OPERATING SYSTEM NOTES

SYLLABUS

UNIT-1

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

UNIT- II

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

UNIT III

Process Management: Deadlock, Deadlock Characterization, Necessary and Sufficient Conditions for Deadlock, Deadlock Handling Approaches: Deadlock Prevention, Deadlock Avoidance and Deadlock Detection and Recovery. Concurrent and Dependent Processes, Critical Section, Semaphores, Methods for Interprocess Communication; Process Synchronization, Classical Process Synchronization Problems: Producer-Consumer, Reader-Writer.

UNIT IV

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

UNIT V

File and I/O Management, OS security: Directory Structure, 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.

UNIT-1

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

1.What is Operating System?

An operating system is a program on which application programs are executed and acts as an communication bridge (interface) between the user and the computer hardware. The main task an operating system carries out is the allocation of resources and services, such as allocation of: memory, devices, processors and information. The operating system also includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.

2.History of OS ?

History of operating system:
operating system has been evolving through the years

Generation	Year	Electronic devices	Types of os and devices
First	1945-55	Vaccum tubes	Plug boards
Second	1955-1965	Transistors	Batch systems
Third	1965-1980	Integrated circuit	Multiprogramming
Fourth	Since-1980	Large scale integration	PC

3. write about functions of an operating System?

Important functions of an operating System:

1. Security: – The operating system uses password protection to protect user data and similar other techniques. it also prevents unauthorized access to programs and user data.

2. Control over system performance – Monitors overall system health to help improve performance. records the response time between service requests and system response to have a complete view of the system health. This can help improve performance by providing important information needed to troubleshoot problems.

3. Job accounting– Operating system Keeps track of time and resources used by various tasks and users, this information can be used to track resource usage for a particular user or group of user.

4. Error detecting aids – Operating system constantly monitors the system to detect errors and avoid the malfunctioning of computer system.

5. Coordination between other software and users – Operating systems also coordinate and assign interpreters, compilers, assemblers and other software to the various users of the computer systems.

6. Memory Management – The operating system manages the Primary Memory or Main Memory. Main memory is made up of a large array of bytes or words where each byte or word is assigned a certain address. Main memory is a fast storage and it can be accessed directly by the CPU.

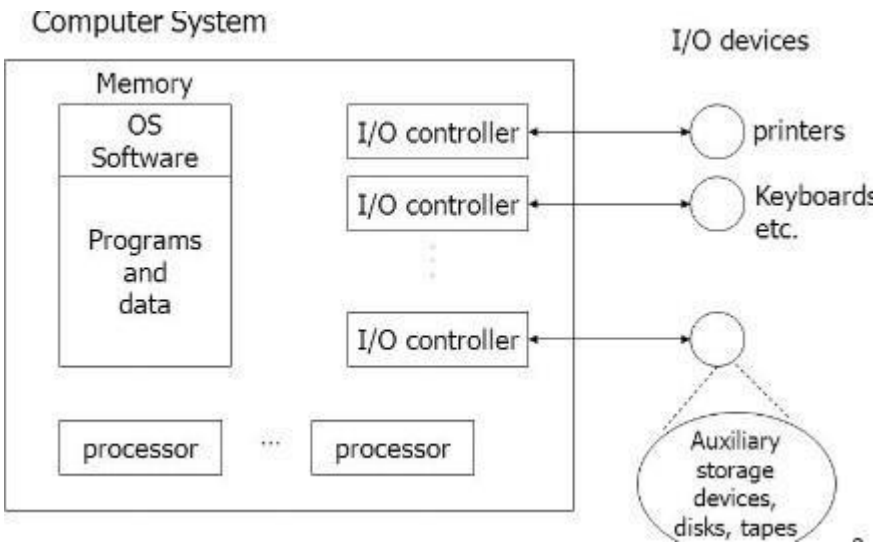
7. Processor Management – In a multi programming environment, the OS decides the order in which processes have access to the processor, and how much processing time each process has. This function of OS is called process scheduling. An Operating System performs the following activities for processor management.

8.Device Management – An OS manages device communication via their respective drivers. It performs the following activities for device management. Keeps tracks of all devices connected to system. designates a program responsible for every device known as the Input/Output controller.

9. File Management – A file system is organized into directories for efficient or easy navigation and usage. These directories may contain other directories and other files.

4. Explain how Operating system acts as resource manager?

Computer is a set of resource for the movement ,storage and processing of data for the control of these functions.The OS is responsible for managing these resources.



Main resources that are managed by the operating system. A portion of the operating system is in main memory.

•This includes the kernel, which contains the most frequently used functions in the operating system and other portions of the OS currently used.

•When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them.

•The operating system manages many different types of resources.

•Some (such as CPU cycles, main memory, and file storage) may have special allocation code, whereas others (such as I/O devices) may have much more general request and release code.

•For instance, in determining how best to use the CPU, operating systems have CPU-scheduling routines that take into account the speed of the CPU, the jobs that must be executed, the number of registers available, and other factors.

•There may also be routines to allocate printers, USB storage drives, and other peripheral devices.

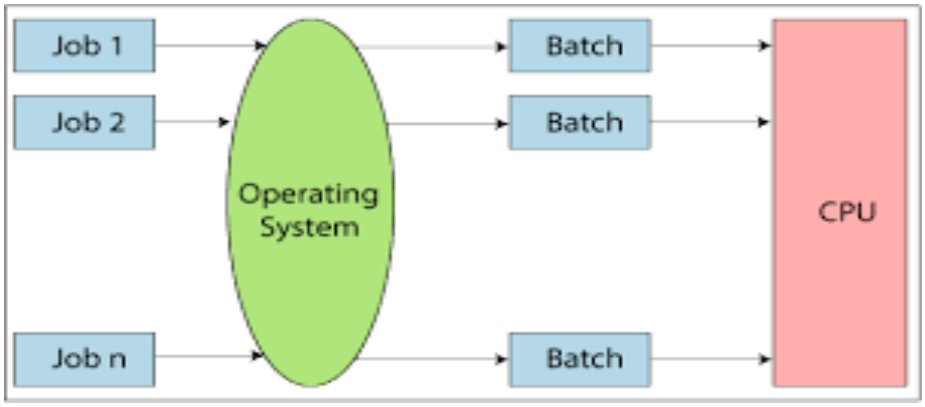
5. Explain about Types of operating systems?

Following are the popular types of OS (Operating System):

- Batch Operating System
- Multitasking/Time Sharing OS
- Multiprocessing OS
- Real Time OS
- Distributed OS
- Network OS
- Mobile OS

Batch Operating System:

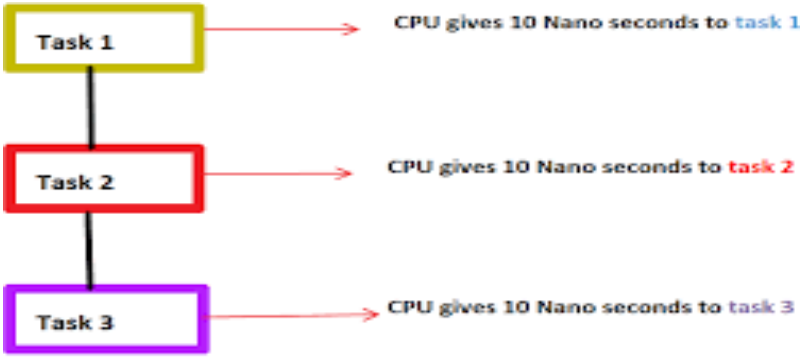
Some computer processes are very lengthy and time-consuming. To speed the same process, a job with a similar type of needs is batched together and run as a group.



The user of a batch operating system never directly interacts with the computer. In this type of OS, every user prepares his or her job on an offline device like a punch card and submit it to the computer operator.

Multi-Tasking/Time-sharing Operating systems

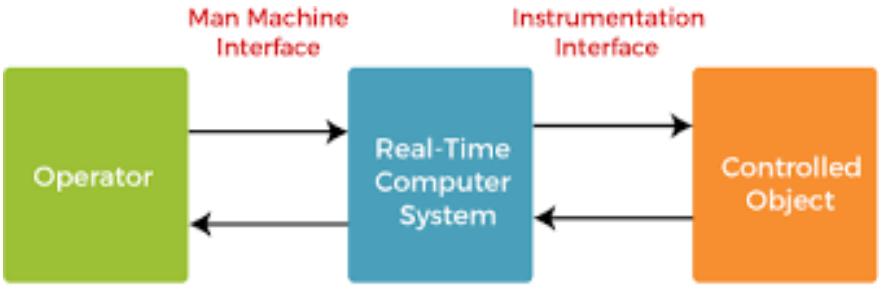
Time-sharing operating system enables people located at a different terminal(shell) to use a single computer system at the same time. The processor time (CPU) which is shared among multiple users is termed as time sharing.



Multi-Tasking In Operating System

Real time OS

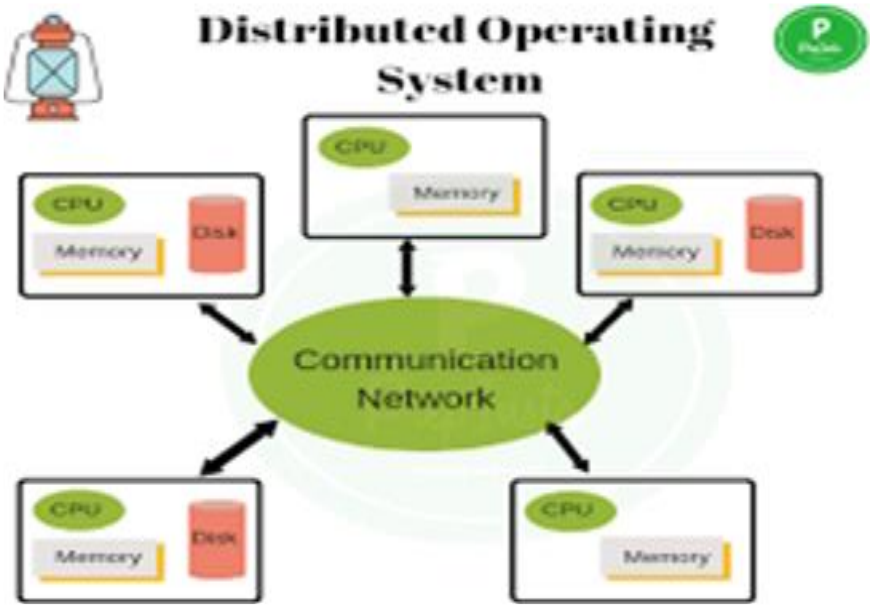
A real time operating system time interval to process and respond to inputs is very small. Examples: Military Software Systems, Space Software Systems are the Real time OS example.



Real-Time System

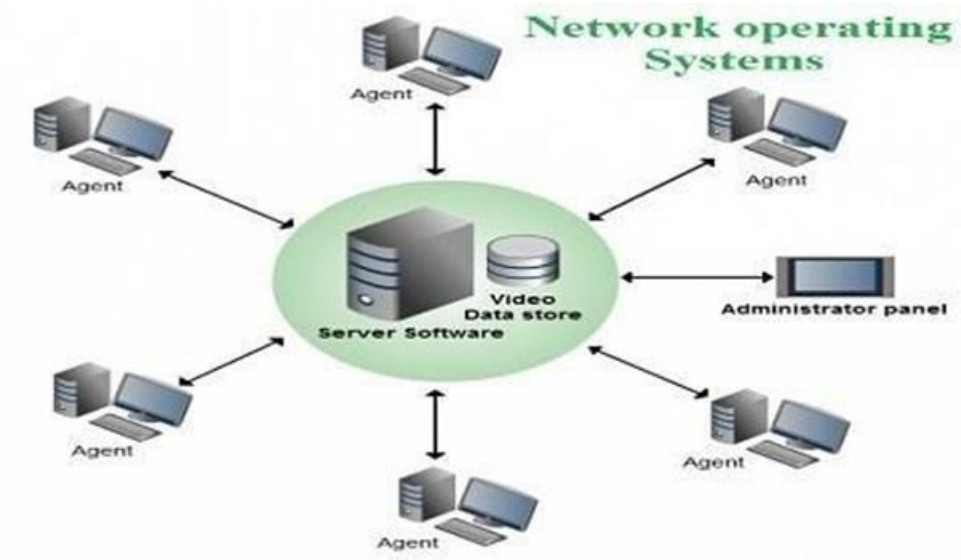
Distributed Operating System

Distributed systems use many processors located in different machines to provide very fast computation to its users.



