	c	c	c	c	c	e	e	e	e	e	d
3	d				d	d	d	d	d	d	d	d	d	c	c
FAULTS		x	x	x	x					x				x	x

There are 7 page faults using LRU algorithm.

D. Page buffering algorithm

⇒ To get a process start quickly, keep a pool of free frames.

⇒ On page fault, select a page to be replaced.

⇒ Write the new page in the frame of free pool, mark the page table and restart the process.

⇒ Now write the dirty page out of disk and place the frame holding replaced page in free pool.

E. Least frequently Used (LFU) algorithm

⇒ The page with the smallest count is the one which will be selected for replacement.

⇒ This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

F. Most frequently Used (MFU) algorithm

⇒ This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

UNIT V

File and I/O Management, OSsecurity : DirectoryStructure, File Operations, File Allocation Methods, Device Management, Pipes, Buffer, Shared Memory, Security Policy Mechanism, Protection, Authentication and Internal Access Authorization

Introduction to Android Operating System, Android Development Framework, Android Application Architecture, Android Process Management and File System, Small Application Development using Android Development Framework

Introduction

File management is one of the basic and important features of operating system. Operating system is used to manage files of computer system. All the files with different extensions are managed by operating system.

A file is collection of specific information stored in the memory of computer system. File management is defined as the process of manipulating files in computer system, it management includes the process of creating, modifying and deleting the files.

Objective of File management System

Here are the main objectives of the file management system:

- It provides I/O support for a variety of storage device types.
- Minimizes the chances of lost or destroyed data
- Helps OS to standardized I/O interface routines for user processes.
- It provides I/O support for multiple users in a multiuser systems environment.

Properties of a File System

Here, are important properties of a file system:

- Files are stored on disk or other storage and do not disappear when a user logs off.
- Files have names and are associated with access permission that permits controlled sharing.
- Files could be arranged or more complex structures to reflect the relationship between them.

File structure

A File Structure needs to be predefined format in such a way that an operating system understands. It has an exclusively defined structure, which is based on its type.

Three types of files structure in OS:

- A text file: It is a series of characters that is organized in lines.
- An object file: It is a series of bytes that is organized into blocks.
- A source file: It is a series of functions and processes.

File Attributes

A file has a name and data. Moreover, it also stores meta information like file creation date and time, current size, last modified date, etc. All this information is called the attributes of a file system.

Here, are some important File attributes used in OS:

- ❖ **Name:** It is the only information stored in a human-readable form.
- ❖ **Identifier**: Every file is identified by a unique tag number within a file system known as an identifier.
- ❖ **Location:** Points to file location on device.
- ❖ **Type:** This attribute is required for systems that support various types of files.
- ❖ **Size**. Attribute used to display the current file size.
- ❖ **Protection**. This attribute assigns and controls the access rights of reading, writing, and executing the file.
- ❖ **Time, date and security:** It is used for protection, security, and also used for monitoring

File Type

It refers to the ability of the operating system to differentiate various types of files like text files, binary, and source files. However, Operating systems like MS_DOS and UNIX has the following type of files:

Character Special File

It is a hardware file that reads or writes data character by character, like mouse, printer, and more.

Ordinary files

- These types of files stores user information.
- It may be text, executable programs, and databases.
- It allows the user to perform operations like add, delete, and modify.

Directory Files

- Directory contains files and other related information about those files. Its basically a folder to hold and organize multiple files.

Special Files

- These files are also called device files. It represents physical devices like printers, disks, networks, flash drive, etc.

Q: Explain different Operations on Files?

The operations that can performed on a file are −

Creating a file

To create a file, there should be space in the file system. Then the entry for the new file must be made in the directory. This entry should contain information about the file such as its name, its location etc.

Reading a file

To read from a file, the system call should specify the name and location of the file. There should be a read pointer at the location where the read should take place. After the read process is done, the read pointer should be updated.

Writing a file

To write into a file, the system call should specify the name of the file and the contents that need to be written. There should be a write pointer at the location where the write should take place. After the write process is done, the write pointer should be updated.

Deleting a file

The file should be found in the directory to delete it. After that all the file space is deleted so it can be reused by other files.

Repositioning in a file

This is also known as file seek. To reposition a file, the current file value is set to the appropriate entry. This does not require any actual I/O operations.

Truncating a file

This deletes the data from the file without destroying all its attributes. Only the file length is reset to zero and the file contents are erased. The rest of the attributes remain the same.

Q: Explain about File Systems?

A file is a collection of related information that is recorded on secondary storage. Or file is a collection of logically related entities. From user’s perspective a file is the smallest allotment of logical secondary storage.

Attributes	Types	Operations
Name	Doc	Create
Type	Exe	Open
Size	Jpg	Read
Creation Data	Xis	Write
Author	C	Append
Last Modified	Java	Truncate
protection	class	Delete
		Close

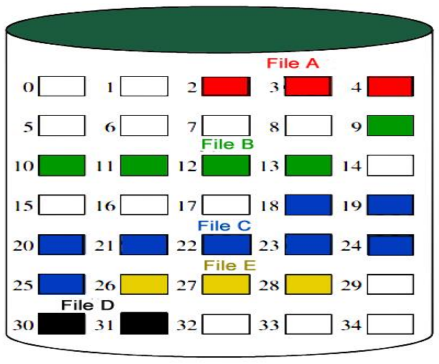
File type	Usual extension	Function
Executable	exe, com, bin	Read to run machine language program
Object	obj, o	Compiled, machine language not linked
Source Code	C, java, pas, asm, a	Source code in various languages
Batch	bat, sh	Commands to the command interpreter
Text	txt, doc	Textual data, documents
Word Processor	wp, tex, rrf, doc	Various word processor formats
Archive	arc, zip, tar	Related files grouped into one compressed file
Multimedia	mpeg, mov, rm	For containing audio/video information

Q. Explain about File Allocation Methods?

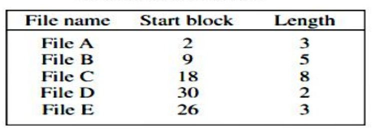
Continuous Allocation: A single continuous set of blocks is allocated to a file at the time of file creation. Thus, this is a pre-allocation strategy, using variable size portions.

The file allocation table needs just a single entry for each file, showing the starting block and the length of the file. This method is best from the point of view of the individual sequential file.

Multiple blocks can be read in at a time to improve I/O performance for sequential processing. It is also easy to



File allocation table



File name	Start block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

retrieve a single block. For example, if a file starts at block b, and the ith block of the file is wanted, its location on secondary storage is simply b+i-1.

Disadvantage

- External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length. Compaction algorithm will be necessary to free up additional space on disk.
- Also, with pre-allocation, it is necessary to declare the size of the file at the time of creation.

Linked Allocation (Non-contiguous allocation) :

Allocation is on an individual block basis. Each block contains a pointer to the next block in the chain. Again the file table needs just a single entry for each file, showing the starting block and the length of the file. Although pre-allocation is possible, it is more common simply to allocate blocks as needed. Any free block can be added to the chain. The blocks need not be continuous. Increase in file size is always possible if free disk block is available. There is no external fragmentation because only one block at a time is needed but there can be internal fragmentation but it exists only in the last disk block of file.

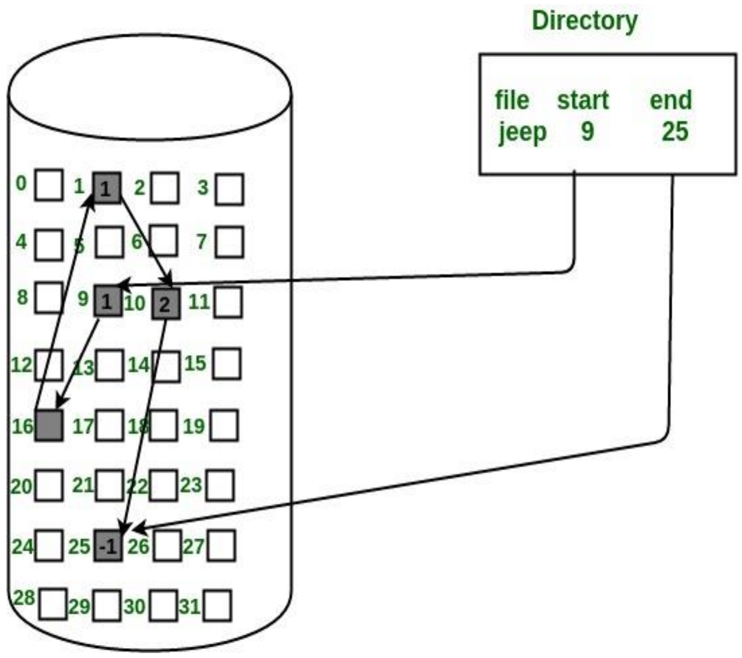
Disadvantage:

⇒ Internal fragmentation exists in last disk block of file.

⇒ There is an overhead of maintaining the pointer in every disk block.

⇒ If the pointer of any disk block is lost, the file will be truncated.

⇒ It supports only the sequential access of files.

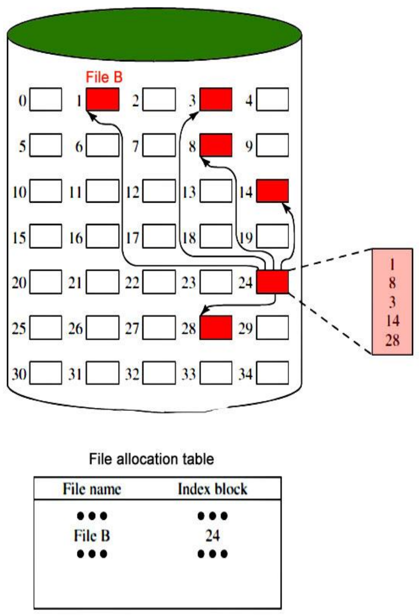


Indexed Allocation:

It addresses many of the problems of contiguous and chained allocation. In this case, the file allocation table contains a separate one-level index for each file:

The index has one entry for each block allocated to the file. Allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improves locality.

