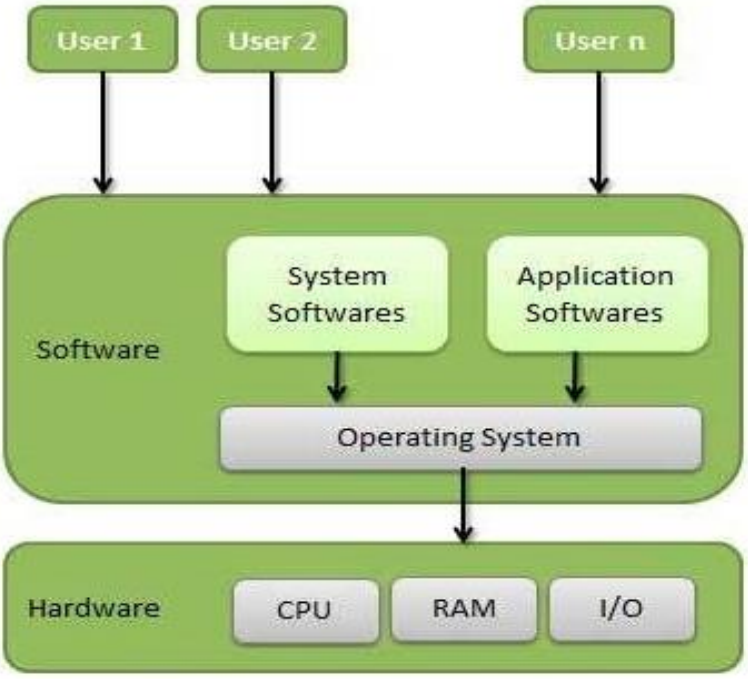
UNIT-1 : OPERATING SYSTEM STRUCTURES

1. Define OS ? What are the objectives of Operating systems?

- An operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
- An operating system is concerned with the allocation of resources and services, such as memory, processors, devices, and information. The operating system correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, a memory management module, I/O programs, and a file system.



The objectives of the operating system are −

- To make the computer system convenient to use in an efficient manner.
- To hide the details of the hardware resources from the users.
- To provide users a convenient interface to use the computer system.
- To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
- To manage the resources of a computer system.
- To keep track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
- To provide efficient and fair sharing of resources among users and programs.

2. List and explain any four services provided by an operating system.

➢ Following are a few common services provided by an operating system –

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

- Communication
- Error Detection
- Resource Allocation
- Protection

Program execution-

Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process. A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management –

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

I/O Operation-

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

An Operating System manages the communication between user and device drivers. • I/O operation means read or write operation with any file or any specific I/O device. • Operating system provides the access to the required I/O device when required.

File system manipulation-

A file represents a collection of related information. Computers can store files on the disk (secondary storage), for longterm storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD.

Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

Following are the major activities of an operating system with respect to file management –

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

Communication –

In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.
The OS handles routing and connection strategies, and the problems of contention and security.
Following are the major activities of an operating system with respect to communication –
• Two processes often require data to be transferred between them
• Both the processes can be on one computer or on different computers, but are connected through a computer network.
• Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

Error handling –

Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware.
Following are the major activities of an operating system with respect to error handling –
• The OS constantly checks for possible errors.
• The OS takes an appropriate action to ensure correct and consistent computing.

Resource Management -

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

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

Protection-

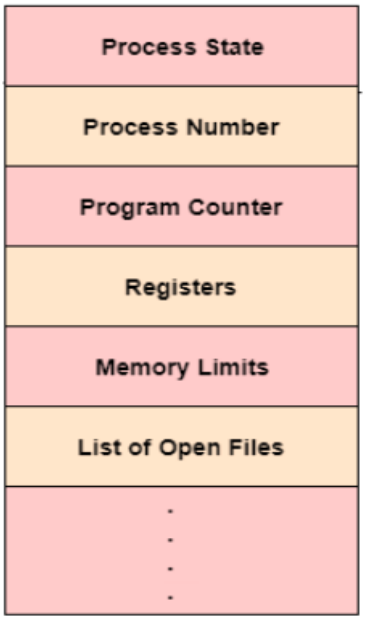
Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.
Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection –
• The OS ensures that all access to system resources is controlled.
• The OS ensures that external I/O devices are protected from invalid access attempts. • The OS provides authentication features for each user by means of passwords.

3. Define process. Explain process control block.

- **Process** is the execution of a program that performs the actions specified in that program. It can be defined as an execution unit where a program runs. The OS helps you to create, schedule, and terminates the processes which is used by CPU. A process created by the main process is called a child process.

Process Control Block:

- Process Control Block is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.
- It is very important for process management as the data structuring for processes is done in terms of the PCB. It also defines the current state of the operating system.



Process Control Block (PCB)

Every process is represented in the operating system by a process control block, which is also called a task control block.

The following are the data items −

- **Process State:**
  This specifies the process state i.e. new, ready, running, waiting or terminated.
- **Process Number:**
  This shows the number of the particular process.
- **Program Counter :**
  This contains the address of the next instruction that needs to be executed in the process.

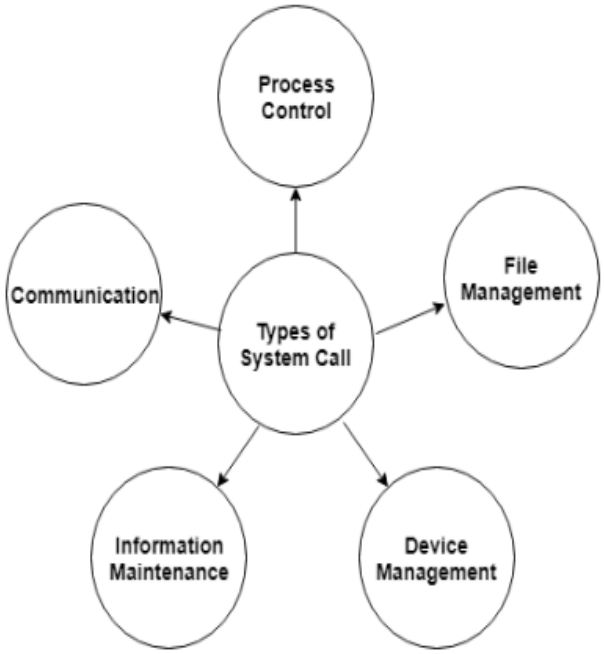
- **Registers :**
  This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.
- **List of Open Files :**
  These are the different files that are associated with the process

- **CPU Scheduling Information**
  The process priority, pointers to scheduling queues etc. is the CPU scheduling information that is contained in the PCB. This may also include any other scheduling parameters.
- **Memory Management Information:**
  The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc.
- **I/O Status Information:**
  This information includes the list of I/O devices used by the process, the list of files etc.
- **Accounting information**
  The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.
- **Location of the Process Control Block**
  The process control block is kept in a memory area that is protected from the normal user access. This is done because it contains important process information. Some of the operating systems place the PCB at the beginning of the kernel stack for the process as it is a safe location.

4. Define system call. Enlist and explain types of system calls.

- A **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on.
- A system call is a way for programs to **interact with the operating system**.
- System call **provides** the services of the operating system to the user programs via Application Program Interface(API).
- It provides an interface between a process and operating system to allow user-level processes to request services of the operating system.

➢ **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 inter-process communication. They also deal with creating and deleting a communication connection.

5. What are the advantages of peer to peer systems over client server systems?

- A peer-to-peer (P2P) network is created when two or more PCs are connected and share resources without going through a separate server computer.
- Advantages of Peer-to-peer networking over Client –Server Systems are **:-**
  - It is easy to install and so is the configuration of computers on this network.
  - All the resources and contents are shared by all the peers, unlike server-client architecture where Server shares all the contents and resources.