Network Operating System

Network Operating System runs on a server. It provides the capability to serve to manage data, user, groups, security, application, and other networking functions.



Mobile OS

Mobile operating systems are those OS which is especially that are designed to power smartphones, tablets, and wearables devices.

UNIT- II

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

1.Explain about Computer-System Architecture? (or)What are various Types of processor Systems explain?

A computer system can be organized in a number of different ways, which we can categorize roughly according to the number of general-purpose processors used.

1.Single-processor system

2.Multiprocessor system

3.Clustered Systems:

1.Single-Processor Systems:

•On a single processor system, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes.
•Almost all single processor systems have other special-purpose processors as well.
•They may come in the form of device-specific processors, such as disk, keyboard, and graphics controllers; or, on mainframes.
•They may come in the form of more general-purpose processors, such as I/O processors that move data rapidly among the components of the system.
•All of these special-purpose processors run a limited instruction set and do not run user processes.
•If there is only one general-purpose CPU, then the system is a single- processor system.

2.Multiprocessor Systems

•In multiprocessor system, two or more processors work together. In this system, multiple programs (more than one program) are executed on different processors at the same time.
•This type of processing is known as multiprocessing. Some operating systems have features of multiprocessing.
•UNIX is an example of multiprocessing operating system. Some versions of Microsoft Windows also support multiprocessing.
•Multiprocessor system is also known as parallel system.

Types of Multiprocessor Systems:
The multiprocessor systems are further divided into two types;
(i). Asymmetric multiprocessing system
(ii). Symmetric multiprocessing system

(i)Asymmetric Multiprocessing System (AMS):
•The multiprocessing system, in which each processor is assigned a specific task, is known as Asymmetric Multiprocessing System.
•In this system, one processor works as master processor, while other processors work as slave processors.
•The master processor controls the operations of system.
•It also schedules and distributes tasks among the slave processors.
•The slave processors perform the predefined tasks.
(ii)Symmetric Multiprocessing System (SMP):

•The multiprocessing system, in which multiple processors work together on the same task, is known as Symmetric Multiprocessing System.
•In this system, each processor can perform all types of tasks.
•All processors are treated equally and no master-slave relationship exists between the processors.

3.Clustered Systems:

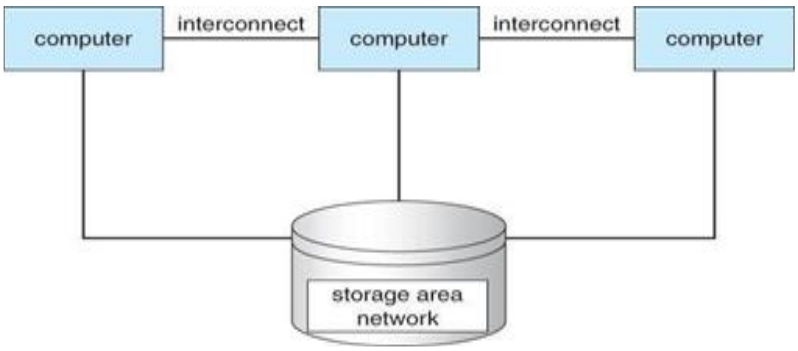
•Clustered system is another form of multiprocessor system.
•This system also contains multiple processors but it differs from multiprocessor system.
•The clustered system consists of two or more individual systems that are coupled together.
•In clustered system, individual systems (or clustered computers) share the same storage and are linked together ,via Local Area Network (LAN).

Types of Clustered Systems:

Like multiprocessor systems, clustered system can also be of two types
(i). Asymmetric Clustered System
(ii). Symmetric Clustered System

Asymmetric Clustered System:

•In asymmetric clustered system, one machine is in hot-standby mode while the other machine is running the application.
•The hot-standby host machine does nothing. It only monitors the active server.
•If the server fails, the hot-standby machine becomes the active server.
Symmetric Clustered System:
•In symmetric clustered system, multiple hosts (machines) run the applications.
•They also monitor each other.
•This mode is more efficient than asymmetric system, because it uses all the available hardware.



2.Define System calls Explain types of system calls?

•System calls provide an interface to the services made available by an operating system.
•These calls are generally available as routines written in C and C++.
•Application developers design programs according to an application programming interface (API).
•The API specifies a set of functions that are available to an application programmer, including the parameters that are passed to each function and the return values the programmer can expect.

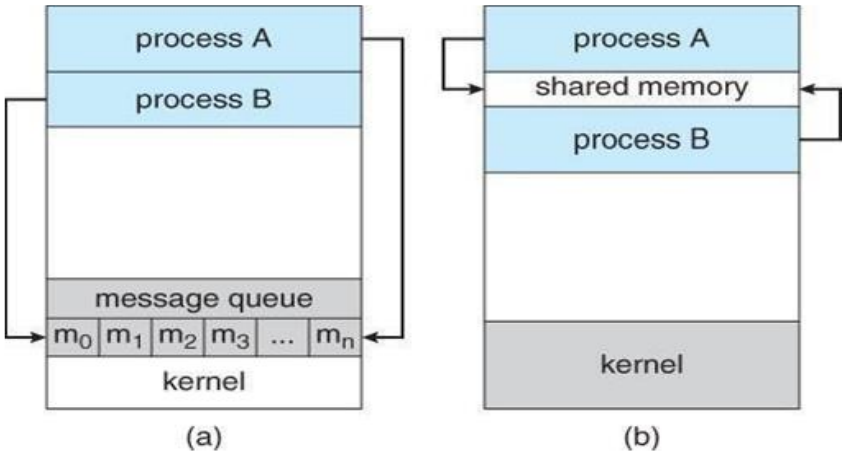
Types of System Calls

System calls can be grouped roughly into five major categories:
•Process control.
•File manipulation.
•Device manipulation.
•Information maintenance.
•Communications, and protection

Process control:

A running program needs to be able to halt its execution either normally (end())or abnormally (abort()).

File Management:
•We first need to be able to create() and delete() files.
•Once the file is created, we need to open() it and to use it.
•We may also read(), write(), or reposition() (rewind or skip to the end of the file, for example).
•Finally, we need to close() the file, indicating that we are no longer using it.
Device Management :
•A process may need several resources to execute—main memory, diskdrives, access to files, and so on.
•If the resources are available, they can be granted, and control can be returned to the user process.
•Otherwise, the process will have to wait until sufficient resources are available.
Information Maintenance:
•Most systems have a system call to return the current time() and date().
•Other system calls may return information about the system, such as the number of current users, the version number of the operating system, the amount of free memory or disk space, and so on.
Communication:
•There are two common models of interprocess communication:
•The messagepassing model and the shared-memory model.
•In the message-passing model, the communicating processes exchange messages with one another to transfer information.



•Messages can be exchanged between the processes either directly or indirectly through a common mailbox.
•In the shared-memory model, processes use shared memory create() and shared memory attach() system calls to create and gain access to regions of memory owned by other processes.
Protection :
•provides a mechanism for controlling access to the resources provided by a computer system.
•protection include set permission() andget permission(), system calls.

3.Explain System View of the Process and Resources?
According to the computer system, the operating system is the bridge between applications and hardware. It is most intimate with the hardware and is used to control it as required.
The different types of system view for operating system can be explained as follows:
•The system views the operating system as a resource allocator. There are many resources such as CPU time, memory space, file storage space, I/O devices etc. that are required by processes for execution. It is the duty of the operating system to allocate these resources judiciously to the processes so that the computer system can run as smoothly as possible.
•The operating system can also work as a control program. It manages all the processes and I/O devices so that the computer system works smoothly and there are no errors. It makes sure that the I/O devices work in a proper manner without creating problems.
•Operating systems can also be viewed as a way to make using hardware easier.
•Computers were required to easily solve user problems. However it is not easy to work directly with the computer hardware. So, operating systems were developed to easily communicate with the hardware.

•An operating system can also be considered as a program running at all times in the background of a computer system (known as the kernel) and handling all the application programs. This is the definition of the operating system that is generally followed.

4.Explain about process abstraction?

A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

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 computer program.

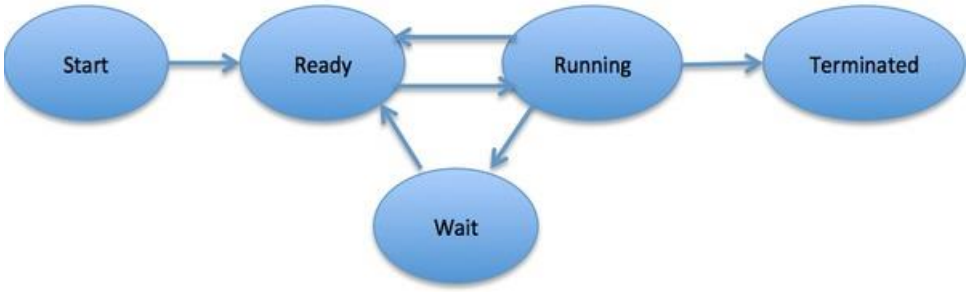
A part of a computer program that performs a well-defined task is known as an algorithm. A collection of computer programs, libraries and related data are referred to as a software.

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.

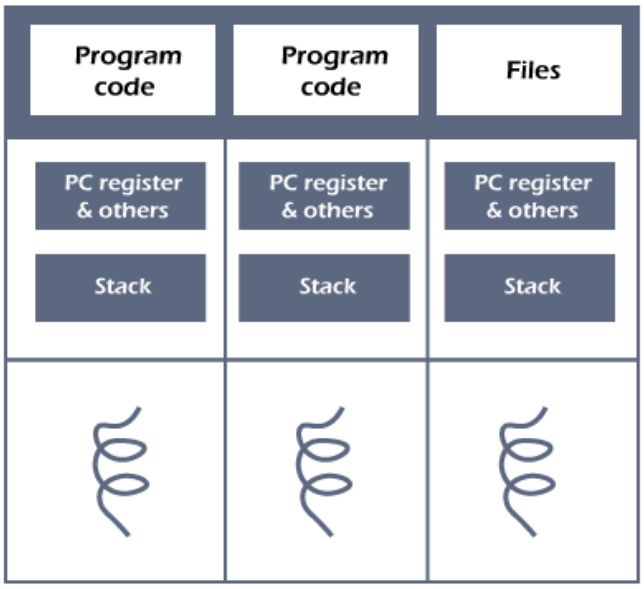


6. Explain about THREADS? (Thread state-muiti & single threading-Types of threads-threading models) ?

Or

Define Thread ?Explain about types of threads ?

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



Three threads of same process

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

Types of Threads

In the operating system, there are two types of threads.

1.Kernel level thread.

2.User-level thread.

User-level thread:

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

Advantages of User-level threads

1.The user threads can be easily implemented than the kernel thread.

2.User-level threads can be applied to such types of operating systems that do not support threads at the kernel-level.

3.It is faster and efficient.

4.Context switch time is shorter than the kernel-level threads.

5.It does not require modifications of the operating system.