This allocation technique supports both sequential and direct access to the file and thus is the most popular form of file allocation.



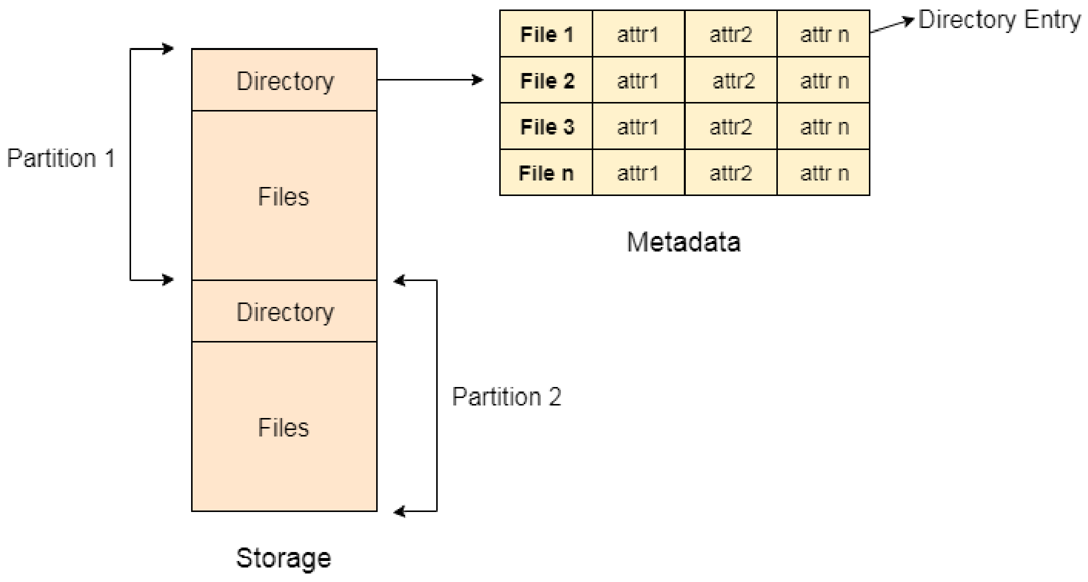
File name	Index block
•••	•••
File B	24
•••	•••

Q: Explain about DIRECTORY STRUCTURE?

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

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

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



File 1	attr1	attr2	attr n
File 2	attr1	attr2	attr n
File 3	attr1	attr2	attr n
File n	attr1	attr2	attr n

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

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

1. File Creation
2. Search for the file
3. File deletion
4. Renaming the file
5. Traversing Files
6. Listing of files
7. Delete a file
8. List a directory

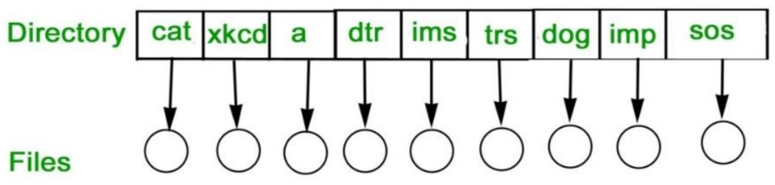
Advantages of maintaining directories are:

- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

SINGLE-LEVEL DIRECTORY

In this a single directory is maintained for all the users.

Naming problem: Users cannot have same name for two files.



Grouping problem: Users cannot group files according to their need.

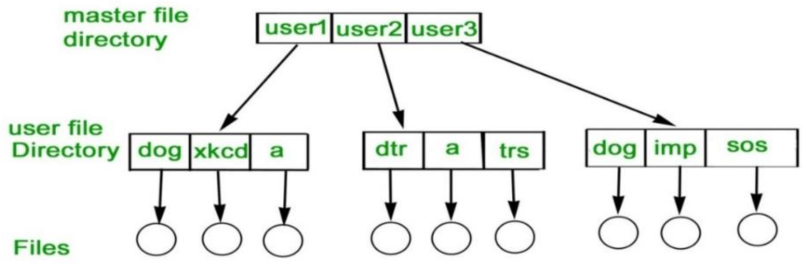
TWO-LEVEL DIRECTORY:

In this separate directories for each user is maintained.

⇒ Path name:Due to two levels there is a path name for every file to locate that file.

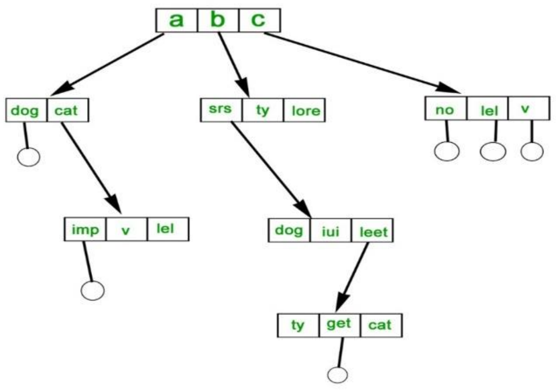
⇒ Now,we can have same file name for different user.

⇒ Searching is efficient in this method.



TREE-STRUCTURED DIRECTORY:

Directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.



Q: Explain about Device management?

Device management in operating system known as the management of the I/O devices such as a keyboard, magnetic tape, disk, printer, microphone, USB ports, scanner, etc.as well as the supporting units like control channels.

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.

The OS peripheral devices can be categorized into 3: **Dedicated, Shared, and Virtual**.

1. Dedicated : A technique where a device is assign to a single process.

2. Shared : A technique where a device shared by many processes.

3. Virtual : A technique where one physical device this simulated on another physical device. Its Combination of dedicated devices that have been transformed into shared devices.

Q: EXPLAIN ABOUT PIPES?

In computer programming, especially in UNIX operating systems, a pipe is a technique for passing information from one program process to another. a pipe is one-way communication only. Basically, a pipe passes a parameter such as the output of one process to another process which accepts it as input. The system temporarily holds the piped information until it is read by the receiving process.

Pipe System Call

- pipe() is a system call that facilitates inter-process communication. It opens a **pipe**, which is an area of main memory that is treated as a "virtual file". The pipe can be used by the creating process, as well as all its child processes, for reading and writing.
- One process can write to this "virtual file" or pipe and another related process can read from it.
- If a process tries to read before something is written to the pipe, the process is suspended until something is written.
- The pipe system call finds the first two available positions in the process's open file table and allocates them for the read and write ends of the pipe. Recall that the open system call allocates only one position in the open file table.

Q: EXPLAIN Buffering in Operating System?

The **buffer** is an area in the **main memory** used to store or hold the data **temporarily**. In other words, buffer temporarily stores data transmitted from one place to another, either between two devices or an application. The act of storing data temporarily in the buffer is called **buffering**.

A buffer may be used when moving data between processes within a computer. Buffers can be implemented in a fixed memory location in hardware or by using a virtual data buffer in software, pointing at a location in the physical memory. In all cases, the data in a data buffer are stored on a physical storage medium.

Most buffers are implemented in software, which typically uses the faster RAM to store temporary data due to the much faster access time than hard disk drives. Buffers are typically used when there is a difference between the rate of received data and the rate of processed data, for example, in a printer spooler or online video streaming.

Q: What is shared memory?

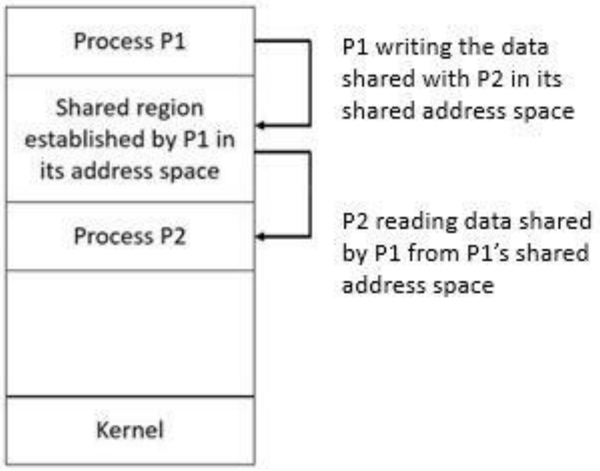
Shared memory is the fastest interprocess communication mechanism. The operating system maps a memory segment in the address space of several processes, so that several processes can read and write in that memory segment without calling operating system functions.

Shared memory system is the fundamental model of inter process communication. In a shared memory system, in the address space region the cooperating communicate with each other by establishing the shared memory region.Shared memory concept works on fastest inter process communication.

If the process wants to initiate the communication and it has some data to share, then establish the shared memory region in its address space. After that, another process wants to communicate and tries to read the shared data, and must attach itself to the initiating process's shared address space.

working mechanism:

In the Shared Memory system, the cooperating processes communicate, to exchange the data with each other. Because of this, the cooperating processes establish a shared region in their memory. The processes share data by reading and writing the data in the shared segment of the processes. Let us discuss it by considering two processes. The diagram is shown below –


Process P1
Shared region established by P1 in its address space
Process P2
Kernel
P1 writing the data shared with P2 in its shared address space
P2 reading data shared by P1 from P1's shared address space

Let the two cooperating processes P1 and P2. Both the processes P1 and P2, have their different address spaces. Now let us assume, P1 wants to share some data with P2.

So, P1 and P2 will have to perform the following steps −

Step 1 − Process P1 has some data to share with process P2. First P1 takes initiative and establishes a shared memory region in its own address space and stores the data or information to be shared in its shared memory region.

Step 2 − Now, P2 requires the information stored in the shared segment of P1. So, process P2 needs to attach itself to the shared address space of P1. Now, P2 can read out the data from there.

Step 3 − The two processes can exchange information by reading and writing data in the shared segment of the process.

The advantages of Shared Memory are as follows −

- Shared memory is a faster inter process communication system.
- It allows cooperating processes to access the same pieces of data concurrently.
- It speeds up the computation power of the system and divides long tasks into smaller sub-tasks and can be executed in parallel.
- Modularity is achieved in a shared memory system.
- Users can perform multiple tasks at a time.