- P2P is more reliable as central dependency is eliminated. Failure of one peer doesn't affect the functioning of other peers. In case of Client –Server network, if server goes down whole network gets affected.
- There is no need for full-time System Administrator. Every user is the administrator of his machine. User can control their shared resources.
- The over-all cost of building and maintaining this type of network is comparatively very less.

6. Differentiate between hard real time systems and soft real time systems

Terms	Hard Real-Time System	Soft Real-Time System
Definition	A hard-real time system is a system in which a failure to meet even a single deadline may lead to complete or appalling system failure.	A soft real-time system is a system in which one or more failures to meet the deadline are not considered complete system failure, but that performance is considered to be degraded.
File size	In a hard real-time system, the size of a data file is small or medium.	In a soft real-time system, the size of the data file is large.
Response time	In this system, response time is predefined that is in a millisecond.	In this system, response time is higher.
Utility	A hard-real time system has more utility.	A soft real-time system has less utility.
Database	A hard real-time system has short databases.	A soft real-time system has enlarged databases.
Performance	Peak load performance should be predictable.	In a soft real-time system, peak load can be tolerated.
Safety	In this system, safety is critical.	In this system, safety is not critical.
Integrity	Hard real-time systems have short term data integrity.	Soft real-time systems have long term data integrity.
Restrictive nature	A hard real-time system is very restrictive.	A Soft real-time system is less restrictive.
Computation	In case of an error in a hard real-time system, the computation is rolled back.	In a soft real-time system, computation is rolled back to a previously established checkpoint to initiate a recovery action.
Flexibility and laxity	Hard real-time systems are not flexible, and they have less laxity and generally provide full deadline compliance.	Soft real-time systems are more flexible. They have greater laxity and can tolerate certain amounts of deadline misses.
Validation	All users of hard real-time systems get validation when needed.	All users of soft real-time systems do not get validation.

Examples	Satellite launch, Railway signalling systems, and Safety-critical systems are good examples of a hard real-time system.	DVD player, telephone switches, electronic games, Linux, and many other OS provide a soft real-time system.

7. **Describe what virtual machine is and what are the advantages virtual machine.**
   - A **virtual machine (VM)** is a virtual environment which functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical hardware system.
   - It takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
   - A virtual machine provides an interface *identical* to the underlying bare hardware.
   - The operating system host creates the illusion that a process has its own processor and (virtual memory)

- **Benefits-**

VMs are independent of each other, they're also extremely portable. Because of their flexibility and portability, virtual machines provide many benefits, such as:

- **Cost savings**—running multiple virtual environments from one piece of infrastructure means that you can drastically reduce your physical infrastructure footprint. This boosts your bottom line—decreasing the need to maintain nearly as many servers and saving on maintenance costs and electricity.

- **Agility and speed**—Spinning up a VM is relatively easy and quick and is much simpler than provisioning an entire new environment for your developers. Virtualization makes the process of running dev-test scenarios a lot quicker.

- **Lowered downtime**—VMs are so portable and easy to move from one hypervisor to another on a different machine—this means that they are a great solution for backup, in the event the host goes down unexpectedly.

- **Scalability**—VMs allow you to more easily scale your apps by adding more physical or virtual servers to distribute the workload across multiple VMs. As a result you can increase the availability and performance of your apps.

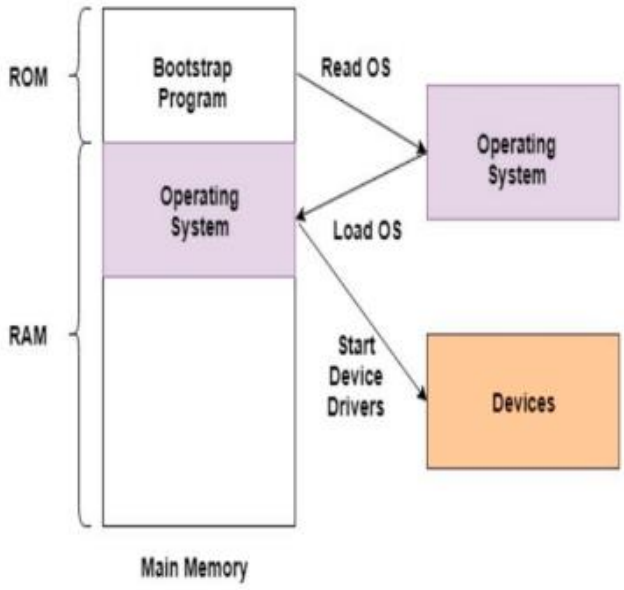
- **Security benefits**— Because virtual machines run in multiple operating systems, using a guest operating system on a VM allows you to run apps of questionable security and protects your host operating system. VMs also allow for better security forensics, and are often used to safely study computer viruses, isolating the viruses to avoid risking their host computer.

- **Advantages-**
  - Virtual Machine facilitates compatibility of the software to that software which is executing on it. Hence, each software specified for a virtualized host would also execute on the VM.
  - It offers isolation among distinct types of processors and OSes. Hence, the processor OS executing on a single virtual machine can't change the host of any other host systems and virtual machines.

- Virtual Machine facilitates encapsulation. various software present over the VM could be controlled and modified.
- Virtual machines give several features such as the addition of **new operating system.** An error in a single operating system will not affect any other operating system available on the host. It offers the transfer of many files between VMs, and no dual booting for the multi-OS host.
- VM provides better management of software because VM can execute a complete stack of software of the run legacy operating system, host machine, etc.
- It can be possible to distribute hardware resources to software stacks independently. The VM could be transferred to distinct computers for balancing the load.

8. **Define system call**.
   Same as 4.

9. **Describe bootstrap program.**
   A bootstrap program is the first code that is executed when the computer system is started. The entire operating system depends on the bootstrap program to work correctly as it loads the operating system. A figure that demonstrates the use of the bootstrap program is as follows:



In the above image, the bootstrap program is a part of ROM which is the non-volatile memory. The operating system is loaded into the RAM by the bootstrap program after the start of the computer system. Then the operating system starts the device drivers.

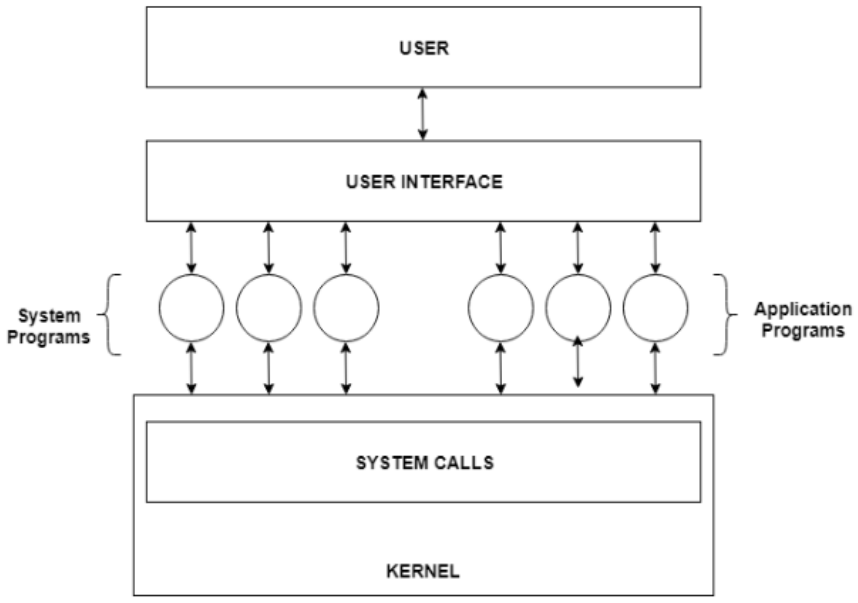
- **<u>Bootstrapping Process</u>**-
  The bootstrapping process does not require any outside input to start. Any software can be loaded as required by the operating system rather than loading all the software automatically.