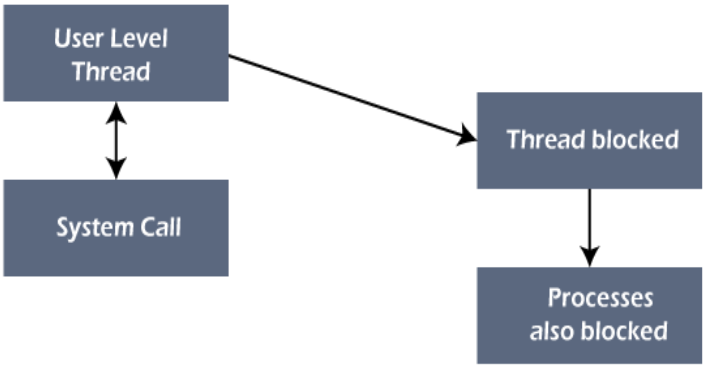
6.User-level threads representation is very simple. The register, PC, stack, and mini thread control blocks are stored in the address space of the user-level process.

7.It is simple to create, switch, and synchronize threads without the intervention of the process.

Disadvantages of User-level threads

1.User-level threads lack coordination between the thread and the kernel.

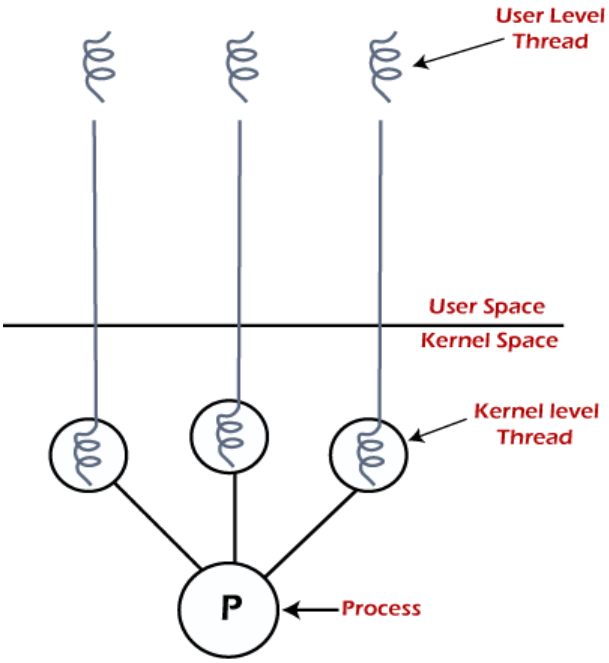
2.If a thread causes a page fault, the entire process is blocked



.

Kernel level thread:

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



Advantages of Kernel-level threads

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

Disadvantages of Kernel-level threads

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

7.difference between user level and kernel level?

Sr.No	User level thread	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.

8.Write about Thread Issues ?

1.The fork() and exec() System Calls
2.Signal Handling
3.Thread Cancellation
4.Signal cancellation
5.Thread Specific data.

1.The fork() and exec() System Call:

•The semantics of the fork() and exec() system calls change in a multithreaded program.
•If one thread in a program calls fork(), does the new process duplicate all threads, or is the new process single-threaded
•Some UNIX systems have chosen to have two versions of fork(), one that duplicates all threads and another that duplicates only the thread that invoked the fork() system call.
•if a thread invokes the exec() system call, the program specified in the parameter to exec() will replace the entire process—including all threads.

2.Signal Handling:

A signal is used in UNIX systems to notify a process that a particular event has occurred.
A signal may be received either synchronously or asynchronously, depending on the source of and the reason for the event being signaled.
All signals, whether synchronous or asynchronous, follow the same pattern
1.A signal is generated by the occurrence of a particular event.
2.The signal is delivered to a process.
3.Once delivered, the signal must be handled

3.Thread Cancellation

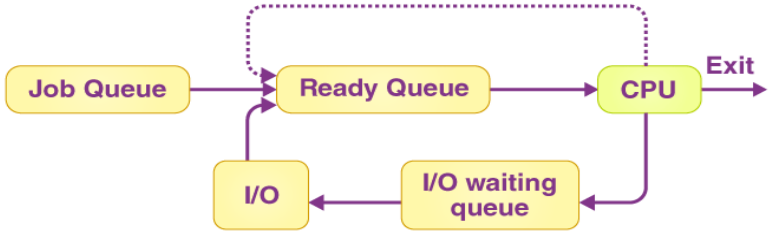
Thread cancellation involves terminating a thread before it has completed A thread that is to be canceled is often referred to as the target thread. Cancellation of a target thread may occur in two different scenarios:
1.Asynchronous cancellation: One thread immediately terminates the target thread.
2.Deferred cancellation: The target thread periodically checks whether it should terminate, allowing it an opportunity to terminate itself in an orderly fashion.

4.Signal Cancellation:

A signal is used in UNIX systems to notofy a process that a particular event has occurred.
A signal may be synchronous (or)Asynchronously received.
A signal may be handled by
a) default signal handler: It is run by the kernel when handling that signal.
b) User-defined Signal handler: It is called by the user to handle the signal.

5.Thread Pools:
Thread pool is to create a number of thread process start up and place them into a pool.

9.Discuss about process scheduling?
Definition:
The process manager's activity is process scheduling, which involves removing the running process from the CPU and selecting another process based on a specific strategy.
Process Scheduling Queues:
All PCBs (Process Scheduling Blocks) are kept in Process Scheduling Queues by the OS. Each processing state has its own queue in the OS, and PCBs from all processes in the same execution state are put in the very same queue. A process's PCB is unlinked from its present queue and then moved to its next state queue when its status changes.
The following major process scheduling queues are maintained by the Operating System:
Job queue – It contains all of the system's processes.
Ready queue – This queue maintains a list of all processes in the main memory that are ready to run. This queue is always filled with new processes.
Device queue – This queue is made up of processes that are stalled owing to the lack of an I/O device.



Each queue can be managed by the OS using distinct policies (FIFO, Priority, Round Robin, etc.). The OS scheduler governs how tasks are moved between the ready and run queues, each of which can only have one item per processor core on a system; it has been integrated with the CPU.
Types of Schedulers:
Schedulers are specialised computer programmes that manage process scheduling in a variety of ways. Their primary responsibility is to choose which jobs to submit into the system and which processes to run. Click here to learn more on Process Schedulers in Operating System. There are three types of schedulers:
Long-Term
A job scheduler is another name for it. A long-term scheduler determines which programs are accepted for processing into the system. It picks processes from the queue and then loads them into memory so they can be executed. For CPU scheduling, a process loads into memory.
Short-Term
CPU scheduler is another name for it. Its major goal is to improve system performance according to the set of criteria defined. It refers to the transition from the process's ready state to the running stage. The CPU scheduler happens to choose a process from those that are ready to run and then allocates the CPU to it.
Medium-Term
It includes medium-term scheduling. Swapping clears the memory of the processes. The degree of multiprogramming is reduced. The switched-out processes are handled by the medium-term scheduler.
Context Switch
A context switch is a mechanism in the Process Control block that stores and restores the state or context of a CPU so that a process execution can be resumed from the very same point at a later time. A context switcher makes use of this technique to allow numerous programs to share a single CPU. Thus, context switching is one of the most important elements of a multitasking operating system.

10.Explain about C.P.U Scheduling Algorithms ?

Scheduling algorithms may be preemptive and non preemptive.

The scheduling algorithms are

1.First come first served Method (FCFS)(non preemptive)

2.Shorthest Job first(SJF) (preemptive or non preemptive)

3.Priority(preemptive or non preemptive)

4. Round robin algorithm(preemptive)

1.First-Come, First-Served Scheduling:

The simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this scheme, the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue.

Example: Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P_1	24
P_2	3
P_3	3

If the processes arrive in the order P1, P2, P3, and are served in FCFS order.

we get the result shown in the following Gantt chart, which is a bar chart that illustrates a particular schedule, including the start and finish times of each of the participating processes.

The waiting time is 0 milliseconds for process P1,24 milliseconds for process P2, and 27 milliseconds for process P3.

Thus, the average waiting time is (0 + 24 + 27)/3 = 17 milliseconds.

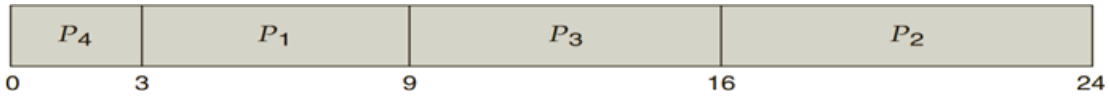
2.Shortest-Job-First Scheduling :(preemptive Scheduling)

A different approach to CPU scheduling is the shortest-job-first(SJF) scheduling algorithm.

Ex: consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Using SJF scheduling, we would schedule these processes according to thefollowing Gantt chart:



The waiting time is 3 milliseconds for process P1,16 milliseconds for process P2, 9 milliseconds for process P3, and0 milliseconds for process P4.

Thus, the average waiting time is (3 + 16+9+ 0)/4 = 7 milliseconds.

3.Priority Scheduling:

The SJF algorithm is a special case of the general priority-scheduling algorithm. Apriority is associated with each process, and the CPU is allocated to the process with the highest priority.

Equal-priority processes are scheduled in FCFS order.
The larger the CPU burst, the lower the priority, and vice versa.
Example:
Consider the following set of processes, assumed to have arrived at time 0 in the order P1, P2, ⋯, P5, with the length of the CPU burst given in milliseconds:

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

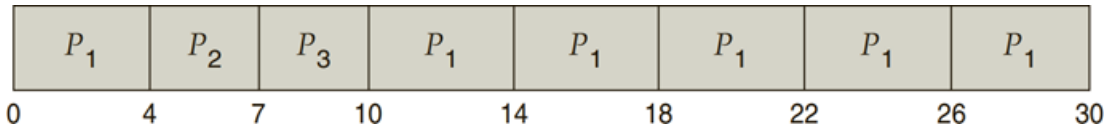
Using priority scheduling, we would schedule these processes according to the following Gantt chart:

The average waiting time is 40.2 milliseconds.

4.Round-Robin Scheduling :
The round-robin (RR) scheduling algorithm is designed especially for timesharing systems. It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes. A small unit of time, called a time quantum or time slice, is defined. A time quantum is generally from 10 to 100 milliseconds in length.
EX: Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P_1	24
P_2	3
P_3	3

If we use a time quantum of 4 milliseconds, then process P1 gets the first 4 milliseconds.



Let's calculate the average waiting time for this schedule.P1 waits for 6 milliseconds (10 - 4),
P2 waits for 4 milliseconds, and P3 waits for 7 milliseconds.
Thus, the average waiting time is 17/3 = 5.66 milliseconds.

UNIT III

Process Management: Deadlock, Deadlock Characterization, Necessary and Sufficient Conditions for Deadlock, Deadlock Handling Approaches: Deadlock Prevention, Deadlock Avoidance and Deadlock Detection and Recovery. Concurrent and Dependent Processes, Critical Section, Semaphores, Methods for Interprocess Communication; Process Synchronization, Classical Process Synchronization Problems: Producer-Consumer, Reader-Writer.

1.Define Deadlock?

Deadlock is a situation where each of the computer process waits for a resource which is being assigned to some another process. In this situation, none of the process gets executed since the resource it needs, is held by some other process which is also waiting for some other resource to be released.

2.What are Necessary conditions for Deadlocks? (Or) Explain about deadlock characteristics?

there are four necessary conditions for deadlock

1.Mutual Exclusion

A resource can only be shared in mutually exclusive manner. It implies, if two process cannot use the same resource at the same time.

2.Hold and Wait

A process waits for some resources while holding another resource at the same time.

3.No preemption

The process which once scheduled will be executed till the completion. No other process can be scheduled by the scheduler meanwhile.

4.Circular Wait

All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

3.Explain about deadlock approaches?

There are four deadlock approaches

1.Deadlock Ignorance

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

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

2.Deadlock prevention

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

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

3.Deadlock avoidance

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

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

4. Deadlock detection and recovery

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

3. Explain about Interprocess Communication and methods in IPC?

Processes executing concurrently in the operating system may be either independent processes or cooperating processes. A process is independent if it cannot affect or be affected by the other processes executing in the system. Any process that does not share data with any other process is independent. A process is cooperating if it can affect or be affected by the other processes executing in the system. There are several reasons for providing an environment that allows process cooperation:

METHODS:

1.Pipes (Same Process) –
This allows flow of data in one direction only. Analogous to simplex systems (Keyboard). Data from the output is usually buffered until input process receives it which must have a common origin.