Q:What is Disk Scheduling Algorithm?

A Process makes the I/O requests to the operating system to access the disk. Disk Scheduling Algorithm manages those requests and decides the order of the disk access given to the requests.

Why Disk Scheduling Algorithm is needed?

Disk Scheduling Algorithms are needed because a process can make multiple I/O requests and multiple processes run at the same time. The requests made by a process may be located at different sectors on different tracks. Due to this, the seek time may increase more. These algorithms help in minimizing the seek time by ordering the requests made by the processes.

Important Terms related to Disk Scheduling Algorithms

- **Seek Time -** It is the time taken by the disk arm to locate the desired track.
- **Rotational Latency -** The time taken by a desired sector of the disk to rotate itself to the position where it can access the Read/Write heads is called Rotational Latency.

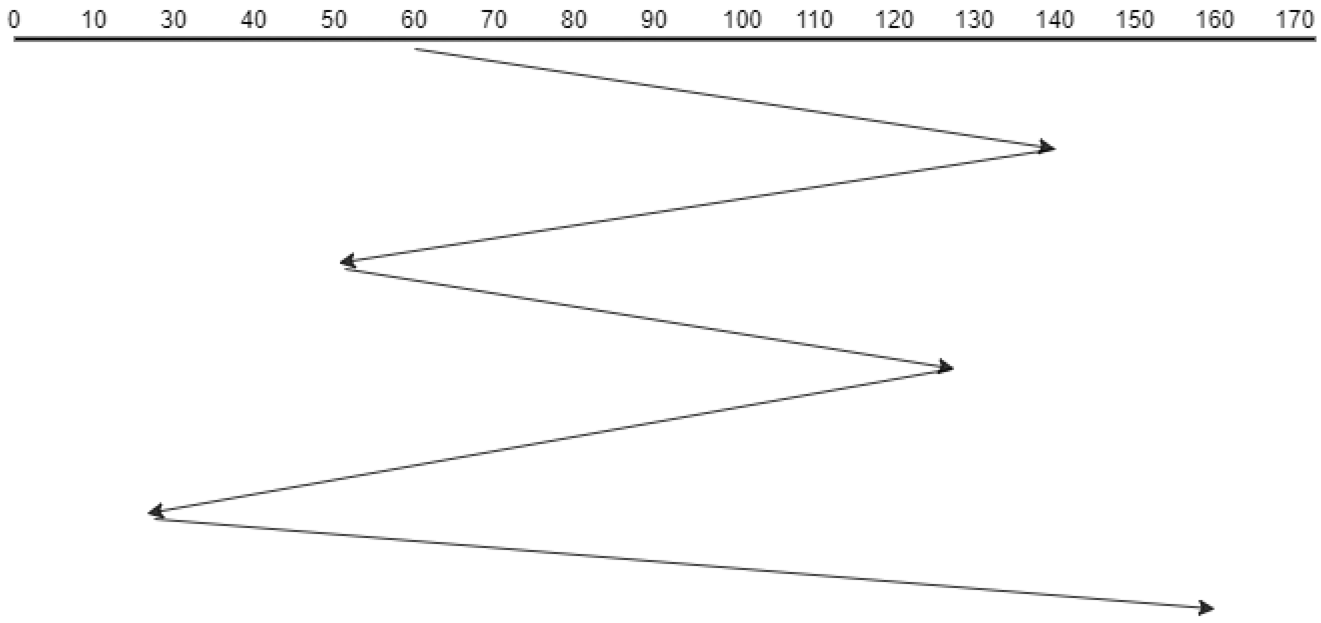
- **Transfer Time -** It is the time taken to transfer the data requested by the processes.
- **Disk Access Time -** Disk Access time is the sum of the Seek Time, Rotational Latency, and Transfer Time.

Disk Scheduling Algorithms

First Come First Serve (FCFS) :

In this algorithm, the requests are served in the order they come. Those who come first are served first. This is the simplest algorithm.

Eg. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60.

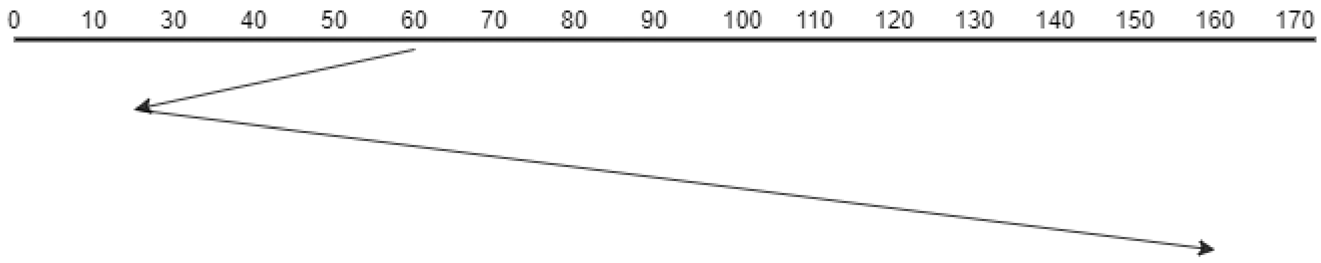


Seek Time = Distance Moved by the disk arm = (140-70)+(140-50)+(125-50)+(125-30)+(30-25)+(160-25)=480

Shortest Seek Time First (SSTF)

In this algorithm, the shortest seek time is checked from the current position and those requests which have the shortest seek time is served first. In simple words, the closest request from the disk arm is served first.

Eg. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60.

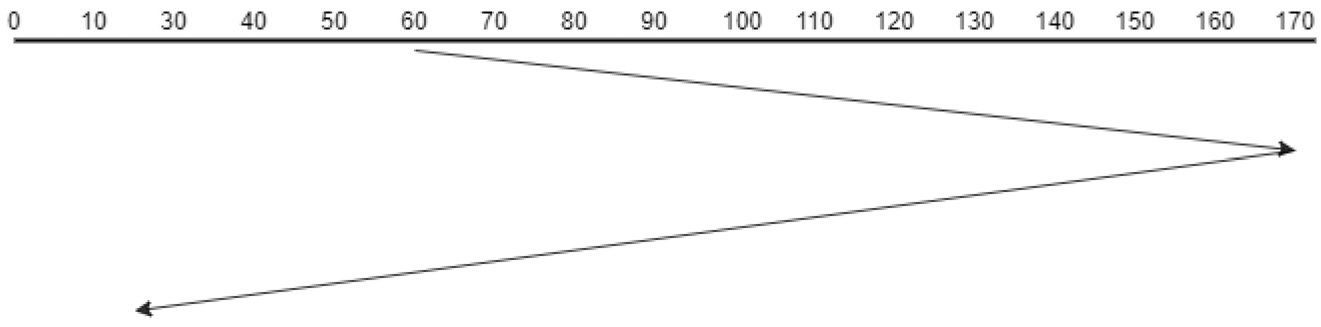


Seek Time = Distance Moved by the disk arm = (60-50)+(50-30)+(30-25)+(70-25)+(125-70)+(140-125)+(160-125)=270

SCAN

In this algorithm, the disk arm moves in a particular direction till the end and serves all the requests in its path, then it returns to the opposite direction and moves till the last request is found in that direction and serves all of them.

Eg. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60. And it is given that the disk arm should move towards the larger value.

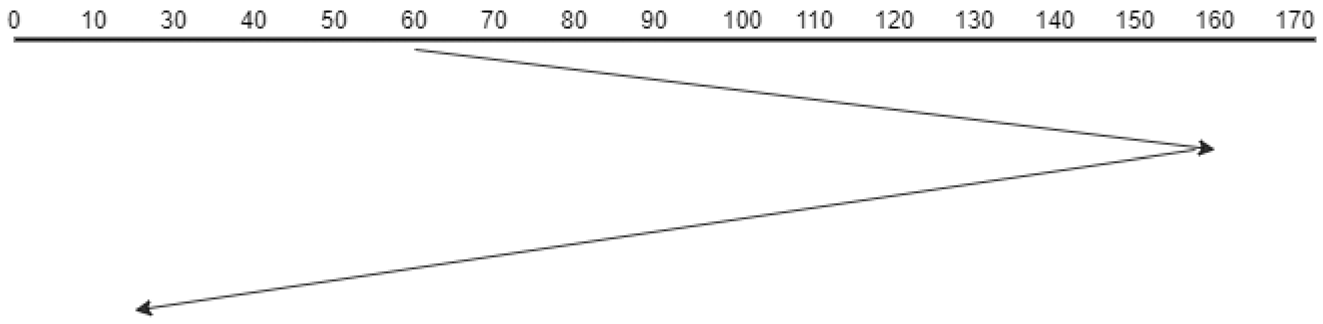


Seek Time = Distance Moved by the disk arm = (170-60)+(170-25)=255

LOOK

In this algorithm, the disk arm moves in a particular direction till the last request is found in that direction and serves all of them found in the path, and then reverses its direction and serves the requests found in the path again up to the last request found. The only difference between SCAN and LOOK is, it doesn't go to the end it only moves up to which the request is found.

Eg. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60. And it is given that the disk arm should move towards the larger value.