The bootstrapping process is performed as a chain i.e. at each stage, it is the responsibility of the simpler and smaller program to load and execute the much more complicated and larger program. This means that the computer system improves in increments by itself.
The booting procedure starts with the hardware procedures and then continues onto the software procedures that are stored in the main memory. The bootstrapping process involves self-tests, loading BIOS, configuration settings, hypervisor, operating system etc.

- **Benefits of Bootstrapping –**
  Without bootstrapping, the computer user would have to download all the software components, including the ones not frequently required. With bootstrapping, only those software components need to be downloaded that are legitimately required and all extraneous components are not required. This process frees up a lot of space in the memory and consequently saves a lot of time.

10. What is the purpose of system programs?

- The system program serves as a part of the operating system. It traditionally lies between the user interface and the system calls. The user view of the system is actually defined by system programs and not system calls because that is what they interact with and system programs are closer to the user interface.
- An image that describes system programs in the operating system hierarchy is as follows:



In the above image, system programs as well as application programs form a bridge between the user interface and the system calls. So, from the user view the operating system observed is actually the system programs and not the system calls.

- **Types of System Programs-**
  System programs can be divided into seven parts. These are given as follows:

- **Status Information**
  The status information system programs provide required data on the current or past status of the system. This may include the system date, system time, available memory in system, disk space, logged in users etc.

- **Communications**
  These system programs are needed for system communications such as web browsers. Web browsers allow systems to communicate and access information from the network as required.

- **File Manipulation**
  These system programs are used to manipulate system files. This can be done using various commands like create, delete, copy, rename, print etc. These commands can create files, delete files, copy the contents of one file into another, rename files, print them etc.

- **Program Loading and Execution**
  The system programs that deal with program loading and execution make sure that programs can be loaded into memory and executed correctly. Loaders and Linkers are a prime example of this type of system programs.

- **File Modification**
  System programs that are used for file modification basically change the data in the file or modify it in some other way. Text editors are a big example of file modification system programs.

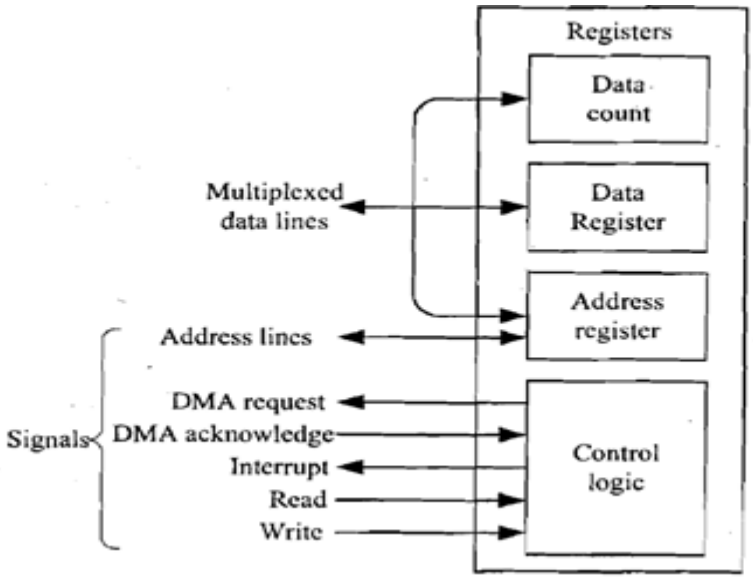
- **Application Programs**
  Application programs can perform a wide range of services as per the needs of the users. These include programs for database systems, word processors, plotting tools, spreadsheets, games, scientific applications etc.

- **Programming Language Support**
  These system programs provide additional support features for different programming languages. Some examples of these are compilers, debuggers etc. These compile a program and make sure it is error free respectively.

11. Summarize the functions of DMA

- Though CPU intervention in DMA is minimised however it should use path between interfaces that is system bus. So DMA involves an extra interface on system bus.
- A scheme known as cycle stealing permits DMA interface to transfer one data word at a time, after that it should return control of bus to processor. Processor merely delays its operation for one memory cycle to permit directly memory I/O transfer to 'steal' one memory cycle.
- When an I/O is requested processor issues a command to DMA interface by transmitting to DMA interface the subsequent information
- Which operations (write or read) to be performed using write or read control lines.
- Address of I/O devices that is to be used communicated on data lines.
- Starting location on memory where information will be read or written to be communicated on data lines and is stored by DMA interface in its address register.

- Number of words which are to be read or written is communicated on data lines and is stored in data count register.



Explanation :

The DMA controller has three registers as follows.

- **Address register –** It contains the address to specify the desired location in memory.
- **Word count register –** It contains the number of words to be transferred.
- **Control register –** It specifies the transfer mode.

- The CPU initializes the DMA by sending the given information through the data bus.
- The starting address of the memory block where the data is available (to read) or where data are to be stored (to write).
- It also sends word count which is the number of words in the memory block to be read or write.
- Control to define the mode of transfer such as read or write.
- A control to begin the DMA transfer.

12. Summarize the objectives and functions of an operating system

➢ **Objectives of Operating System-**

➢ The objectives of the operating system are −

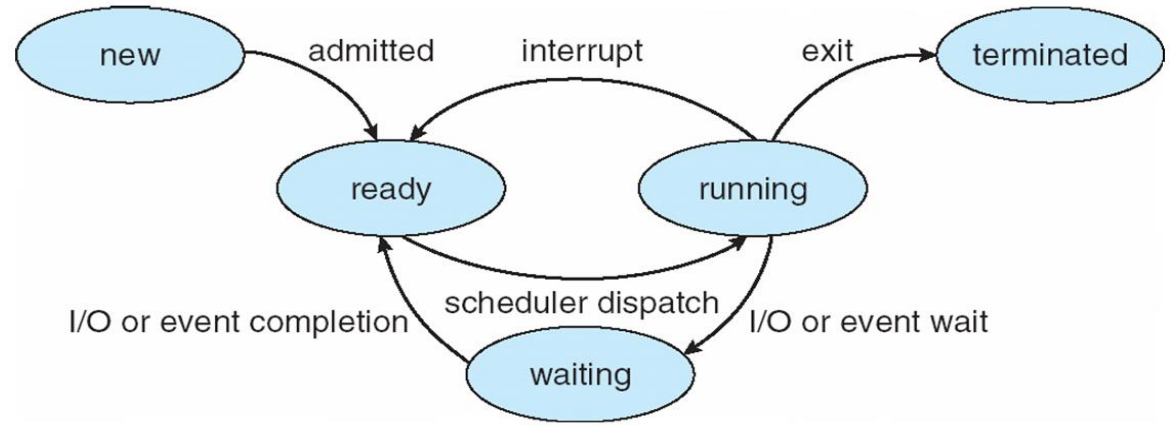
- To make the computer system convenient to use in an efficient manner.
- To hide the details of the hardware resources from the users.
- To provide users a convenient interface to use the computer system.
- To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
- To manage the resources of a computer system.
- To keep track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
- To provide efficient and fair sharing of resources among users and programs.

- **Characteristics of Operating System-**
  - Here is a list of some of the most prominent characteristic features of Operating Systems −

- **Memory Management** − Keeps track of the primary memory, i.e. what part of it is in use by whom, what part is not in use, etc. and allocates the memory when a process or program requests it.
- **Processor Management** − Allocates the processor (CPU) to a process and deallocates the processor when it is no longer required.
- **Device Management** − Keeps track of all the devices. This is also called I/O controller that decides which process gets the device, when, and for how much time.
- **File Management** − Allocates and de-allocates the resources and decides who gets the resources.
- **Security** − Prevents unauthorized access to programs and data by means of passwords and other similar techniques.
- **Job Accounting** − Keeps track of time and resources used by various jobs and/or users.
- **Control Over System Performance** − Records delays between the request for a service and from the system.
- **Interaction with the Operators** − Interaction may take place via the console of the computer in the form of instructions. The Operating System acknowledges the same, does the corresponding action, and informs the operation by a display screen.
- **Error-detecting Aids** − Production of dumps, traces, error messages, and other debugging and error-detecting methods.
- **Coordination Between Other Software and Users** − Coordination and assignment of compilers, interpreters, assemblers, and other software to the various users of the computer systems.