2.Names Pipes (Different Processes) –
This is a pipe with a specific name it can be used in processes that don't have a shared common process origin. E.g. is FIFO where the details written to a pipe is first named.

3.Message Queuing –
This allows messages to be passed between processes using either a single queue or several message queue. This is managed by system kernel these messages are coordinated using an API.

4.Semaphores –
This is used in solving problems associated with synchronization and to avoid race condition. These are integer values which are greater than or equal to 0.

5.Shared memory –
This allows the interchange of data through a defined area of memory. Semaphore values have to be obtained before data can get access to shared memory.

6.Sockets –
This method is mostly used to communicate over a network between a client and a server. It allows for a standard connection which is computer and OS independent.

4.What is semaphore? Explain about semaphore and its types?

Semaphores refer to the integer variables that are primarily used to solve the critical section problem via combining two of the atomic procedures, wait and signal, for the process synchronization.

The definitions of signal and wait are given below:

1.Wait

It decrements the value of it's a argument in case it is positive. In case S is zero or negative, then no operation would be performed.

```
Wait(A)
{
While (A<=0);
A–;
}
```

2.Signal

This operation increments the actual value of its argument A.

```
Signal(A)
{
A++;
}
```

Types of Semaphores

Semaphores are of two types:

Binary Semaphore

Mutex lock is another name for binary Semaphore. It can only have two possible values: 0 and 1, and its value is set to 1 by default. It's used to implement numerous processes to solve critical section problems. Read more on Binary Semaphore here.

Counting Semaphore

Counting Semaphore's worth can be found anywhere in the world. It's used to restrict access to a resource with many copies. Read more on Counting Semaphore here.

Pros of Semaphores
Given below are some of the pros of semaphores:

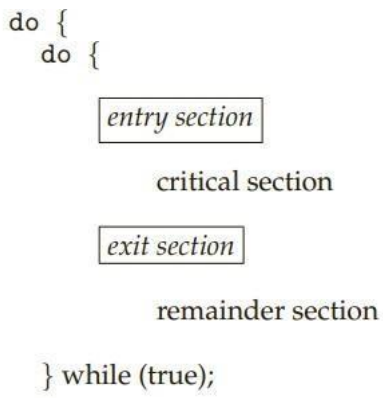
Only one process is allowed to enter the critical part because of semaphores. They adhere closely to the mutual exclusion principle. Also, they are far more efficient as compared to the other synchronization methods.
Since the processor time is not wasted in checking whether a condition is met so as to allow a process to access its critical section, there is ultimately no resource wastage due to busy waiting in the semaphores.
Semaphores are implemented in the microkernel's machine-independent code. They are, therefore, machine-independent.

Cons of Semaphores
The following are some of the downsides of semaphores:

Due to how complex semaphores are, the wait and signal actions must be implemented in the proper order to avoid deadlocks.
Semaphores are very impractical for their usage at the final scale since they reduce modularity. This mainly occurs because the wait, as well as the signal procedures, bar the system from forming an organised layout.
Semaphores can cause a priority inversion, with low-priority processes getting access to the important portion first and high-priority processes getting access afterwards.

5. Explain about process synchronization?
Synchronization
Message passing may be either blocking or non blocking— also known as synchronous and asynchronous.
•Blocking send : The sending processes blocked until the messages
Received y the receiving process or by the mailbox.
•Non blocking send: The sending process sends the message and resumes operation.
•Blocking receive: The receiver blocks until a message is available.
•Non blocking receive :The receiver retrieves either a valid message or a null
Buffering:
•Zero capacity. The queue has a maximum length of zero;
•Bounded capacity. The queue has finite length n;
•Unbounded capacity. The queue's length is potentially infinite;

6. Explain about critical section problem?
Each process has a segment code called a critical Section (cs).
•CS is used to avoid race condition.
•Almost only one process must be executed in critical section.
•When one process is executing in the cs ,no other process is to be allowed to enter into the cs.
•Each process has a segment of code, called a critical section, in which the process may be changing common variables, updating a table, writing a file, and so on
•The critical-section problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section.
•Structure of process (p) is as show in the fig,



Fig

Figure 6.1 General structure of a typical process P_i.

Each process contains three sections.
1.Entry section
2.Exit section
3.Remainder Section

Entry Section: The section of code implementing this request is the entry section. To check no other process to enter into CS when other process in CS.
Exit section: The critical section may be followed by an exit section.
Remainder Section: Instructions are not modified the shared data. The remaining code is the remainder section.

7. Explain about Classical Process Synchronization Problems?
Classic problem of synchronization are list below
1.Bounded –Buffer problem
2.Readers and writers problem

1.The Bounded-Buffer Problem:
The basic code for producer &consumer process is
Code for producer process:

```
while(true)
{
While(counter===buffer_size);
buffer[in]=nextproduced;
in=(in+1)%buffer_size;
counter++;
}
```

code for consumer Process:

```
while (true)
{
while (counter == 0) ;
/* do nothing */
next consumed = buffer[out];
out = (out + 1) % BUFFER SIZE;
counter--; /* consume the item in next consumed */

}
```

The producer and consumer processes share the following data structures:

```
int n;
semaphore mutex = 1;
semaphore empty = n;
semaphore full = 0
```

•We assume that the pool consists of n buffers, each capable of holding one item.
•The mutex semaphore provides mutual exclusion for accesses to the buffer pool and is initialized to the value 1.
•The empty and full semaphores count the number of empty and full buffers.
•The semaphore empty is initialized to the value n;
•The semaphore full is initialized to the value 0.
•A producer must not overwrite a full buffer,
•A consumer must not consume an empty buffer
•Producer and consumers must access buffers in a mutually exclusive manner.
We can interpret above code as the producer producing full buffers for the consumer or as the consumer producing empty buffers for the producer.

2.The Readers –Writers Problem
•Suppose that a database is to be shared among several concurrent processes.
•Some of these processes may want only to read the database, whereas others may want to update (that is, to read and write) the database.
•We distinguish between these two types of processes by referring to the former as readers and to the latter as writers.
•Obviously, if two readers access the shared data simultaneously, no adverse effects will result.
•However, if a writer and some other process (either a reader or a writer) access the database simultaneously, chaos may ensue.
•This synchronization problem is referred to as the readers –writers problem.
•The simplest one, referred to as the first readers –writers problem, no reader should wait for other readers to finish simply because a writer is waiting.
•The second readers – writers problem if a writer is waiting to access the object, no new readers may start reading.
•A solution to either problem may result in starvation.
•In the first case, writers may starve;
•In the second case, readers may starve.
•we present a solution to the first readers – writers problem.
•In the solution to the first readers –writers problem, the reader processes share the following data structures.

```
semaphore rw_mutex = 1;
semaphore mutex = 1;
int read_count = 0;
```

•The semaphores mutex and rw mutex are initialized to 1;
•read count is initialized to 0.
•The semaphore rw mutex is common to both reader and write processes.
•The mutex semaphore is used to ensure mutual exclusion when the variable read count is updated.
•The read count variable keeps track of how many processes are currently reading the object.
•The semaphore rw mutex functions as a mutual exclusion semaphore for the writers.
•It is also used by the first or last reader that enters or exits the critical section.
•It is not used by readers who enter or exit while other readers are in their critical sections.

UNIT IV

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

1.write about physical and virtual address space?

•An address generated by the CPU is commonly refereed as Logical Address, whereas the address seen by the memory unit that is one loaded into the memory address register of the memory is commonly refereed as the Physical Address.

•The compile time and load time address binding generates the identical logical and physical addresses.

•However, the execution time addresses binding scheme results in differing logical and physical addresses.

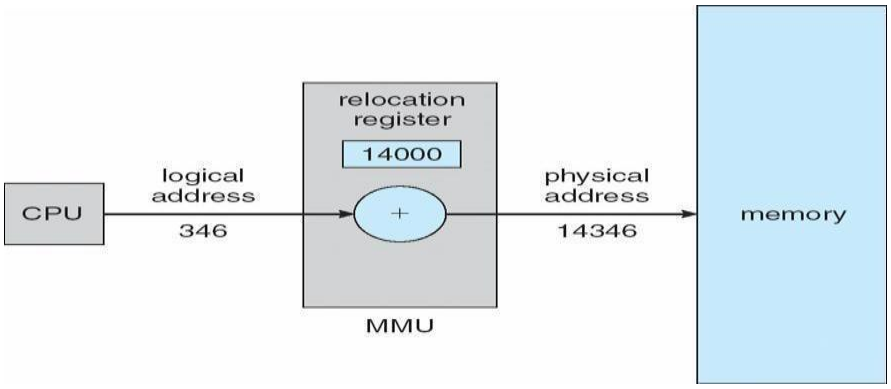
•The set of all logical addresses generated by a program is known as Logical Address Space, whereas the set of all physical addresses corresponding to these logical addresses is Physical Address Space.

•Now, the run time mapping from virtual address to physical address is done by a hardware device known as Memory Management Unit.

•Here in the case of mapping the base register is known as relocation register.

•The value in the relocation register is added to the address generated by a user process at the time it is sent to memory.

•The user program deals with logical addresses; it never sees the real physical addresses

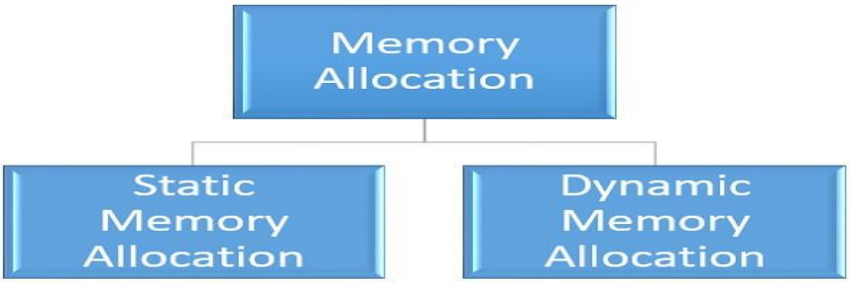


2.Explain about Memory allocation strategies?

Memory allocation is an action of assigning the physical or the virtual memory address space to a process (its instructions and data). The two fundamental methods of memory allocation are static and dynamic memory allocation.

The static memory allocation method assigns the memory to a process, before its execution. On the other hand, the dynamic memory allocation method assigns the memory to a process, during its execution.

have two types of memory allocation or we can say two methods of binding, static and dynamic binding.



Types of Memory Allocation

1. Static Memory Allocation

Static memory allocation is performed when the compiler compiles the program and generates object files. The linker merges all these object files and creates a single executable file. The loader loads this single

executable file in the main memory, for execution. In static memory allocation, the size of the data required by the process must be known before the execution of the process initiates.
If the data sizes are not known before the execution of the process, then they have to be guessed. If the data size guessed is larger than the required, then it leads to wastage of memory. If the guessed size is smaller, then it leads to inappropriate execution of the process.
The static memory allocation method does not need any memory allocation operation during the execution of the process. All the memory allocation operation required for the process is done before the execution of the process has started. So, it leads to faster execution of a process.
Static memory allocation provides more efficiency when compared to dynamic memory allocation.

2.. Dynamic Memory Allocation

Dynamic memory allocation is performed while the program is in execution. Here, the memory is allocated to the entities of the program when they are to be used for the first time while the program is running.
The actual size, of the data required, is known at the run time so, it allocates the exact memory space to the program thereby, reducing the memory wastage.

Dynamic memory allocation provides flexibility to the execution of the program. As it can decide what amount of memory space will be required by the program. If the program is large enough then a dynamic memory allocation is performed on the different parts of the program, which is to be used currently. This reduces memory wastage and improves the performance of the system.