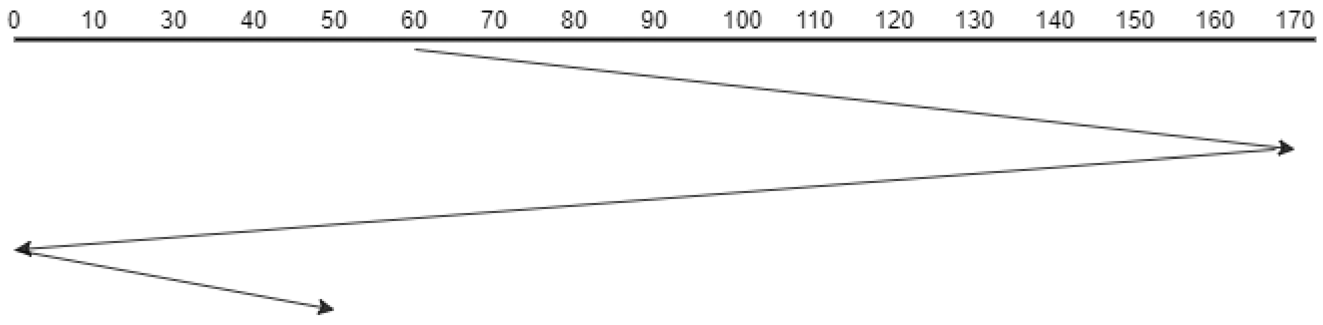


Seek Time = Distance Moved by the disk arm = (170-60)+(170-25)=235

C-SCAN

This algorithm is the same as the SCAN algorithm. The only difference between SCAN and C-SCAN is, it moves in a particular direction till the last and serves the requests in its path. Then, it returns in the opposite direction till the end and doesn't serve the request while returning. Then, again reverses the direction and serves the requests found in the path. It moves circularly.

Eg. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60. And it is given that the disk arm should move towards the larger value.

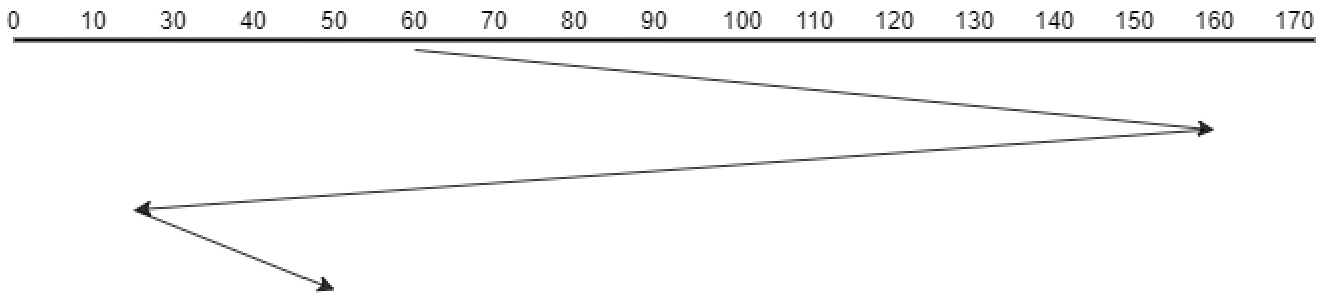


Seek Time = Distance Moved by the disk arm = (170-60)+(170-0)+(50-0)=330

C-LOOK

This algorithm is also the same as the LOOK algorithm. The only difference between LOOK and C-LOOK is, it moves in a particular direction till the last request is found and serves the requests in its path. Then, it returns in the opposite direction till the last request is found in that direction and doesn't serve the request while returning. Then, again reverses the direction and serves the requests found in the path. It also moves circularly.

Eg. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60. And it is given that the disk arm should move towards the larger value.



Seek Time = Distance Moved by the disk arm = (160-60)+(160-25)+(50-25)=260

Q: **Explain about Protection and security in OS?**

Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc. We're going to discuss following topics :

- Authentication
- One Time passwords
- Program Threats
- System Threats
- Computer Security Classifications

Authentication: Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways −

- **Username / Password** − User need to enter a registered username and password with Operating system to login into the system.
- **User card/key** − User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.
- **User attribute - fingerprint/ eye retina pattern/ signature** − User need to pass his/her attribute via designated input device used by operating system to login into the system.

One Time passwords: One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time password are implemented in various ways.

- **Random numbers** − Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
- **Secret key** − User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
- **Network password** − Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

Program Threats Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common example of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.

- **Trojan Horse** − Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** − If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** − Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** − Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generatlly a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user

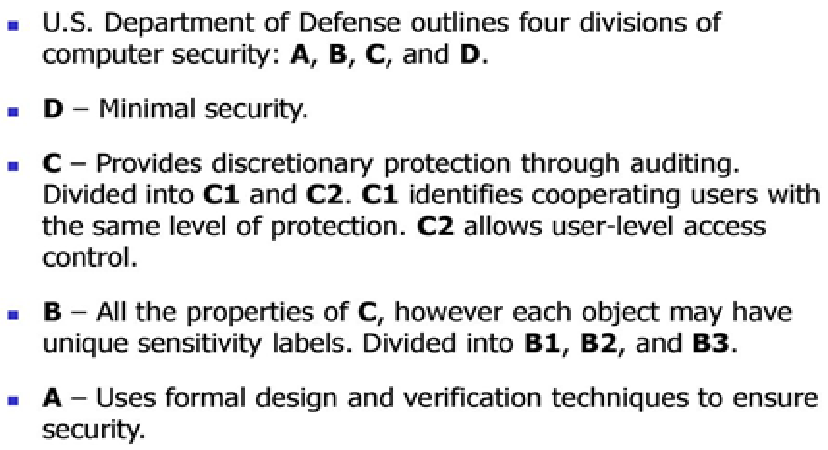
System Threats: System threats refer to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats create such an environment that operating system resources/ user files are misused. Following is the list of some well-known system threats.

- **Worm** − Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.

- **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.
- **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.

Computer Security Classifications

This is widely used specifications to determine and model the security of systems and of security solutions. Following is the brief description of each classification.



- U.S. Department of Defense outlines four divisions of computer security: **A**, **B**, **C**, and **D**.
- **D** – Minimal security.
- **C** – Provides discretionary protection through auditing. Divided into **C1** and **C2**. **C1** identifies cooperating users with the same level of protection. **C2** allows user-level access control.
- **B** – All the properties of **C**, however each object may have unique sensitivity labels. Divided into **B1**, **B2**, and **B3**.
- **A** – Uses formal design and verification techniques to ensure security.

Q: Explain about Access Authorization and Authentication in Operating System?

Access Authorization:

Authorization is the process of giving someone permission to do or have something. In multi-user computer systems, a system administrator defines for the system which users are allowed access to the system and what privileges of use (such as access to which file directories, hours of access, amount of allocated storage space, and so forth).

- Authorization is a process by which a server determines if the client has permission to use a resource or access a file.
- Authorization is usually coupled with authentication so that the server has some concept of who the client is that is requesting access.
- The type of authentication required for authorization may vary; passwords may be required in some cases but not in others.
- In some cases, there is no authorization; any user may be use a resource or access a file simply by asking for it.

Authentication in Operating System

Authentication mechanism determines the users identity before revealing the sensitive information. It is very crucial for the system or interfaces where the user priority is to protect the confidential information. In the process, the user makes a provable claim about individual identity (his or her) or an entity identity.

The credentials or claim could be a username, password, fingerprint etc. The authentication and non-repudiation, kind of issues are handled in the application layer. The inefficient authentication mechanism could significantly affect the availability of the service.

Use of Authentication in OS

- Authentication is used by a server when the server needs to know exactly who is accessing their information or site.
- Authentication is used by a client when the client needs to know that the server is system it claims to be.
- In authentication, the user or computer has to prove its identity to the server or client.
- Usually, authentication by a server entails the use of a user name and password. Other ways to authenticate can be through cards, retina scans, voice recognition, and fingerprints.
- Authentication by a client usually involves the server giving a certificate to the client in which a trusted third party such as Verisign or Thawte states that the server belongs to the entity (such as a bank) that the client expects it to.
- Authentication does not determine what tasks the individual can do or what files the individual can see. Authentication merely identifies and verifies who the person or system is.

Q: What is Android?

Android is a software package and Linux based operating system for mobile devices such as tablet computers and smart phones.

It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

There are many code names of android such as

Aestro, **Blender**, **Cupcake**, **Donut**, **Eclair**, **Froyo**, **Gingerbread**, **Honeycomb**, **Ice ream,Sandwitch**, **Jelly Bean**, **KitKat** and **Lollipop** etc.

Q:What is Open Handset Alliance (OHA)?

It's a consortium of 84 companies such as google, samsung, AKM, synaptics, KDDI, Garmin, Teleca, Ebay, Intel etc.

It was established on 5th November, 2007, led by Google. It is committed to advance open standards, provide services and deploy handsets using the Android Platform.

Features of Android:

The important features of android are given below:

A. It is open-source.
B. Anyone can customize the Android Platform.
C. There are a lot of mobile applications that can be chosen by the consumer.
D. It provides many interesting features like weather details, location finder, news coverage etc.
E. It provides support for messaging services (SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

Categories of Android applications:

There are many android applications in the market. The top categories are:

- Entertainment
- Tools
- Communication
- Productivity
- Personalization
- Music and Audio

- Social
- Media and Video
- Travel and Local etc.

Q: Explain the History of Android?

The history of android are interesting to know. Let's understand the android history in a sequence.

1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003.

2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are **Andy Rubin**, **Rich Miner**, **Chris White** and **Nick Sears**.

4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

5) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

6) In 2007, Google announces the development of android OS.