UNIT NO-2 :PROCESS AND CPU SCHEDULING

1. **Draw and explain process state diagram**



The process, from its creation to completion, passes through various states. The minimum number of states is five.

The names of the states are not standardized although the process may be in one of the following states during execution.

1. **NEW:** A program which is going to be picked up by the OS into the main memory is called a new process.
2. **READY:** Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned. The OS picks the new processes from the secondary memory and put all of them in the main memory. The processes which are ready for the execution and reside in the main memory are called ready state processes. There can be many processes present in the ready state.
3. **RUNNING:** One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm. Hence, if we have only one CPU in our system, the number of running processes for a particular time will always be one. If we have n processors in the system then we can have n processes running simultaneously.
4. **BLOCK OR WAIT:** From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process. When a process waits for a certain resource to be assigned or for the input from the user then the OS move this process to the block or wait state and assigns the CPU to the other processes.
5. **COMPLETION OR TERMINATION:** When a process finishes its execution, it comes in the termination state. All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system.
6. **SUSPEND READY:** A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state.

   If the main memory is full and a higher priority process comes for the execution then the OS have to make the room for the process in the main memory by throwing the lower priority

process out into the secondary memory. The suspend ready processes remain in the secondary memory until the main memory gets available.

2. Consider the following set of processes, with the length of the CPU burst given in miliseconds:

Process	Burst Time	Arrival Time
P1	3	0
P2	1	1
P3	6	3
P4	6	7
P5	5	8

a. Draw Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: SJF preemptive, SJF non-pre-emptive, FCFS, Round Robin (Time Quantum = 5 ms), Priority scheduling.

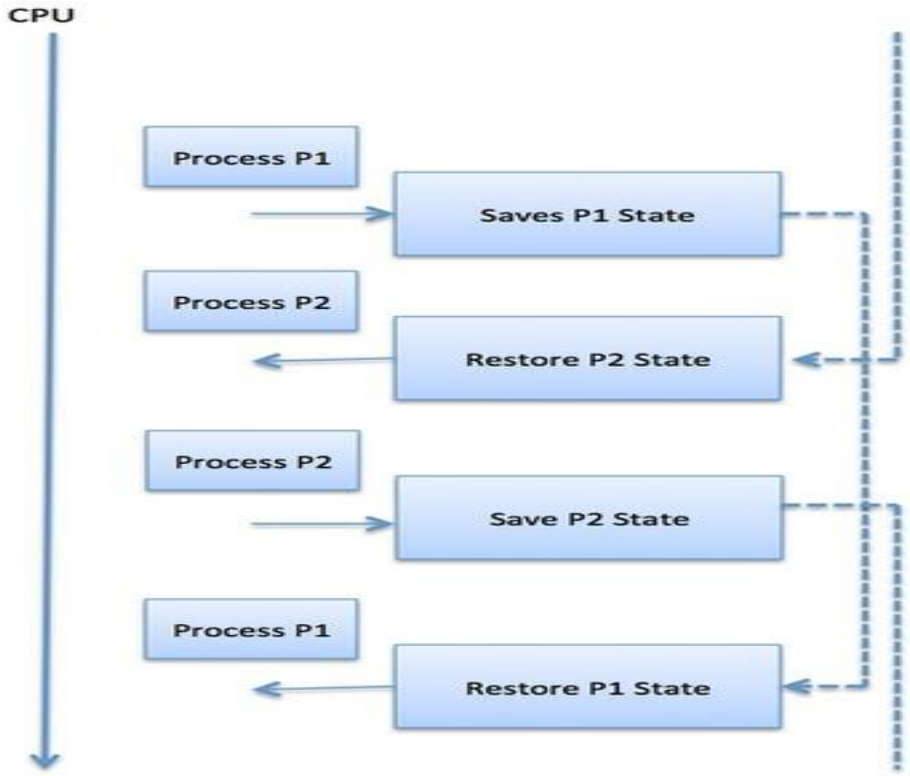
b. What is the average waiting time in each case.

3.Describe in brief:

i) 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.



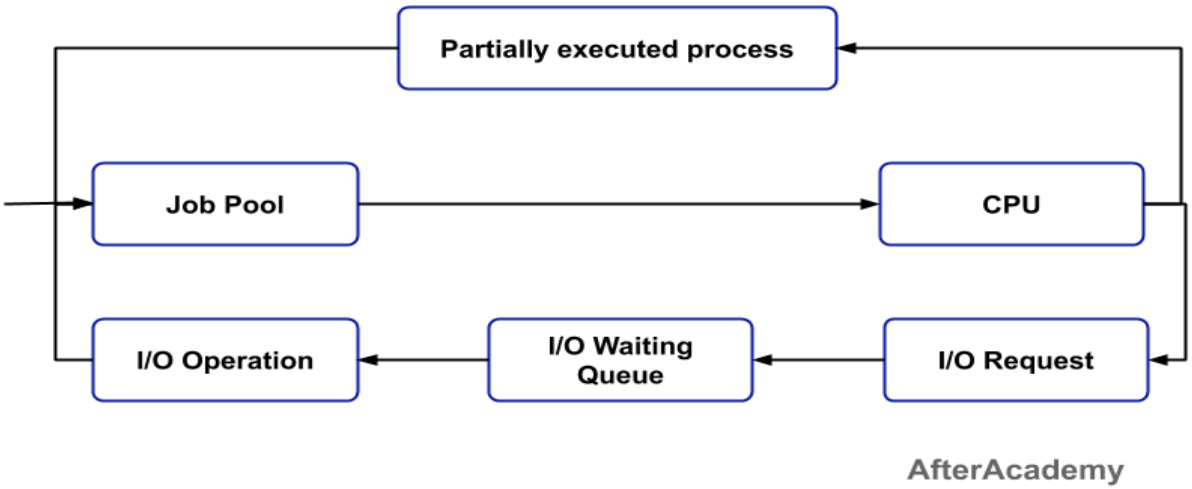
Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.

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

ii) 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.

- Medium-term schedulers are those schedulers whose decision will have a mid-term effect on the performance of the system. It is responsible for swapping of a process from the Main Memory to Secondary Memory and vice-versa**.**
- It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.



4. Differentiate between program and process.

Parameter	Process	Program
Definition	An executing part of a program is called a process.	A program is a group of ordered operations to achieve a programming goal.
Nature	The process is an instance of the program being executing.	The nature of the program is passive, so it's unlikely to do to anything until it gets executed.
Resource management	The resource requirement is quite high in case of a process.	The program only needs memory for storage.
Overheads	Processes have considerable overhead.	No significant overhead cost.
Lifespan	The process has a shorter and very limited lifespan as it gets terminated after the completion of the task.	A program has a longer lifespan as it is stored in the memory until it is not manually deleted.
Creation	New processes require duplication of the parent process.	No such duplication is needed.
Required Process	Process holds resources like CPU, memory address, disk,	The program is stored on disk in

Parameter	Process	Program
	I/O, etc.	some file and does not require any other resources.
Entity type	A process is a dynamic or active entity.	A program is a passive or static entity.
Contain	A process contains many resources like a memory address, disk, printer, etc.	A program needs memory space on disk to store all instructions.

5. Draw and explain short term scheduler.

Short-term schedulers are those schedulers whose decision will have a short-term effect on the performance of the system. The duty of the short-term scheduler is to schedule the process from the ready state to the running state. This is the place where all the scheduling algorithms are used i.e. it can be FCFS or Round-Robin or SJF or any other scheduling algorithm.

Short-Term Scheduler is also known as CPU scheduler and is responsible for selecting one process from the ready state for scheduling it on the running state.