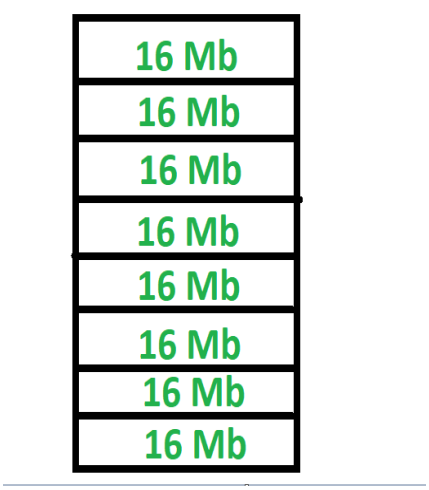
Allocating memory dynamically creates an overhead over the system. Some allocation operations are performed repeatedly during the program execution creating more overheads, leading in slow execution of the program.

Dynamic memory allocation does not require special support from the operating system. It is the responsibility of the programmer to design the program in a way to take advantage of dynamic memory allocation method.

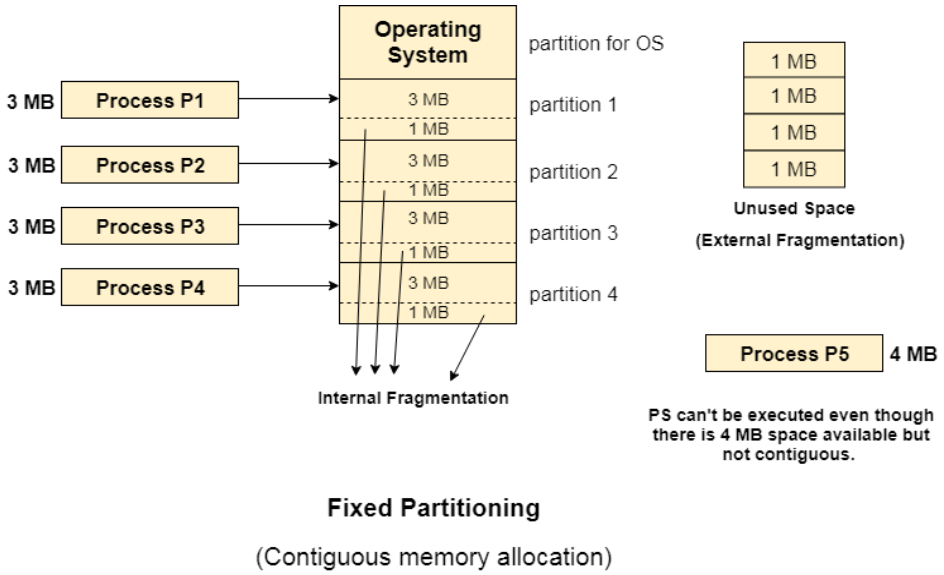
3.discuss about fixed and variable partitions in memory?

1. <u>Fixed Partitioning</u> :

Multi-programming with fixed partitioning is a contiguous memory management technique in which the main memory is divided into fixed sized partitions which can be of equal or unequal size. Whenever we have to allocate a process memory then a free partition that is big enough to hold the process is found. Then the memory is allocated to the process.If there is no free space available then the process waits in the queue to be allocated memory. It is one of the most oldest memory management technique which is easy to implement.



fixed partitioning,
1.The partitions cannot overlap.
2.A process must be contiguously present in a partition for the execution.



Fixed Partitioning

(Contiguous memory allocation)

There are various advantages of using this technique.

1. Internal Fragmentation

If the size of the process is lesser then the total size of the partition then some size of the partition get wasted and remain unused. This is wastage of the memory and called internal fragmentation.

As shown in the image below, the 4 MB partition is used to load only 3 MB process and the remaining 1 MB got wasted.

2. External Fragmentation

The total unused space of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form.

As shown in the image below, the remaining 1 MB space of each partition cannot be used as a unit to store a 4 MB process. Despite of the fact that the sufficient space is available to load the process, process will not be loaded.

3. Limitation on the size of the process

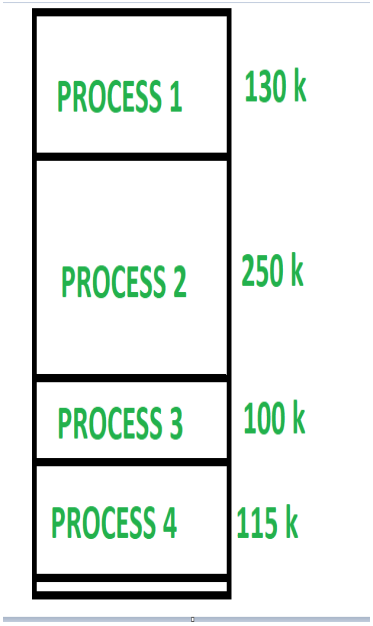
If the process size is larger than the size of maximum sized partition then that process cannot be loaded into the memory. Therefore, a limitation can be imposed on the process size that is it cannot be larger than the size of the largest partition.

4. Degree of multiprogramming is less

By Degree of multi programming, we simply mean the maximum number of processes that can be loaded into the memory at the same time. In fixed partitioning, the degree of multiprogramming is fixed and very less due to the fact that the size of the partition cannot be varied according to the size of processes.

2. <u>Variable Partitioning</u> :

Multi-programming with variable partitioning is a contiguous memory management technique in which the main memory is not divided into partitions and the process is allocated a chunk of free memory that is big enough for it to fit. The space which is left is considered as the free space which can be further used by other processes. It also provides the concept of compaction. In compaction the spaces that are free and the spaces which not allocated to the process are combined and single large memory space is made.



Dynamic partitioning tries to overcome the problems caused by fixed partitioning. In this technique, the partition size is not declared initially. It is declared at the time of process loading.
The first partition Is reserved for the operating system. The remaining space is divided into parts. The size of each partition will be equal to the size of the process. The partition size varies according to the need of the process so that the internal fragmentation can be avoided.

Advantages of Dynamic Partitioning over fixed partitioning

1.No Internal Fragmentation

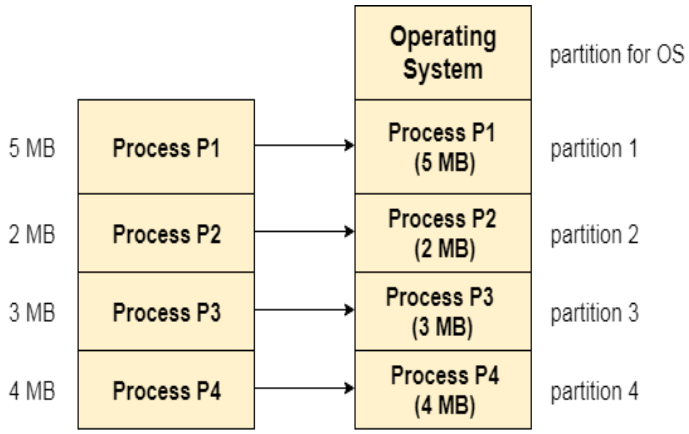
Given the fact that the partitions in dynamic partitioning are created according to the need of the process, It is clear that there will not be any internal fragmentation because there will not be any unused remaining space in the partition.

2.No Limitation on the size of the process

In Fixed partitioning, the process with the size greater than the size of the largest partition could not be executed due to the lack of sufficient contiguous memory. Here, In Dynamic partitioning, the process size can't be restricted since the partition size is decided according to the process size.

3.Degree of multiprogramming is dynamic

Due to the absence of internal fragmentation, there will not be any unused space in the partition hence more processes can be loaded in the memory at the same time.



Dynamic Partitioning

(Process Size = Partition Size)

4.Explain about paging concept in detail?

Paging:

•Paging is another memory-management scheme that offers this advantage.
•Paging avoids external fragmentation and the need for compaction whereas segmentation does not.
•Paging is implemented through cooperation between the operating system and the computer hardware.
•Memory is divided into fixed size blocks called page frames.
•The virtual address space of a process is also split into fixed size blocks of the same size called pages.
•When a process is loaded into memory the operating system loads each page into an unused page frame.

When a process is loaded into memory the operating system loads each page into an unused page

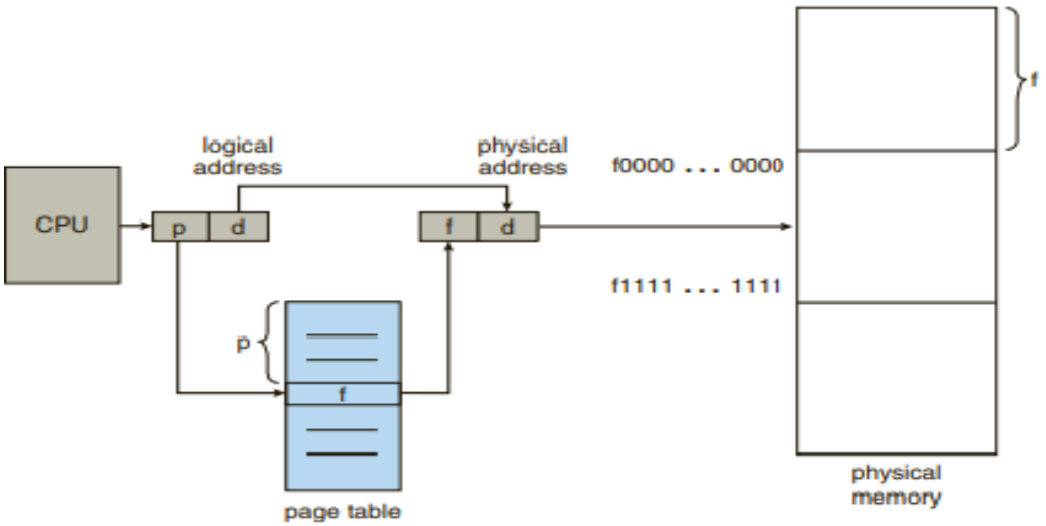
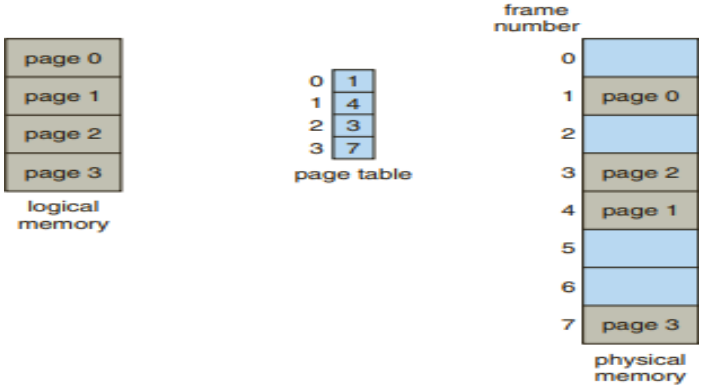


Figure 8.10 Paging hardware.

frame.

OS maintains the page table for each process.
CPU generates the logical address and it consist of two parts:
•Page number(p).
•Page offset(d)
Page table stores the number of the page frame allocated for each .
•The page numbers is used as an index into a page table.
•Page table contains the base address of each page in physical memory.
•The size of the page is typically a power of 2.



Thus, the Logicaladdress is as follows:

When a process arrives in the system to be executed, its size, expressed in pages, is examined.
•Each page of the process needs one frame.
•Thus, if the process requires n pages, at least n frames must be available in memory.
•If n frames are available, they are allocated to this arriving process.
•The first page of the process is loaded into one of the allocated frames, and the frame number is put in the page table for this process.
•The next page is loaded into another frame, its frame number is put into the page table, and so on .

Protection and sharing:

•Protection bits are used with each page frame for protecting memory in paging.
•A bit is generally attached to each entry in the page table a valid-invalid bit.
•When valid bit is set then the page is in the process logical address space .
•Thus page is legal.
•If invalid bit is set ,page is not in the process logical address space and illegal page Illegal address are trapped by using valid-invalid bit.
•The operating system sets bits for each page to allow or disallow access to that page.

5.Explain about segmentation?