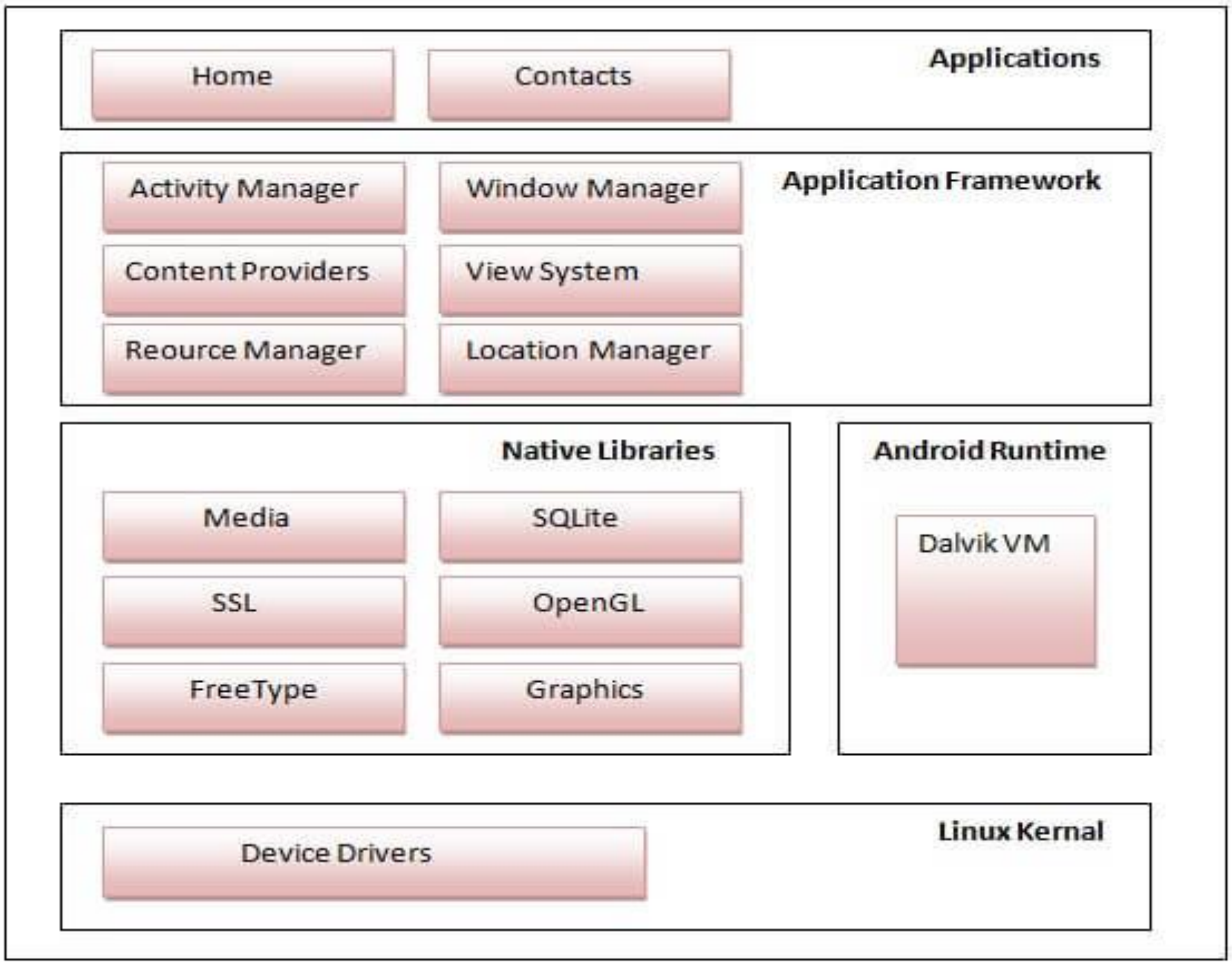
7) In 2008, HTC launched the first android mobile

Android Architecture:

Android architecture or **Android software stack** is categorized into five parts:

1. linux kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications

Let's see the android architecture first.



1) Linux kernel:

It is the heart of android architecture that exists at the root of android architecture. **Linux kernel** is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries:

On the top of linux kernel, there are **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

The WebKit library is responsible for browser support. SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

3) Android Runtime:

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

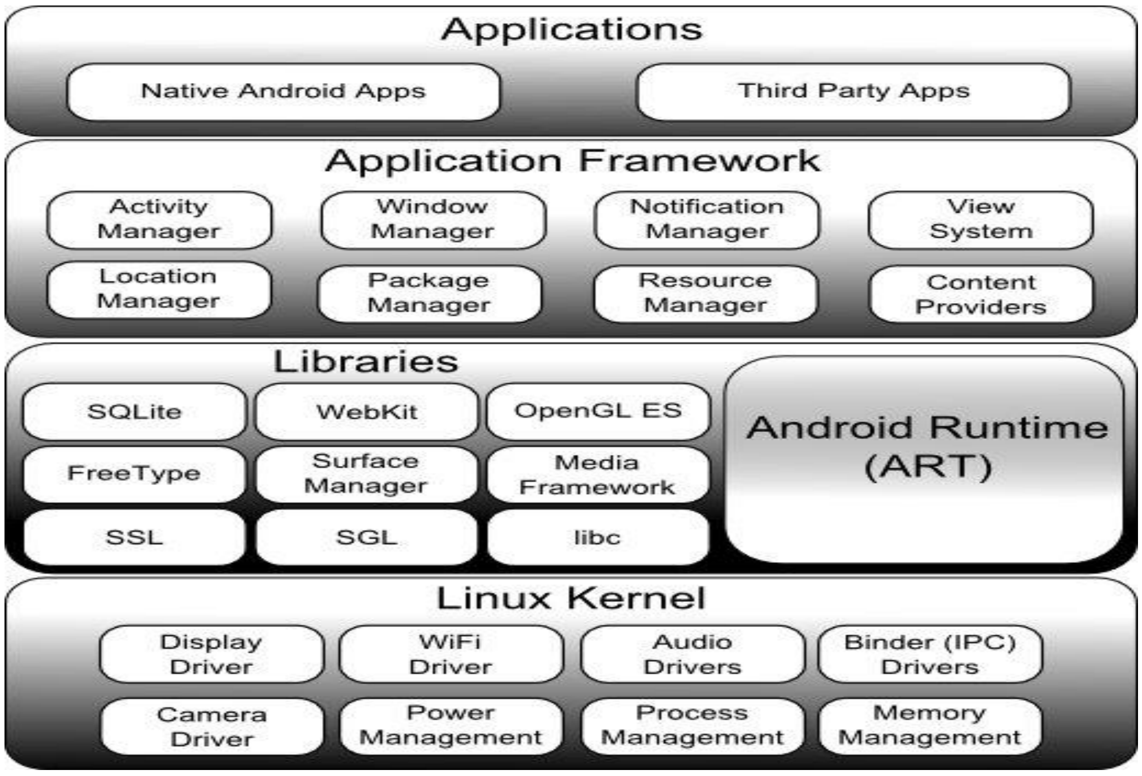
4) Android Framework:

On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5) Applications

On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernal.

Android Architecture:



The android is a operating system and is a stack of software components which is divided into five sections and four main layers that is

1. Linux kernel.
2. Libraries.
3. Android Runtime.
4. Application frame work.
5. Applications

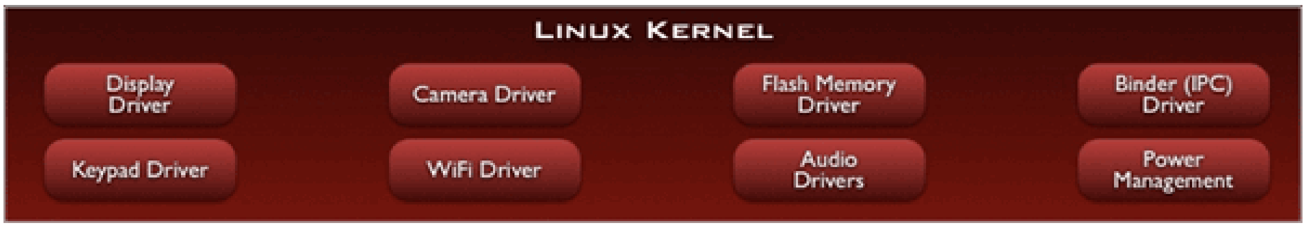
1. **Linux kernel:**

At the bottom of the Android stack is the Linux Kernel. The android uses the powerful Linux kernel and it supports wide range of hardware drivers (Drivers are programs that control hardware devices.). It never really interacts with the users and developers, but it is the heart of the whole system.

It contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

It provides the following functions in the Android system:

- ❑ Hardware Abstraction
- ❑ Memory Management Programs
- ❑ Security Settings
- ❑ Power Management Software
- ❑ Support for Shared Libraries
- ❑ Network Stack
- ❑ Other Hardware Drivers

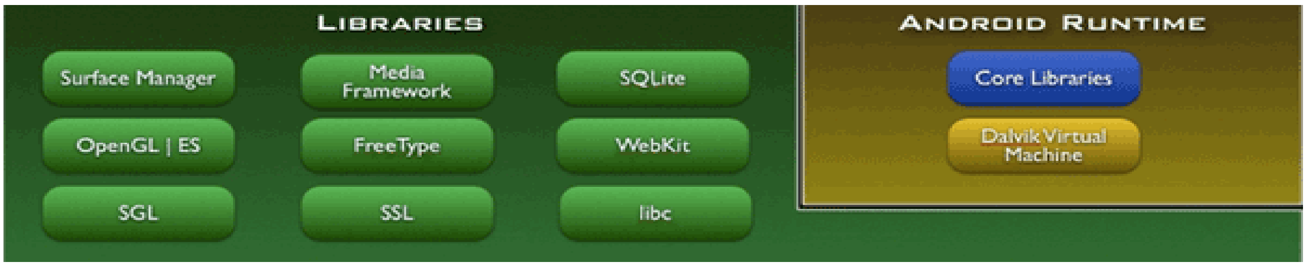


2. **Libraries:**

The on top of a Linux kennel there is a set of libraries including,

- ❑ Surface Manager: composing windows on the screen
- ❑ SGL: 2D Graphics
- ❑ Open GL|ES: 3D Library
- ❑ Media Framework for playing and recording audio and video formats.FreeType for font support
- ❑ WebKit is responsible for browser support
- ❑ libc (System C libraries)

- SQLite is a data base which is useful for storage and sharing of application data
- Open SSL for internet security



Android Libraries:

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. **A summary of some key core Android libraries available to the Android developer is as follows –**

- **android.app – Provides** access to the application model and is the cornerstone of all Android applications.
- **android.content** – Facilitates content access, publishing and messaging between applications and application components.
- **android.database** – Used to access data published by content providers and includes SQLite database management classes.
- **android.opengl** – A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** – Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** – Used to render and manipulate text on a device display.
- **android.view** – The fundamental building blocks of application user interfaces.
- **android.widget** – A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** – A set of classes intended to allow web-browsing capabilities to be built into applications.