Effect on performance

- The choice of the short-term scheduler is very important for the performance of the system. If the short-term scheduler only selects a process that is having very high burst time(learn more about burst time from here) then the other process may go into the condition of starvation(learn more about starvation from here). So, be specific when you are choosing short-term scheduler because the performance of the system is our highest priority.

The following image shows the scheduling of processes using the long-term and short-term scheduler.

Long-Term Scheduler

Long-Term schedulers are those schedulers whose decision will have a long-term effect on the performance. The duty of the long-term scheduler is to bring the process from the JOB pool to the Ready state for its execution.

Long-Term Scheduler is also called Job Scheduler and is responsible for controlling the Degree of Multiprogramming i.e. the total number of processes that are present in the ready state.

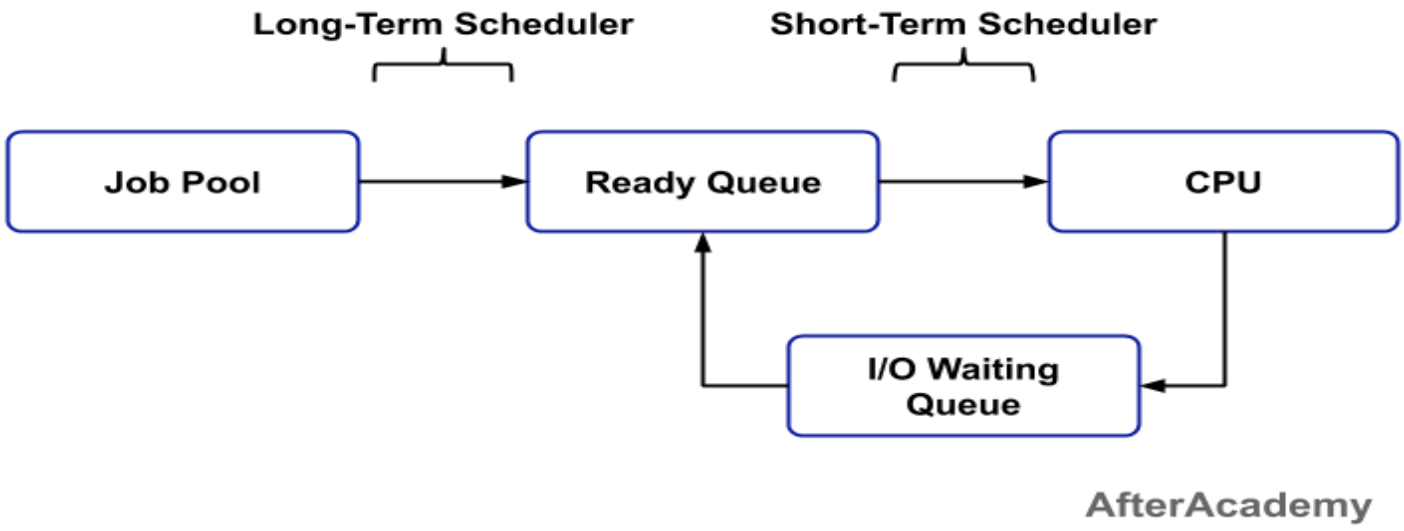
So, the long-term scheduler decides which process is to be created to put into the ready state.

Effect on performance

- The long term scheduler is responsible for creating a balance between the I/O bound(a process is said to be I/O bound if the majority of the time is spent on the I/O operation) and CPU bound(a process is said to be CPU bound if the majority of the time is spent on the CPU). So, if we create processes

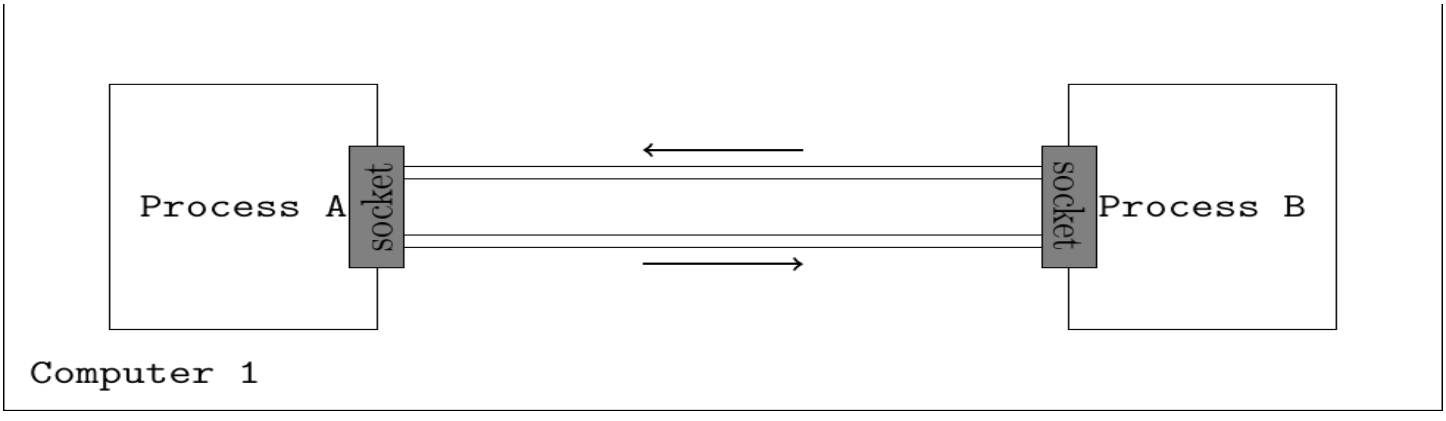
which are all I/O bound then the CPU might not be used and it will remain idle for most of the time. This is because the majority of the time will be spent on the I/O operation.

- So, if we create processes that are having high a CPU bound or a perfect balance between the I/O and CPU bound, then the overall performance of the system will be increased.



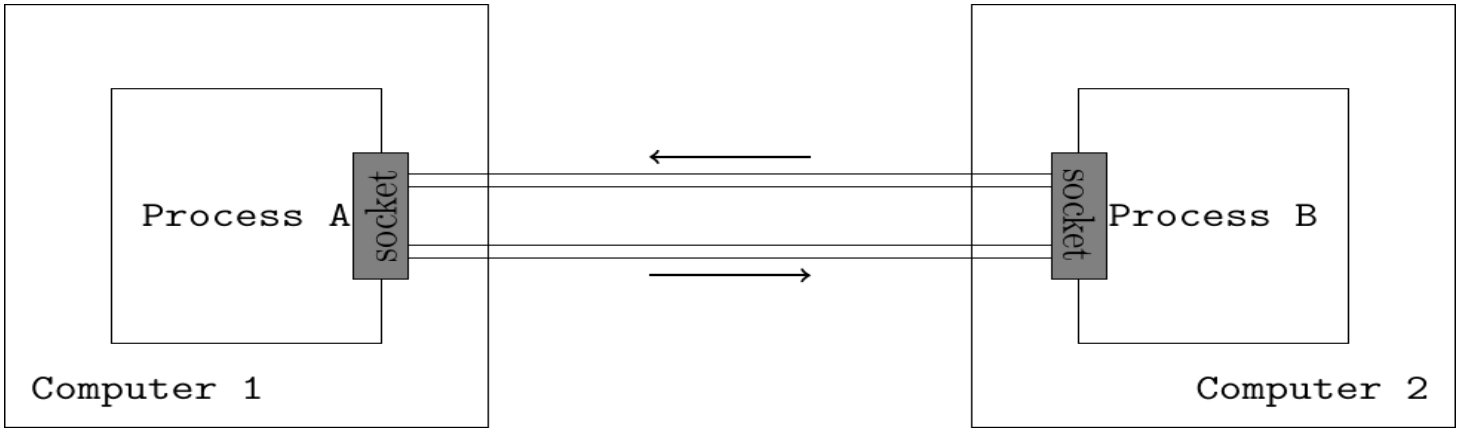
6. Justify role of 'Socket' in inter-process communication with suitable example.