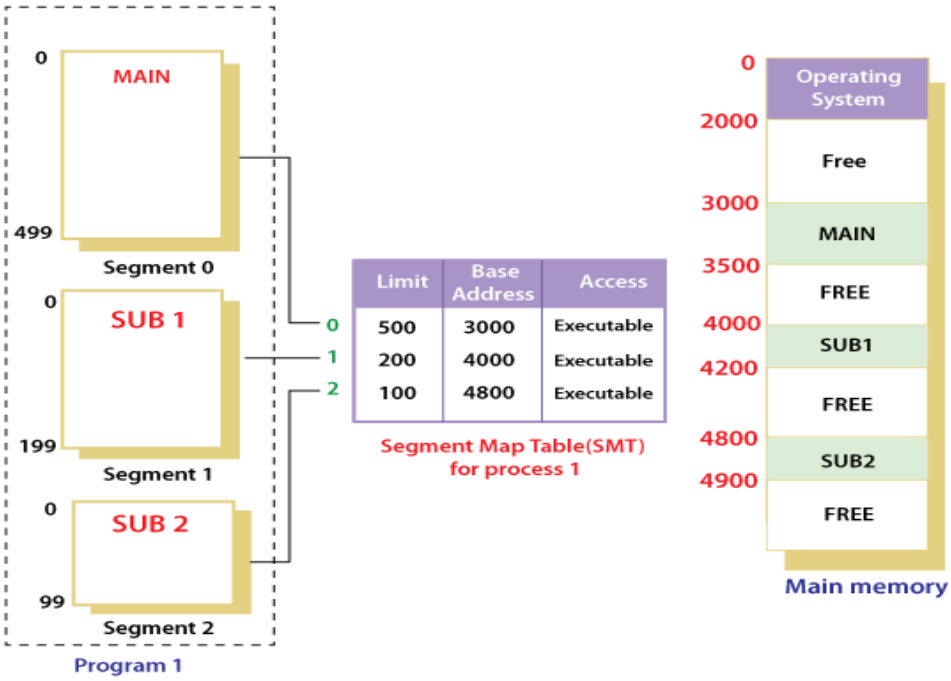
In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.
The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.
Segment table contains mainly two information about segment:
Base: It is the base address of the segment
Limit: It is the length of the segment.

Translation of Logical address into physical address by segment table

CPU generates a logical address which contains two parts:
•Segment Number
•Offset

For Example:

Suppose a 16 bit address is used with 4 bits for the segment number and 12 bits for the segment offset so the maximum segment size is 4096 and the maximum number of segments that can be refereed is 16.
When a program is loaded into memory, the segmentation system tries to locate space that is large enough to hold the first segment of the process, space information is obtained from the free list maintained by memory manager. Then it tries to locate space for other segments. Once adequate space is located for all the segments, it loads them into their respective areas.
The operating system also generates a segment map table for each program.



•With the help of segment map tables and hardware assistance, the operating system can easily translate a logical address into physical address on execution of a program.
•The Segment number is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.
•In the case of valid addresses, the base address of the segment is added to the offset to get the physical address of the actual word in the main memory.
The above figure shows how address translation is done in case of segmentation.

Advantages of Segmentation

1.No internal fragmentation
2.Average Segment Size is larger than the actual page size.
3.Less overhead
4.It is easier to relocate segments than entire address space.
5.The segment table is of lesser size as compared to the page table in paging.

Disadvantages

1.It can have external fragmentation.
2.it is difficult to allocate contiguous memory to variable sized partition.
3.Costly memory management algorithms.

6.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 large virtual memory is provided for user programs when a very small physical memory is there.
In real scenarios, most processes never need all their pages at once, for 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 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.

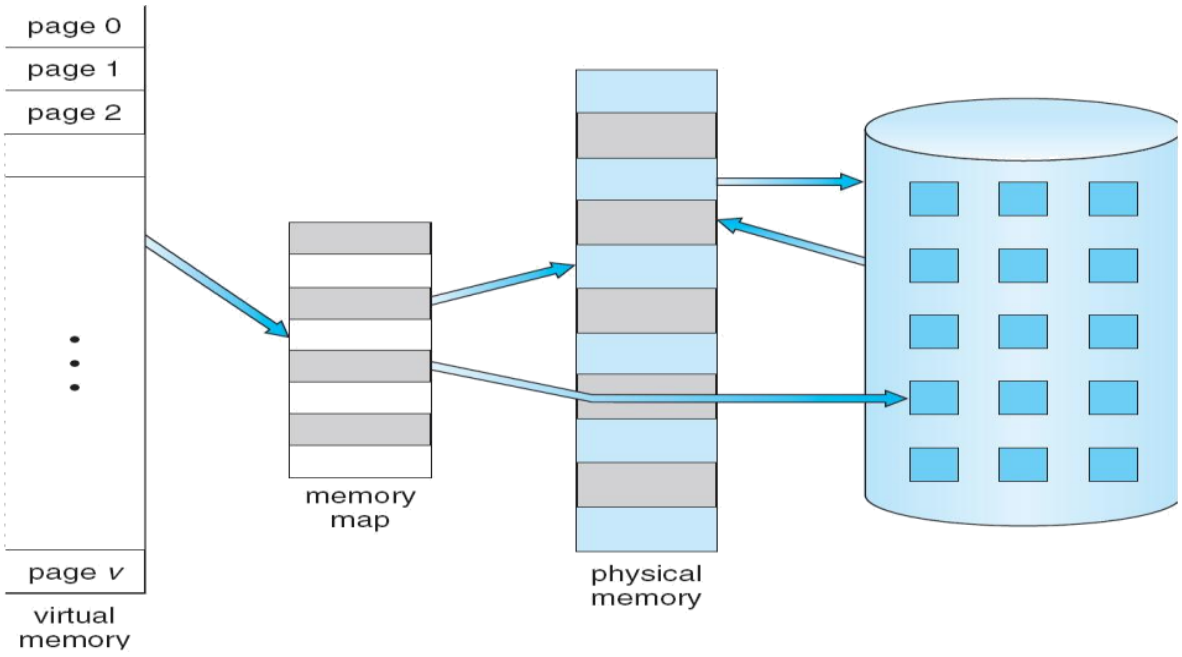


Fig. Diagram showing virtual memory that is larger than physical memory.

Benefits of virtual memory :

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

UNIT V

File and I/O Management, OS security: Directory Structure, 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.

1.Define file and its file operations?

Computers can store information on various storage media such as, magnetic disks, magnetic tapes, optical disks.

The physical storage is converted into a logical storage unit by operating system. The logical storage unit is called FILE.

A file is a collection of similar records. A record is a collection of related fields that can be treated as a unit by some application program.

File Operations:

There are different file operations:

1.Creating a file:

•Two steps are needed to create a file. They are:

•Check whether the space is available or not.

•If the space is available then made an entry for the new file in the directory.

•The entry includes name of the file, path of the file,etc…

2.Writing a file :

•To write a file, we have to know 2 things.

•One is name of the file and second is the information or data to be written on the file, the system searches the entired given location for the file.

•If the file is found, the system must keep a write pointer to the location in the file where the next write is to take place.

3.Reading a file:

•To read a file, first of all we search the directories for the file, if the file is found, the system needs to keep a read pointer to the location in the file where the next read is to take place.

•Once the read has taken place, the read pointer is updated.

4. Repositioning within a file:

•The directory is searched for the appropriate entry and the current file position pointer is repositioned to a given value.

•This operation is also called file seek.

5. Deleting a file:

To delete a file, first of all search the directory for named file, then released the file space and erase the directory entry.

6.Truncating a file:

To truncate a file, remove the file contents only but, the attributes are as it is.

2.Write about File Access Methods?

Files stores information, this information must be accessed and read into computer memory. There are so many ways that the information in the file can be accessed.

There are 3 file access methods:

1.Sequential file access

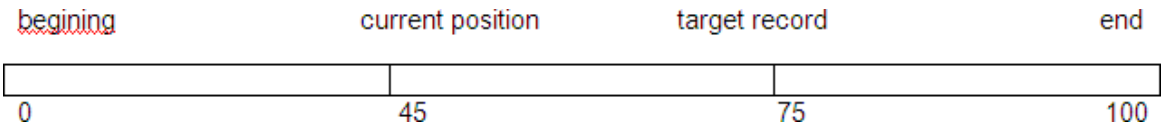
2.direct access method
3.indexed sequential access method

1.Sequential file access:

•Information in the file is processed in order i.e. one record after the other. Magnetic tapes are supporting this type of file accessing.

Eg : A file consisting of 100 records, the current position of read/write head is 45th record, suppose we want to read the 75th record then, it access sequentially from 45, 46, 47 74, 75.

So the read/write head traverse all the records between 45 to 75.



2.Direct access:

•Direct access is also called relative access. Here records can read/write randomly without any order.

•The direct access method is based on a disk model of a file, because disks allow random access to any file block.

•Eg : A disk containing of 256 blocks, the position of read/write head is at 95th block. The block is to be read or write is 250th block. Then we can access the 250th block directly without any restrictions.

•Eg : CD consists of 10 songs, at present we are listening song 3, If we want to listen song 10, we can shift to 10.

3.Indexed Sequential File access:

•The main disadvantage in the sequential file is, it takes more time to access a Record.

•Records are organized in sequence based on a key field.

•Eg: A file consisting of 60000 records, the master index divide the total records into 6 blocks, each block consisting of a pointer to secondary index.

•The secondary index divides the 10,000 records into 10 indexes.

•Each index consisting of a pointer to its original location.

•Each record in the index file consisting of 2 field, A key field and a pointer field.

3.write about directory structure?

•Sometimes the file system consisting of millions of files,at that situation it is very hard to manage the files.

•To manage these files grouped these files and load one group into one partition.

•Each partition is called a directory. a directory structure provides a mechanism for organizing many files in the file system**.**

operation on the directories:

Search for a file: Search a directory structure for required file.

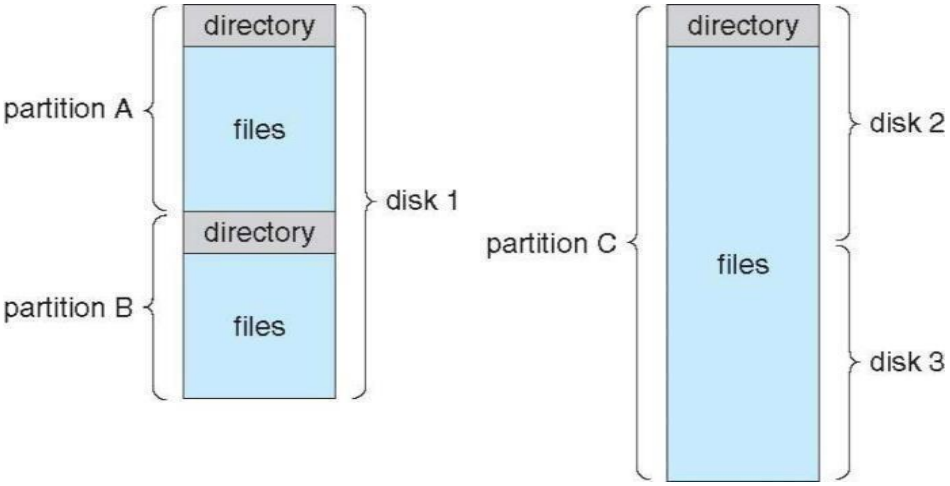
Create file: new files need to be created, added to the directory.

Delete a file: When a file is no longer needed, we want to remove it from the directory.

List a directory: We can know the list of files in the directory.

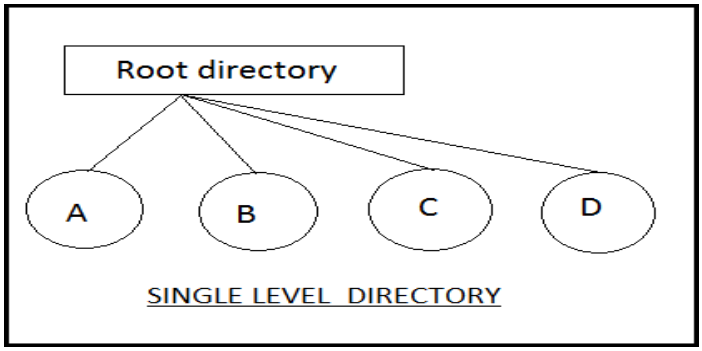
Rename a file: Whenever we need to change the name of the file, we can change the name.