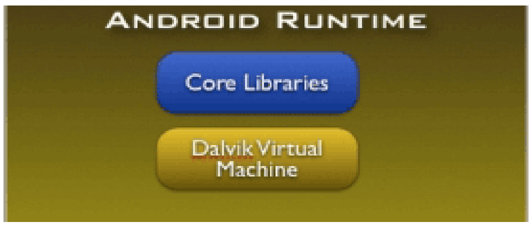
Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

3. Android Runtime:

This is the third section of the architecture and available on the second layer from the bottom. In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is It is specially designed for running apps on Android devices. DVM is like JVM but it is optimized for mobile devices. It is a Register based Virtual Machine.It consumes less memory and provides fast performance. x The Dalvik VM executes the files in the .dex format.

The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.



4. Application frame work:

On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

Activity Manager – Controls all aspects of the application lifecycle..

Content Providers – Allows applications to publish and share data with other applications.

Resource Manager – Provides access to non-code embedded resources such as strings, colour settings and user interface layouts.

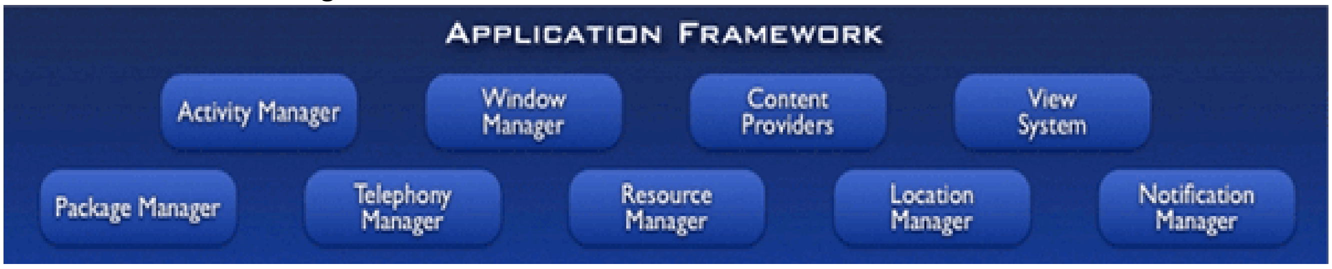
Notifications Manager – Allows applications to display alerts and notifications to the user.

View System – A set of views used to create application user interfaces.

Package Manager – The system by which applications are able to find out information about other applications currently installed on the device.

Telephony Manager – Provides information to the application about the telephony services available on the device such as status and subscriber information.

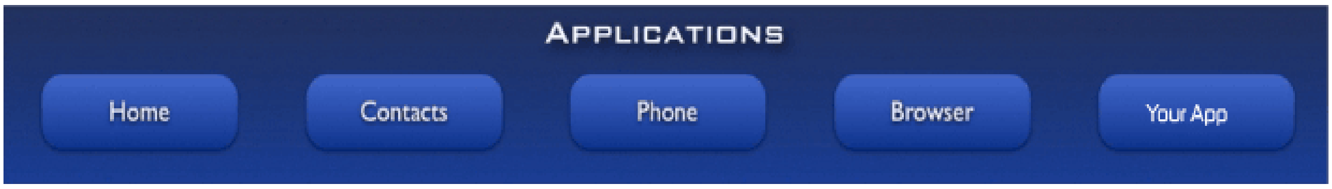
Location Manager – Provides access to the location services allowing an application to receive updates about location changes.



5. **Application Layer:**

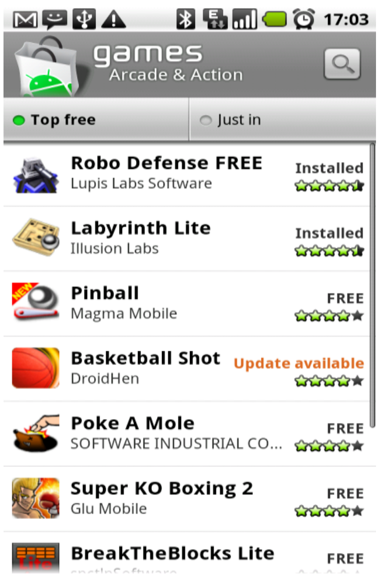
On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernal.

An average user of the Android device would mostly interact with this layer (for basic functions, such as making phone calls, accessing the Web browser etc.).



Android Market: http://www.android.com/market

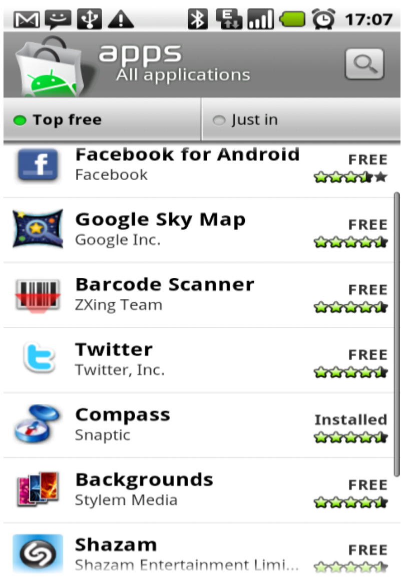
Android Market now makes it fast and easy to find awesome applications, games, and widgets for you. Search and browse over 1,50,000 and apps and games available for download, organized by category, to customize your Android experience.


17:03

games
Arcade & Action

Top free | Just in

Robo Defense FREE — Lupis Labs Software — Installed
Labyrinth Lite — Illusion Labs — Installed
Pinball — Magma Mobile — FREE
Basketball Shot — DroidHen — Update available
Poke A Mole — SOFTWARE INDUSTRIAL CO... — FREE
Super KO Boxing 2 — Glu Mobile — FREE
BreakTheBlocks Lite — FREE

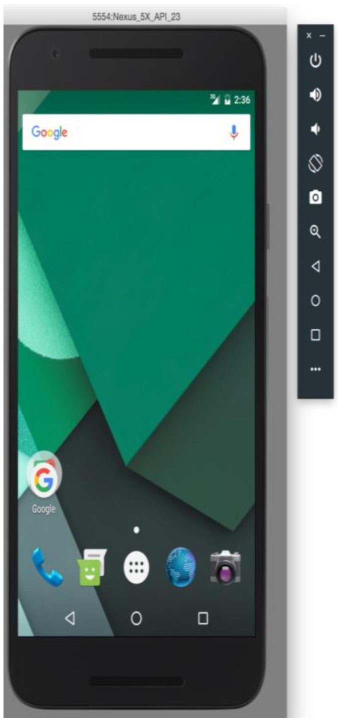

17:07

apps
All applications

Top free | Just in

Facebook for Android — Facebook — FREE
Google Sky Map — Google Inc. — FREE
Barcode Scanner — ZXing Team — FREE
Twitter — Twitter, Inc. — FREE
Compass — Snaptic — Installed
Backgrounds — Stylem Media — FREE
Shazam — Shazam Entertainment Limi... — FREE

Android emulator:



The Emulator is a new application in android operating system. The emulator is a new prototype that is used to develop and test android applications without using any physical device.

The android emulator has all of the hardware and software features like mobile device except phone calls. It provides a variety of navigation and control keys. It also provides a screen to display your application.

The emulators utilize the android virtual device configurations. Once your application is running on it, it can use services of the android platform to help other applications, access the network, play audio, video, store and retrieve the data.

Q: Explain Android - Application Components?

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application:

Sr.No	Components & Description
1	**Activities**: They dictate the UI and handle the user interaction to the smart phone screen.
2	**Services:** They handle background processing associated with an application.
3	**Broadcast Receivers:** They handle communication between Android OS and applications.
4	**Content Providers:** They handle data and database management issues.

Activities:

An activity represents a single screen with a user interface, in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of **Activity** class as follows −

```
public class MainActivity extends Activity
{
//        activity code;
}
```

Services:

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of **Service** class as follows –

```
public class MyService extends Service
        {
        }
```

Broadcast Receivers:

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and each message is broadcaster as an **Intent** object.

```
public class MyReceiver  extends  BroadcastReceiver
              {
 public void onReceive(context,intent)
{      }        }
```

Content Providers:

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends  ContentProvider
             {
              public void onCreate()
             {
                        }
             }
```

Q: Explain How to make android apps?

You need to follow the 3 steps for creating any android application.

1. Create the new android project
2. Write the message (optional)
3. Run the android application

1) Create the New Android project

For creating the new android studio project:

1) Select *Start a new Android Studio project*

2) Provide the following information: Application name, Project location and Package name of application and click next.
3) Select the API level of application and click next.
4) Select the Activity type (Empty Activity).
5) Provide the Activity Name and click finish.
6) After finishing the Activity configuration, Android Studio auto generates the activity class and other required configuration files.

Now an android project has been created. You can explore the android project and see the simple program, it looks like this:

2) Write the message

File: activity_main.xml

Android studio auto generates code for activity_main.xml file. You may edit this file according to your requirement.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="first.javatpoint.com.welcome.MainActivity">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello Android!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
}
```