- Popular operating systems allow isolating different programs by executing them in separate *processes*. A socket is a tool provided by the operating system that enables two separated processes to communicate with each other. A socket takes the form of a file descriptor and can be seen as a communication pipe through which the communicating processes can exchange arbitrary information. In order to receive a message, a process must be attached to a specific address that the peer can use to reach it.



Connecting two processes communicating on the same computer

The socket is a powerful abstraction as it allows processes to communicate even if they are located on different computers. In this specific cases, the inter-processes communication will go through a network.



Connecting two processes communicating on different computers

The socket API is quite low-level and should be used only when you need a complete control of the network access. If your application simply needs, for instance, to retrieve data from a web server, there are much simpler and higher-level APIs.

1.Sending data to a peer using a socket

In order to reach a peer, a process must know its address. An address is a value that identifies a peer in a given network. There exists many different kinds of address families. For example, some of them allow reaching a peer using the file system on the computer. Some others enable communicating with a remote peer through a network.

The **sendto** system call allows to send data to a peer identified by its socket address through a given socket.

2. Receiving data from a peer using a socket

Operating systems allow assigning an address to a socket using the bind system call. This is useful when you want to receive messages from another program to which you announced your socket address. Once the address is assigned to the socket, the program can receive data from others using system calls

the **read** system call as the operating system provides a socket as a file descriptor.

3. connect: connecting a socket to a remote address

Operating systems enable linking a socket to a remote address so that every information sent through the socket will only be sent to this remote address, and the socket will only receive messages sent by this remote address. This can be done using the connect system call

4. Creating a new socket to communicate through a network

The most recent standardized technology used to communicate through a network is the IPv6 network protocol. In the IPv6 protocol, hosts are identified using *IPv6 addresses*. Modern operating systems allow IPv6 network communications between programs to be done using the socket API, just as we did in the previous sections.

A program can use the **socket** system call to create a new socket.

7.Differentiate between direct and indirect process communication.

- Inter process communication(IPC) means the processes to communicate with each other while they are running. IPC allows processes to synchronize their action without sharing the same address space.
- Messages provide basic communication capabilities between two processes. Interprocess communication is best provided by a message passing system. IPC is especially useful in a distributed environment where communicating processes may dwell on different computers connected with a network.

❖ **Message Passing:**

The key idea behind IPC is message passing. That means, one process sends a message and the other process receives it. So the message system should provide at least two operations as follow:

1. send message
2. receive message

For any two processes to communicate with each other a link must be established between them. This link may be unidirectional or bidirectional. Also, the messages can be of fixed size or variable size. The link between two processes can be implemented using direct communication or indirect communication.

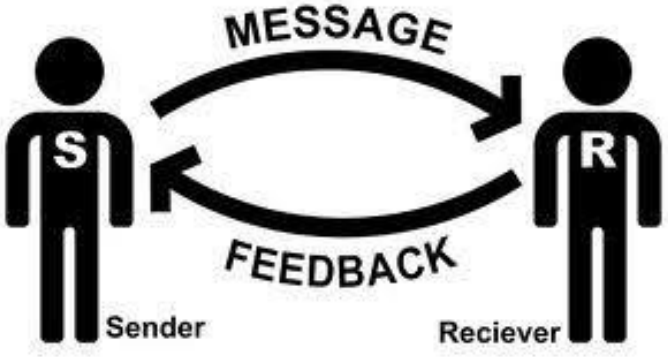
1. Direct Communication:

Here the process that wants to communicate with another process must explicitly state the name of the sender or the receiver. The send and receive primitives, in this case, are defined as follows:

- Send(a, message): send a message to process a
- Receive(b, message): receive a message from process b.

In direct communication,

- A pair of processes will have only one link associated with them.
- A link is automatically established between every pair of processes that want to communicate.
- The link is general, bi-directional.



2. Indirect Communication:

In this case, each pair of processes wants to communicate with each other needs to have a shared mailbox. The sender places the message in the mailbox also known as a port and the receiver removes the data or message from the box. In this case, the send and receive primitives are defined as follows:

- Send(p, message): send a message to mailbox p.
- Receive(q, message): receive a message from mailbox q.

In Indirect communication:

- A link is established between a pair of processes only if those processes have a shared mailbox.
- A link is associated with more than one pair.
- A link may be either unidirectional or bi-directional in nature.

7. Draw and explain Remote procedure communication (RPC) in detail

A remote procedure call is an interprocess communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call.

A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

The sequence of events in a remote procedure call are given as follows –

• The client stub is called by the client.

• The client stub makes a system call to send the message to the server and puts the parameters in the message.

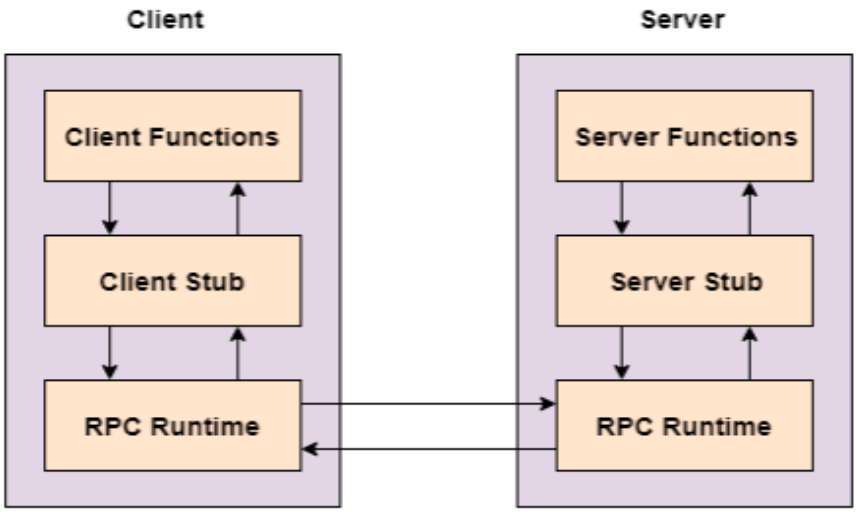
• The message is sent from the client to the server by the client's operating system.

• The message is passed to the server stub by the server operating system.

• The parameters are removed from the message by the server stub

. • Then, the server procedure is called by the server stub.

A diagram that demonstrates this is as follows –



Advantages of Remote Procedure Call

Some of the advantages of RPC are as follows –

• Remote procedure calls support process oriented and thread oriented models.

• The internal message passing mechanism of RPC is hidden from the user

.• The effort to re-write and re-develop the code is minimum in remote procedure calls.

• Remote procedure calls can be used in distributed environment as well as the local environment.

• Many of the protocol layers are omitted by RPC to improve performance.

Disadvantages of Remote Procedure Call

Some of the disadvantages of RPC are as follows –

• The remote procedure call is a concept that can be implemented in different ways. It is not a standard.

• There is no flexibility in RPC for hardware architecture. It is only interaction based.

• There is an increase in costs because of remote procedure call.

9. **Explain Multilevel Queue Scheduling with neat diagram**
   - Multilevel queue scheduling is used when processes in the ready queue can be divided into different classes where each class has its own scheduling needs.
   - For instance, foreground or interactive processes and background or batch processes are commonly divided.

Working:

For each class of processes, the ready queue is divided into separate queues.
For example, there are five processes:

1. System Processes
2. Interactive Processes
3. Interactive Editing Processes
4. Batch Processes
5. Student Process

Every queue will have an absolute priority over low priority queues. No process can run until the high priority queues are empty. For the above example, until and unless the queues for system processes, interactive processes, and interactive editing processes are all empty, no other process can run.

Advantages:

Multilevel Queue Scheduling has some advantages like:

1. Multilevel queue scheduling helps us apply different scheduling algorithms for different processes.
2. It will have a low scheduling overhead.