Traverse the file system: We need to access every directory and every file with in a directory structure we can traverse the file system.



The various directory structures

1.Single level directory: The directory system having only one directory, it consisting of all files sometimes it is said to be root directory.

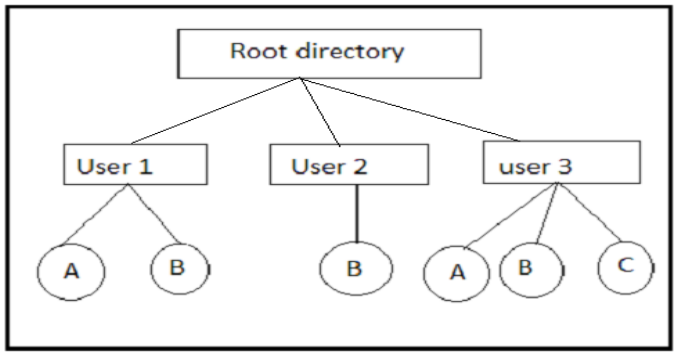


E.g :- Here directory containing 4 files (A,B.C,D).the advantage of the scheme is itssimplicity and the ability to locate files quickly. The problem is different users may accidentally use the same names for their files.

E.g :- If user 1 creates a files called sample and then later user 2 to creates a file called sample, then user2's file will overwrite user 1 file. That's why it is not used inthe multi user system.

2.Two level directory:

•The problem in single level directory is different user may be accidentally use the same name for their files.
•To avoid this problem each user need a private directory, Names chosen by one user don't interfere with names chosen by a different user.
•Root directory is the first level directory. user 1,user2,user3 are user levelof directory A,B,C are files.

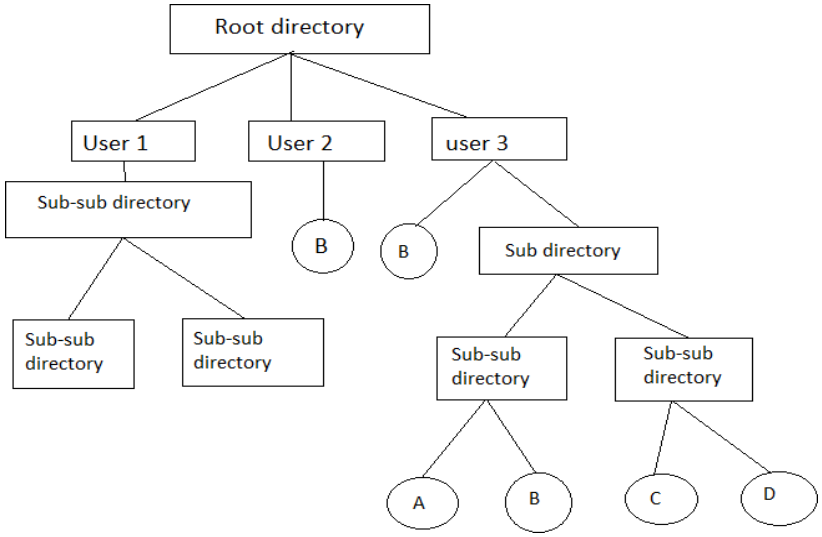


3.Tree structured directory:

•Two level directory eliminates name conflicts among users but it is not satisfactory for users with a large number of files.

•To avoid this create the sub- directory and load the same type of files into the sub-directory.

•so, here each can have as many directories are needed.



4.write about security policy mechanism in os?

In general, an OS security policy is a document that specifies the procedures for ensuring that the operating system maintains a specific level of integrity, confidentiality, and availability.

OS Security protects systems and data from worms, malware, threats, ransomware, backdoor intrusions, viruses, etc. Security policies handle all preventative activities and procedures to ensure an operating system's protection, including steal, edited, and deleted data.

As OS security policies and procedures cover a large area, there are various techniques to addressing them. Some of them are as follows:

1.Installing and updating anti-virus software

2.Ensure the systems are patched or updated regularly

3.Implementing user management policies to protect user accounts and privileges.

4.Installing a firewall and ensuring that it is properly set to monitor all incoming and outgoing traffic.

OS security policies and procedures are developed and implemented to ensure that you must first determine which assets, systems, hardware, and date are the most vital to your organization. Once that is completed, a policy can be developed to secure and safeguard them properly.

5.Explain about Authentication and Internal Access Authorization?

Authentication and authorization are the two words used in the security world. They might similar but are completely different from each other. Authentication is used to authenticate someone's identity, whereas authorization is a way to provide permission to someone to access a particular resource. These are the two basic security terms and hence need to be understood thoroughly.

It is used by both server and client. The server uses authentication when someone wants to access the information, and the server needs to know who is accessing the information. The client uses it when he wants to know that it is the same server that it claims to be.

The authentication by the server is done mostly by using the username and password. Other ways of authentication by the server can also be done using cards, retina scans, voice recognition, and fingerprints.

Authentication does not ensure what tasks under a process one person can do, what files he can view, read, or update. It mostly identifies who the person or system is actually.

Authentication Factors:

As per the security levels and the type of application, there are different types of Authentication factors:

1.Single-Factor Authentication

Single-factor authentication is the simplest way of authentication. It just needs a username and password to allows a user to access a system.

2.Two-factor Authentication

As per the name, it is two-level security; hence it needs two-step verification to authenticate a user. It does not require only a username and password but also needs the unique information that only the particular user knows, such as first school name, a favorite destination. Apart from this, it can also verify the user by sending the OTP or a unique link on the user's registered number or email address.

3.Multi-factor Authentication

This is the most secure and advanced level of authorization. It requires two or more than two levels of security from different and independent categories. This type of authentication is usually used in financial organizations, banks, and law enforcement agencies. This ensures to eliminate any data exposer from the third party or hackers.

Authentication techniques:

1. Password-based authentication

It is the simplest way of authentication. It requires the password for the particular username. If the password matches with the username and both details match the system's database, the user will be successfully authenticated.

2. Password less authentication

In this technique, the user doesn't need any password; instead, he gets an OTP (One-time password) or link on his registered mobile number or phone number. It can also be said OTP-based authentication.

3. 2FA/MFA

2FA/MFA or 2-factor authentication/Multi-factor authentication is the higher level of authentication. It requires additional PIN or security questions so that it can authenticate the user.

4. Social Authentication

Social authentication does not require additional security; instead, it verifies the user with the existing credentials for the available social network.

Authorization:

Authorization is the process of granting someone to do something. It means it a way to check if the user has permission to use a resource or not.

It defines that what data and information one user can access. It is also said as AuthZ.

The authorization usually works with authentication so that the system could know who is accessing the information.

Authorization is not always necessary to access information available over the internet. Some data available over the internet can be accessed without any authorization, such as you can read about any technology from here.

Authorization Techniques

1.Role-based access control

RBAC or Role-based access control technique is given to users as per their role or profile in the organization. It can be implemented for system-system or user-to-system.

2.JSON web token

JSON web token or JWT is an open standard used to securely transmit the data between the parties in the form of the JSON object. The users are verified and authorized using the private/public key pair.

3.SAML

SAML stands for Security Assertion Markup Language. It is an open standard that provides authorization credentials to service providers. These credentials are exchanged through digitally signed XML documents.

4.OpenID authorization

It helps the clients to verify the identity of end-users on the basis of authentication.

5.OAuth

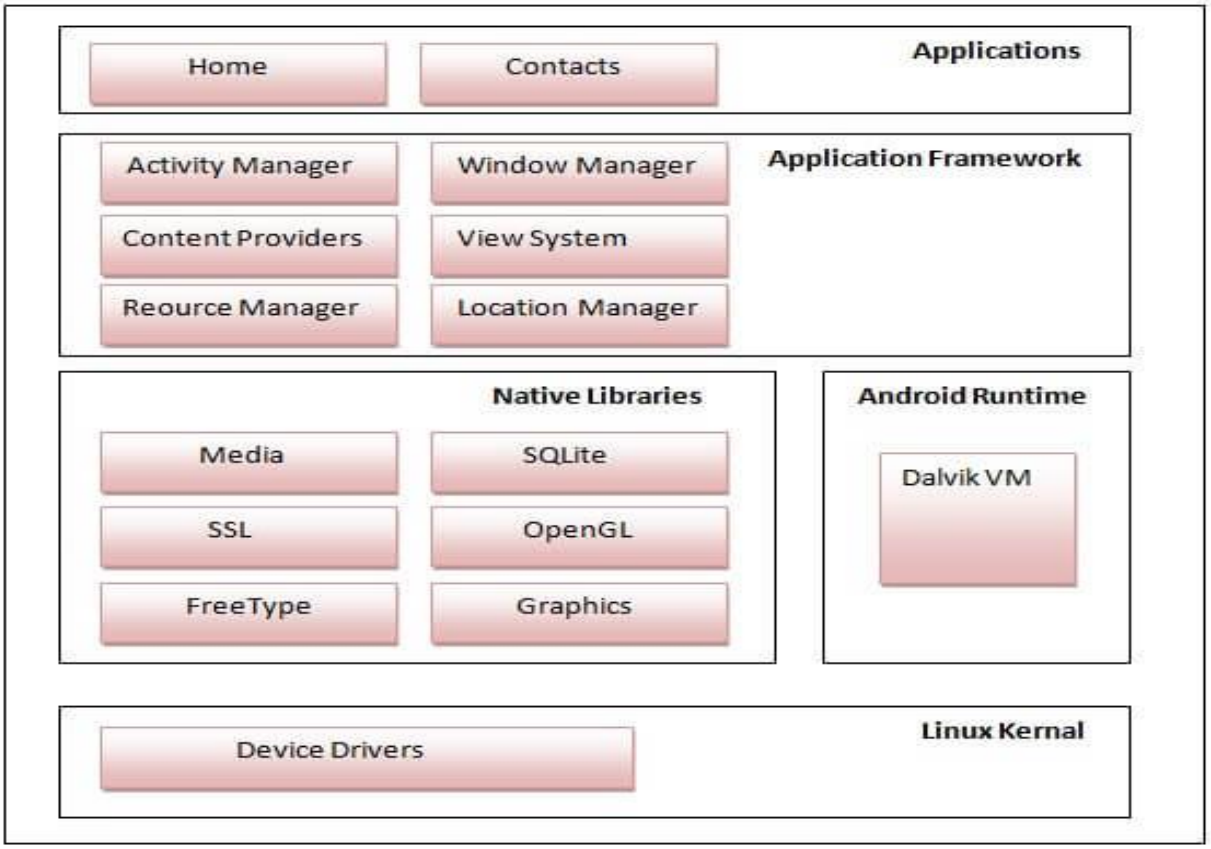
OAuth is an authorization protocol, which enables the API to authenticate and access the requested resources.

6.Define Android os?

Android OS is a Linux-based mobile operating system that primarily runs on smartphones and tablets.
The Android platform includes an operating system based upon the Linux kernel, a GUI, a web browser and end-user applications that can be downloaded.

7.Explain about android application architecture?

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



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 kernel.

8.Explain about Small Application Development using Android Development Framework?
Android is an operating system that is built basically for Mobile phones. It is based on the Linux Kernel and other open-source software and is developed by Google. It is used for touchscreen mobile devices such as smartphones and tablets.