File: MainActivity.java

```
package first.javatpoint.com.welcome;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity
    {
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

3) Run the android application.

To run the android application, click the run icon on the toolbar or simply press Shift + F10.

The android emulator might take 2 or 3 minutes to boot. So please have patience. After booting the emulator, the android studio installs the application and launches the activity.

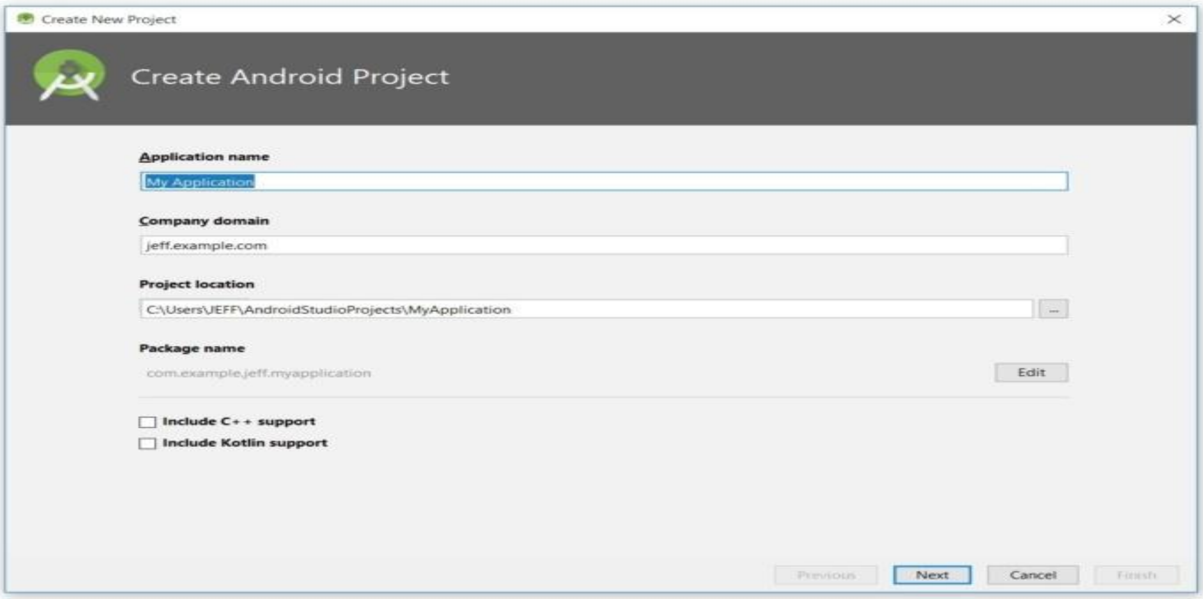
Creating and starting an activity

The quickest way to get to know Android Studio is to use it to develop an app. We'll start with a variation on the "Hello, World" application: a little mobile app that displays a "Welcome to Android" message.

In the steps that follow, you'll start a new Android Studio project and get to know the main window, including the editor window that you'll use to code the app in Part 2.

Starting a new project:

From our setup so far, you should still have Android Studio running with the **Welcome to Android Studio** dialog box. From here, click **Start a new Android Studio project**. Android Studio will respond with


Create New Project

Create Android Project

Application name
My Application

Company domain
jeff.example.com

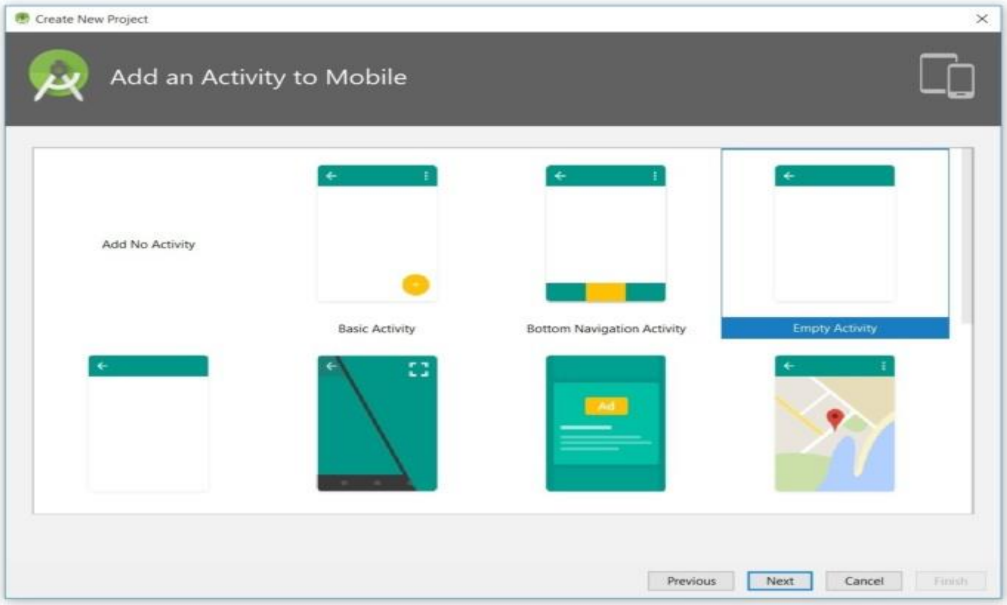
Project location
C:\Users\JEFF\AndroidStudioProjects\MyApplication

Package name
com.example.jeff.myapplication

☐ Include C++ support
☐ Include Kotlin support

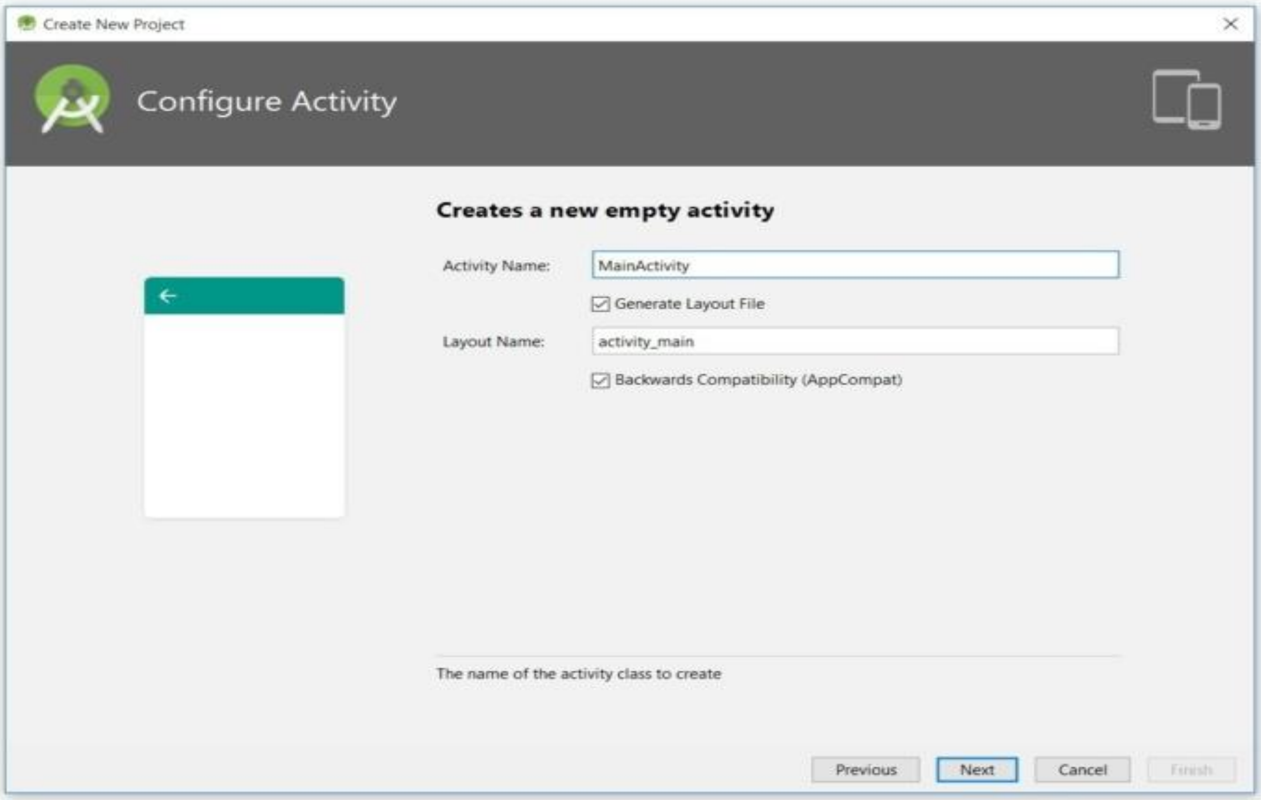
the **Create New Project** dialog box.

Create a new Android project (click to enlarge)



Click **Next**, and you will be given the opportunity to choose a template for your app's main activity. For now we'll stick with **Empty Activity**. Select this template (if necessary) and click **Next**.

Next you'll customize the activity:


Create New Project

Configure Activity

Creates a new empty activity

Activity Name: MainActivity

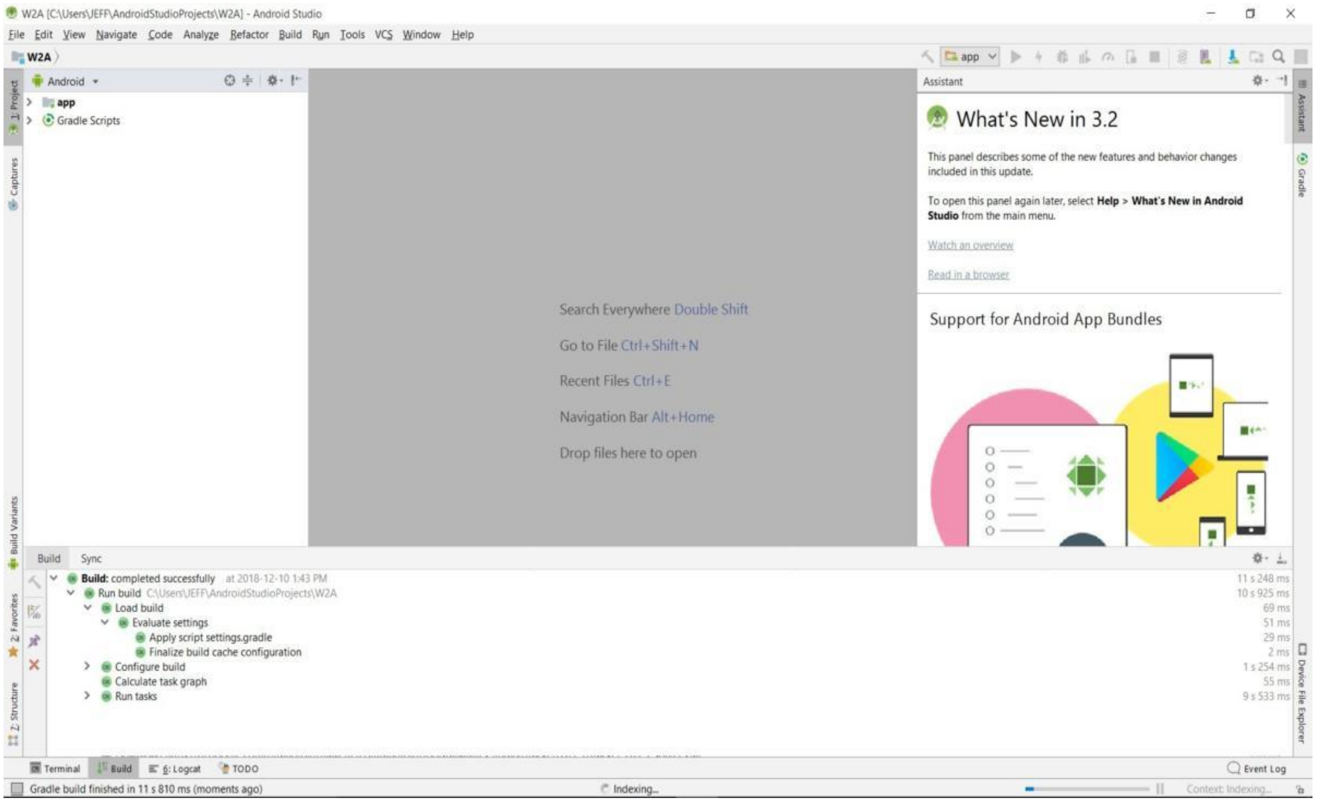
☑ Generate Layout File

Layout Name: activity_main

☑ Backwards Compatibility (AppCompat)

The name of the activity class to create

Android Studio enables **Finish**. Click this button and Android Studio takes you to the main window.


W2A [C:\Users\JEFF\AndroidStudioProjects\W2A] - Android Studio

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
W2A
Android
app
Gradle Scripts
app
Assistant

Search Everywhere Double Shift

Go to File Ctrl+Shift+N

Recent Files Ctrl+E

Navigation Bar Alt+Home

Drop files here to open

What's New in 3.2

This panel describes some of the new features and behavior changes included in this update.

To open this panel again later, select **Help > What's New in Android Studio** from the main menu.

Watch an overview

Read in a browser

Support for Android App Bundles