Disadvantages:

Some of the disadvantages of Multilevel queue scheduling are as follows:

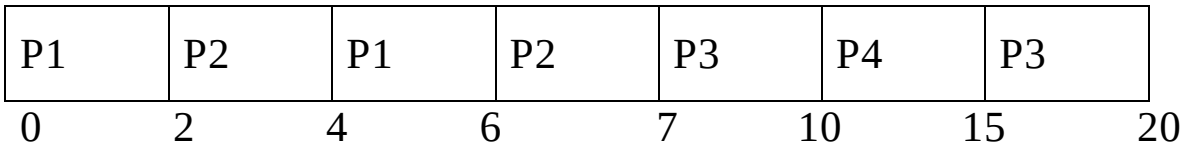
1. There are chances of starving for the lower priority processes.
2. It is inflexible in nature.

Example of Multilevel queue scheduling:

Let us consider the following 4 processes:

Process	Burst time	Arrival time	Queue number
P1	4	0	1
P2	3	0	1
P3	8	0	2
P4	5	10	1

The Gantt Chart will be as follows:



In the above example, P1 and P2 in queue-1 run first as they are of higher priority, they will follow the round-robin scheduling. After 7 units, P3 in queue-2 starts running as processes in queue-1 have completed. P4 comes and interrupts P3 and runs for 5 seconds after which the CPU is taken by P3 till it is fully executed.

UNIT NO-3: PROCESS SYNCHRONIZATION

1. Define Semaphore. List and explain the types of Semaphore.

- Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization.

The definitions of wait and signal are as follows −

- **Wait**
  The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

```
wait(S)
{
  while (S<=0);

  S--;
}
```

- **Signal**
  The signal operation increments the value of its argument S.

```
signal(S)
{
  S++;
}
```

Types of Semaphores

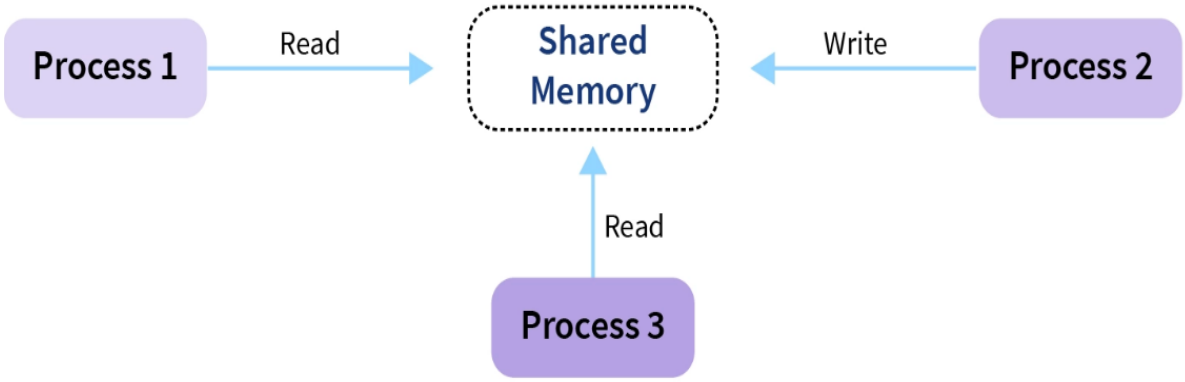
There are two main types of semaphores i.e. counting semaphores and binary semaphores. Details about these are given as follows −

- **Counting Semaphores**
  These are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, 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.

- **Binary Semaphores**
  The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0. It is sometimes easier to implement binary semaphores than counting semaphores.

2. List and explain in brief classical problems of process synchronization.

- Processes Synchronization or Synchronization is the way by which processes that share the same memory space are managed in an operating system. It helps maintain the consistency of data by using variables or hardware so that only one process can make changes to the shared memory at a time.



The classical problems of synchronization are as follows:

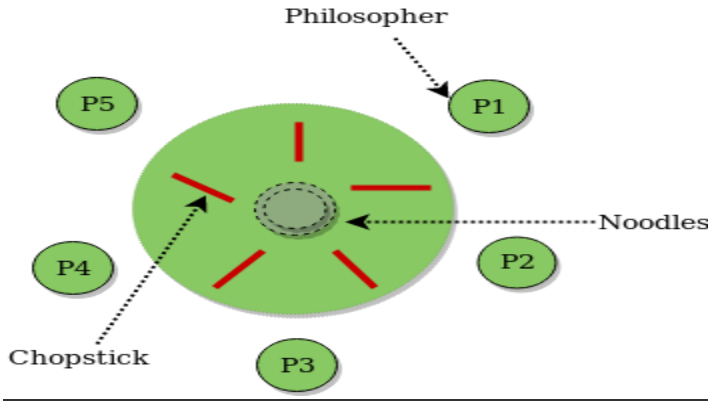
1. Bound-Buffer problem
2. Dining Philosophers problem
3. Readers and writers problem

1. Bound-Buffer problem:

Bounded Buffer problem is also called producer consumer problem. This problem is generalized in terms of the Producer-Consumer problem. Solution to this problem is, creating two counting semaphores "full" and "empty" to keep track of the current number of full and empty buffers respectively. Producers produce a product and consumers consume the product, but both use of one of the containers each time.

2. **Dining-Philosophers Problem:**

The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pickup the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both. This problem involves the allocation of limited resources to a group of processes in a deadlock-free and starvation-free manner.



3. Readers and Writers Problem:

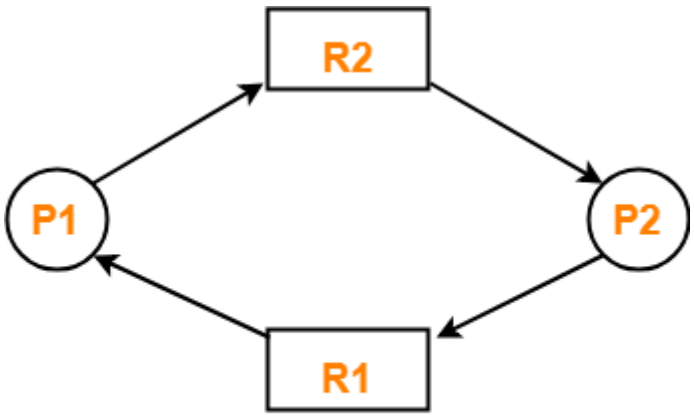
Suppose that a database is to be shared among several concurrent processes. Some of these processes may want only to read the database, whereas others may want to update (that is, to read and write) the database. We distinguish between these two types of processes by referring to the former as readers and to the latter as writers. Precisely in OS we call this situation as the readers-writers problem. Problem parameters:

- One set of data is shared among a number of processes.
- Once a writer is ready, it performs its write. Only one writer may write at a time.
- If a process is writing, no other process can read it.
- If at least one reader is reading, no other process can write.
- Readers may not write and only read.

3. List and explain necessary and sufficient conditions of deadlock

- The execution of two or more processes is blocked because each process holds some resource and waits for another resource held by some other process.

Example-



Example of a deadlock

Here

- Process P1 holds resource R1 and waits for resource R2 which is held by process P2.
- Process P2 holds resource R2 and waits for resource R1 which is held by process P1.
- None of the two processes can complete and release their resource.
- Thus, both the processes keep waiting infinitely.

❖ **Conditions For Deadlock-**

There are following 4 necessary conditions for the occurrence of deadlock-

1. Mutual exclusion
2. Hold and Wait
3. No-preemption
4. Circular wait

1. **Mutual Exclusion:** When two people meet in the landings, they can't just walk through because there is space only for one person. This condition allows only one person (or process) to use the step between them (or the resource) is the first condition necessary for the occurrence of the deadlock.

2. **Hold and Wait:** When the two people refuse to retreat and hold their ground, it is called holding. This is the next necessary condition for deadlock.

3. **No Preemption:** For resolving the deadlock one can simply cancel one of the processes for other to continue. But the Operating System doesn't do so. It allocates the resources to the processors for as much time as is needed until the task is completed. Hence, there is no temporary reallocation of the resources. It is the third condition for deadlock.