Android Fundamentals
1.Android Programming Languages
In Android, basically, programming is done in two languages JAVA or C++ and XML(Extension Markup Language). Nowadays KOTLIN is also preferred. The XML file deals with the design, presentation, layouts, blueprint, etc (as a front-end) while the JAVA or KOTLIN deals with the working of buttons, variables, storing, etc (as a back-end).
2.Android Components
The App components are the building blocks of Android. Each component has its own role and life cycles i.e from launching of an app till the end. Some of these components depend upon others also. Each component has a definite purpose. The four major app components are:
•Activities
•Services
•Broadcast Receivers:
•Content Provider:

Activities:
It deals with the UI and the user interactions to the screen. In other words, it is a User Interface that contains activities. These can be one or more depending upon the App. It starts when the application is launched. At least one activity is always present which is known as MainActivity. The activity is implemented through the following.

Syntax:

```
Public class MainActivity extends Activity
{
  // processes
}
```

Services:
Services are the background actions performed by the app, these might be long-running operations like a user playing music while surfing the Internet. A service might need other sub-services so as to perform specific tasks. The main purpose of the Services is to provide non-stop working of the app without breaking any interaction with the user.

Syntax:

```
public class MyServices extends Services
{
  // code for the services
}
```

Broadcast Receivers:
A Broadcast is used to respond to messages from other applications or from the System. For example, when the battery of the phone is low, then the Android OS fires a Broadcasting message to launch the Battery

Saver function or app, after receiving the message the appropriate action is taken by the app. Broadcast Receiver is the subclass of BroadcastReceiver class and each object is represented by Intent objects.
Syntax:

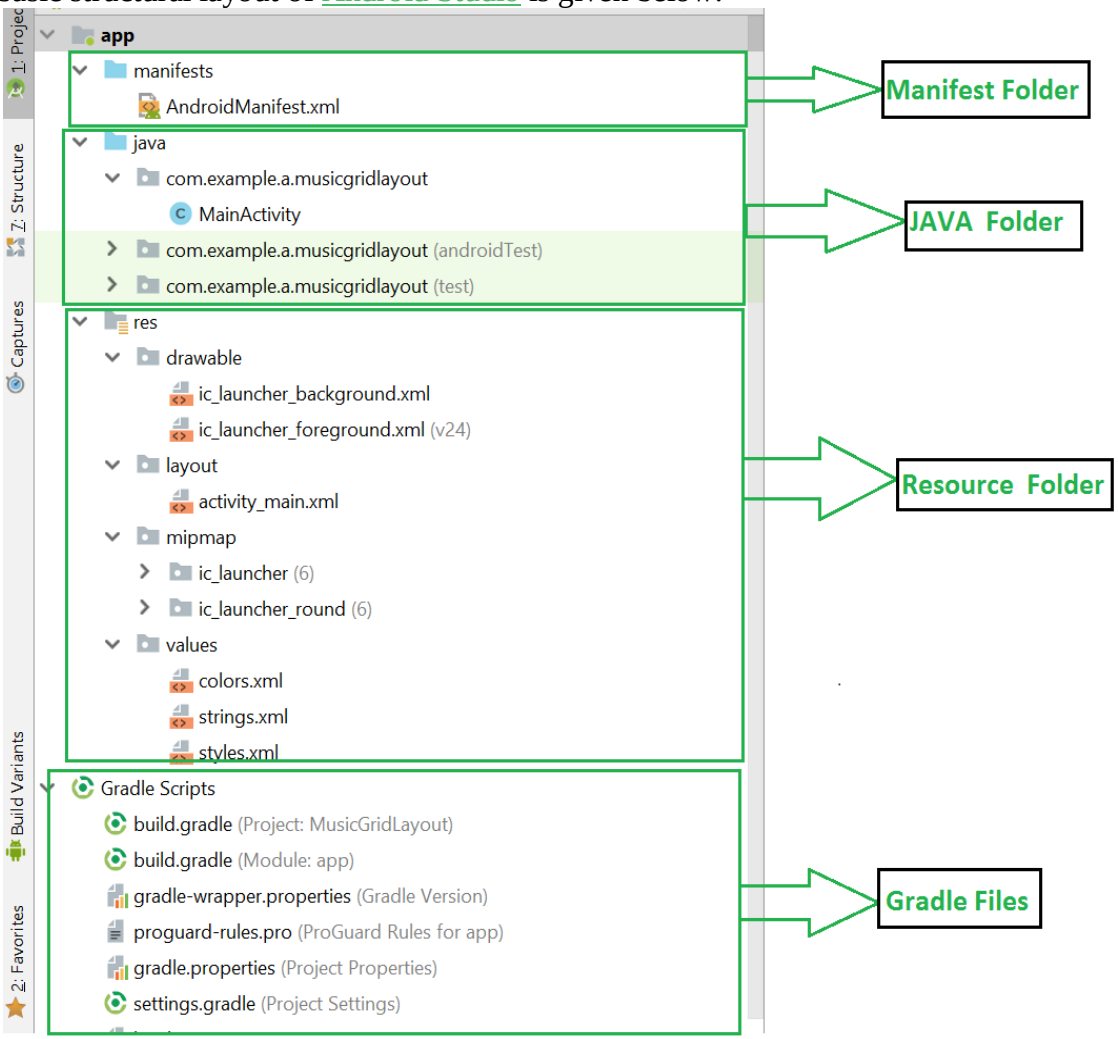
```
public class MyReceiver extends BroadcastReceiver
{
  public void onReceive(context,intent)
  { }
}
```

Content Provider:
Content Provider is used to transferring the data from one application to the others at the request of the other application. These are handled by the class ContentResolver class. This class implements a set of APIs(Application Programming Interface) that enables the other applications to perform the transactions. Any Content Provider must implement the Parent Class of ContentProvider class.

Syntax:

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

3. Structural Layout Of Android Studio
The basic structural layout of Android Studio is given below:



The above figure represents the various structure of an app.
Manifest Folder: Android Manifest is an XML file that is the root of the project source set. It describes the essential information about the app and the Android build tools, the Android Operating System, and Google Play. It contains the permission that an app might need in order to perform a specific task. It also contains the Hardware and the Software features of the app, which determines the compatibility of an app on the Play Store. It also includes special activities like services, broadcast receiver, content providers, package name, etc.

Java Folder: The **JAVA folder** consists of the java files that are required to perform the background task of the app. It consists of the functionality of the buttons, calculation, storing, variables, toast(small popup message), programming function, etc. The number of these files depends upon the type of activities created.

Resource Folder: The res or Resource folder consists of the various resources that are used in the app. This consists of sub-folders like drawable, layout, mipmap, raw, and values. The drawable consists of the images. The layout consists of the XML files that define the user interface layout. These are stored in res.layout and are accessed as R.layout class. The raw consists of the Resources files like audio files or music files, etc. These are accessed through R.raw.filename. values are used to store the hardcoded strings(considered safe to store string values) values, integers, and colors. It consists of various other directories like:
R.array :arrays.xml for resource arrays
R.integer : integers.xml for resource integers
R.bool : bools.xml for resource boolean
R.color :colors.xml for color values
R.string : strings.xml for string values
R.dimen : dimens.xml for dimension values
R.style : styles.xml for styles

Gradle Files: Gradle is an advanced toolkit, which is used to manage the build process, that allows defining the flexible custom build configurations. Each build configuration can define its own set of code and resources while reusing the parts common to all versions of your app. The Android plugin for Gradle works with the build toolkit to provide processes and configurable settings that are specific to building and testing Android applications. Gradle and the Android plugin run independently of Android Studio. This means that you can build your Android apps from within Android Studio. The flexibility of the Android build system enables you to perform custom build configurations without modifying your app's core source files.

Basic Layout Can be defined in a tree structure as:

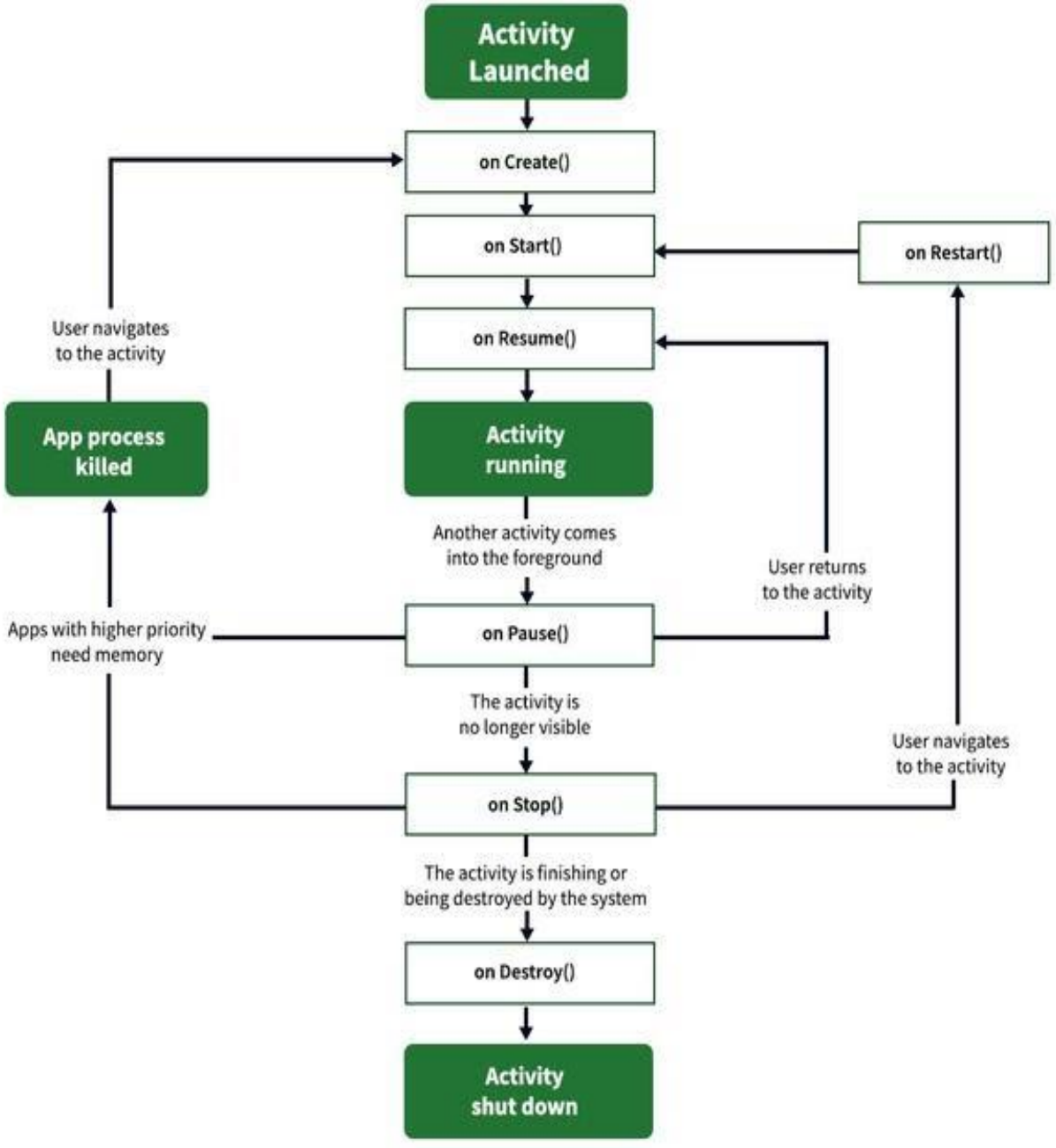
```
Project/
  app/
    manifest/
      AndroidManifest.xml
  java/
    MyActivity.java
    res/
      drawable/
        icon.png
        background.png
      drawable-hdpi/
        icon.png
        background.png
      layout/
        activity_main.xml
        info.xml
      values/
        strings.xml
```

4. Lifecycle of Activity in Android App

The **Lifecycle** of Activity in Android App can be shown through this diagram:

States of Android Lifecycle:

OnCreate: This is called when activity is first created.
OnStart: This is called when the activity becomes visible to the user.
OnResume: This is called when the activity starts to interact with the user.
OnPause: This is called when activity is not visible to the user.
OnStop: This is called when activity is no longer visible.
OnRestart: This is called when activity is stopped, and restarted again.
OnDestroy: This is called when activity is to be closed or destroyed.



Activity Lifecycle in Android