```
Build: completed successfully  at 2018-12-10 1:43 PM          11 s 248 ms
  Run build C:\Users\JEFF\AndroidStudioProjects\W2A            10 s 925 ms
    Load build                                                      69 ms
      Evaluate settings                                             51 ms
        Apply script settings.gradle                                29 ms
        Finalize build cache configuration                           2 ms
    Configure build                                           1 s 254 ms
    Calculate task graph                                            55 ms
    Run tasks                                                 9 s 533 ms
```

Gradle build finished in 11 s 810 ms (moments ago)

Indexing...

. The main window is divided into a menu bar and several other areas,

Accessing AVD Manager and SDK Manager

To access the traditional AVD Manager or SDK Manager, select **AVD Manager** or **SDK Manager** from Android Studio's **Tools** menu.

Q: Explain Process Management in OS?

A Program does nothing unless its instructions are executed by a CPU. 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 process 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

1. Scheduling processes and threads on the CPUs.
2. Creating and deleting both user and system processes.
3. Suspending and resuming processes.
4. Providing mechanisms for process synchronization.
5. Providing mechanisms for process communication.

Q: Explain Android File System?

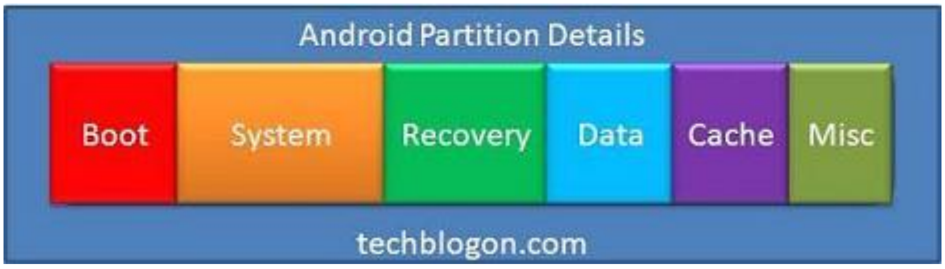
There are mainly 6 partitions in Android phones, tablets and other Android devices.

Below is the list of partition for Android File System.

Note that there might be some other partitions available, it differs from Model to Model.

But logically below 6 partitions can be found in any Android devices.

- **/boot**
- **/system**
- **/recovery**
- **/data**
- **/cache**
- **/misc**



Also Below are the for SD Card Fie System Partitions.

- **/sdcard**
- **/sd-ext**

Please Note:

Only /sdcard partition can be found in all Android devices and the rest are present only in select devices.

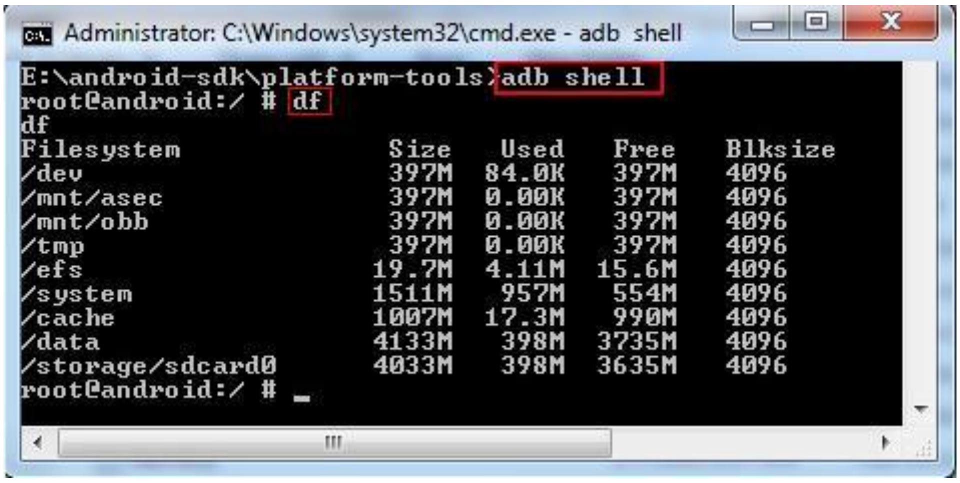
Know Your Android Device Partition Size using adb Command

You can know which partitions are available along with the partition size for all partition in your android device.

Go through the below image and run the adb command as shown in that image.

For more adb commands, you can read my atricle **Useful adb Commands for Android Development.**

Also for more details for android architecture, you can read my article **Android Architecture**.



```
E:\android-sdk\platform-tools>adb shell
root@android:/ # df
df
Filesystem             Size    Used    Free    Blksize
/dev                   397M   84.0K    397M    4096
/mnt/asec              397M   0.00K    397M    4096
/mnt/obb               397M   0.00K    397M    4096
/tmp                   397M   0.00K    397M    4096
/efs                  19.7M   4.11M   15.6M    4096
/system               1511M    957M    554M    4096
/cache                1007M   17.3M    990M    4096
/data                 4133M    398M   3735M    4096
/storage/sdcard0      4033M    398M   3635M    4096
root@android:/ # _
```

Note:

boot and recovery partition is not displayed in the above image.

So after adb shell, you need to run mount command. like

```
cat proc/mounts
```

/boot

- This is the boot partition of your Android device, as the name suggests.
- It includes the android kernel and the ramdisk.
- The device will not boot without this partition.
- Wiping this partition from recovery should only be done if absolutely required and once done, the device must NOT be rebooted before installing a new one, which can be done by installing a ROM that includes a /boot partition.

/system

- As the name suggests, this partition contains the entire Android OS, other than the kernel and the ramdisk.
- This includes the Android GUI and all the system applications that come pre-installed on the device.
- Wiping this partition will remove Android from the device without rendering it unbootable, and you will still be able to put the phone into recovery or bootloader mode to install a new ROM.

/recovery

- This is specially designed for backup.
- The recovery partition can be considered as an alternative boot partition, that lets the device boot into a recovery console for performing advanced recovery and maintenance operations on it.

/data

- Again as the name suggest, it is called userdata partition.
- This partition contains the user's data like your contacts, sms, settings and all android applications that you have installed.
- While you perform factory reset on your device, this partition will be wiped out, Then your device will be in the state, when you used for the first time or the way it was after the last official or custom ROM installation.

/cache

- I hope you have some idea about cache, as you are expert on internet browsing.
- This is the partition where Android stores frequently accessed data and app components.
- Wiping the cache doesn't effect your personal data but simply gets rid of the existing data there, which gets automatically rebuilt as you continue using the device.

/misc

- This partition contains miscellaneous system settings in form of on/off switches.
- These settings may include CID (Carrier or Region ID), USB configuration and certain hardware settingsetc.
- This is an important partition and if it is corrupt or missing, several of the device's features will will not function normally.

/sdcard

- This is not a partition on the internal memory of the device but rather the SD card.
- In terms of usage, this is your storage space to use as you see fit, to store your media, documents, ROMs etc. on it.
- Wiping it is perfectly safe as long as you backup all the data you require from it, to your computer first.
- Though several user-installed apps save their data and settings on the SD card and wiping this partition will make you lose all that data.
- On devices with both an internal and an external SD card – devices like the Samsung Galaxy S and severaltablets –
  the /sdcard partition is always used to refer to the internal SD card.

/sd-ext

- This is not a standard Android partition, but has become popular in the custom ROM scene.
- It is basically an additional partition on your SD card that acts as the /data partition when used with certain ROMs that have special features called APP2SD+ or data2ext enabled.
- It is especially useful on devices with little internal memory allotted to the /data partition.
- Thus, users who want to install more programs than the internal memory allows can make this partition and use it with a custom ROM that supports this feature, to get additional storage for installing their apps.