4. **Circular Wait:** When the two people refuse to retreat and wait for each other to retreat so that they can complete their task, it is called circular wait. It is the last condition for deadlock to occur.

4. Discuss solution for critical section problem.

- ❖ The critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical sections.
- ❖ The critical section problem is used to design a set of protocols which can ensure that the Race condition among the processes will never arise.
- ❖ In order to synchronize the cooperative processes, our main task is to solve the critical section problem. We need to provide a solution in such a way that the following conditions can be satisfied.

❖ **Solution to the Critical Section Problem**

The critical section problem needs a solution to synchronize the different processes. The solution to the critical section problem must satisfy the following conditions −

- **Mutual Exclusion**
  Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.
- **Progress**
  Progress means that if a process is not using the critical section, then it should not stop any other process from accessing it. In other words, any process can enter a critical section if it is free.

- **Bounded Waiting**
  Bounded waiting means that each process must have a limited waiting time. It should not wait endlessly to access the critical section.

5. Define the term 'Monitor' in process synchronization.

- ❖ The monitor is one of the ways to achieve Process synchronization. The monitor is supported by programming languages to achieve mutual exclusion between processes.
- ❖ It is the collection of condition variables and procedures combined together in a special kind of module or a package.
- ❖ The processes running outside the monitor can't access the internal variable of the monitor but can call procedures of the monitor.
- ❖ Only one process at a time can execute code inside monitors

- ❖ **Condition Variables:**
  Two different operations are performed on the condition variables of the monitor.

  1. Wait operation
  x.wait() : Process performing wait operation on any condition variable are suspended. The suspended processes are placed in block queue of that condition variable.

  2. Signal operation
  x.signal(): When a process performs signal operation on condition variable, one of the blocked processes is given chance.

- ❖ **Components of Monitor:**
  There are four main components of the monitor:

1. Initialization
2. Private data
3. Monitor procedure
4. Monitor entry queue

Initialization: - Initialization comprises the code, and when the monitors are created, we use this code exactly once.

Private Data: - Private data is another component of the monitor. It comprises all the private data, and the private data contains private procedures that can only be used within the monitor. So, outside the monitor, private data is not visible.

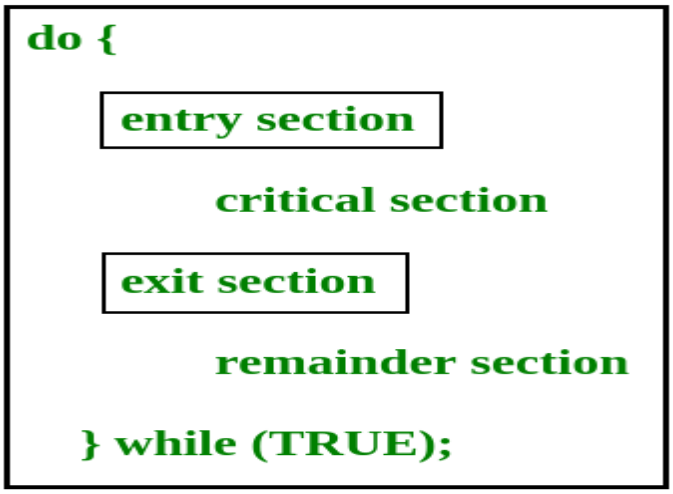
Monitor Procedure: - Monitors Procedures are those procedures that can be called from outside the monitor.

Monitor Entry Queue: - Monitor entry queue is another essential component of the monitor that includes all the threads, which are called procedures.

6. Why there is need of synchronization? What is critical section problem?

❖ Need of synchronization:

The critical section contains shared variables that need to be synchronized to maintain the consistency of data variables. So the critical section problem means designing a way for cooperative processes to access shared resources without creating data inconsistencies.



In the entry section, the process requests for entry in the **Critical Section.**
Any solution to the critical section problem must satisfy three requirements:

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

❖ Critical Section Problem:

- Critical Section is the part of a program which tries to access shared resources. That resource may be any resource in a computer like a memory location, Data structure, CPU or any IO device.
- The critical section cannot be executed by more than one process at the same time; operating system faces the difficulties in allowing and disallowing the processes from entering the critical section.
- The critical section problem is used to design a set of protocols which can ensure that the Race condition among the processes will never arise.
- In order to synchronize the cooperative processes, our main task is to solve the critical section problem. We need to provide a solution in such a way that the following conditions can be satisfied.

UNIT 4 : DEADLOCK

1. Consider a system with five processes Po through P4 and three resource types A, B, and C. Resource type A has ten instances, resource type B has five instances, and resource type C has seven instances. Suppose that, at time T0, the following snapshot of the systemhas been taken:

	Allocation	Max	Available
	A B C	A B C	A B C
P0	0 1 0	7 5 3	3 3 2
P1	2 0 0	3 2 2	
P2	3 0 2	9 0 2	
P3	2 1 1	2 2 2	
P4	0 0 2	4 3 3	

Calculate need matrix using Banker's algorithm. Justify safe sequence if the system is in safe state.

2. List and explain necessary and sufficient conditions of deadlock.
Same answer in chapter 3 Question3

3. Explain resource allocation graph with suitable diagram.

- The resource allocation graph is the pictorial representation of the state of a system. As its name suggests, the resource allocation graph is the complete information about all the processes which are holding some resources or waiting for some resources.
- It also contains the information about all the instances of all the resources whether they are available or being used by the processes.

Representation of Resource Allocation Graph (RAG)

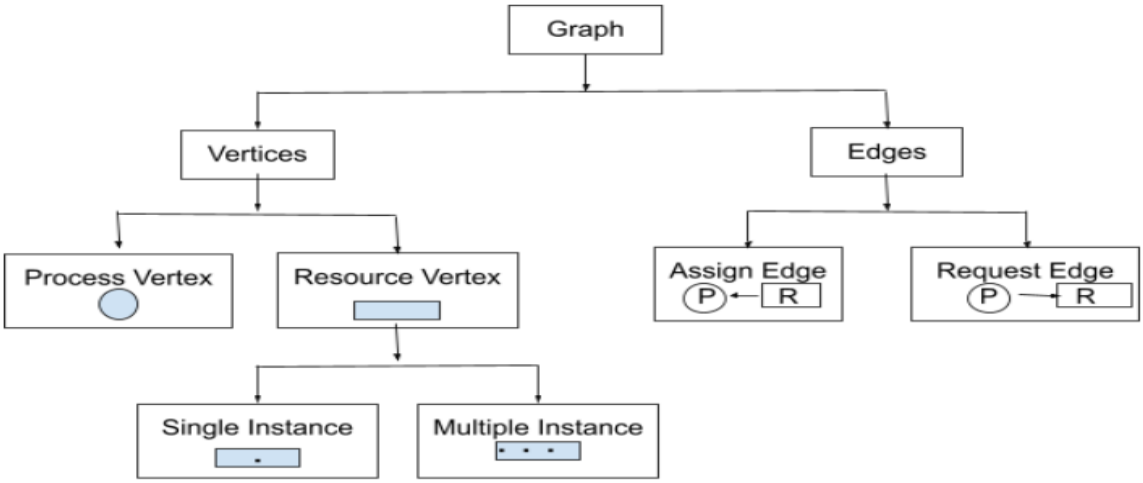
- Like all other graphs, it also has vertices and edges.
- The vertices of the RAG represent the process or the resource, and the edges represent the allocation or request of any resource.
- As both the process and resource are the vertices of the graph, generally, the process vertex will be represented using the circle and the resource vertex using the rectangle.
- The resource can be of two types:

1. **Single Instance** - It contains the single instance of the resource, and is represented by a single dot inside the rectangle which is the resource vertex.
2. **Multiple Instance** - It contains the multiple instances of the resource, and is represented by multiple (more than one) dots inside the rectangle.

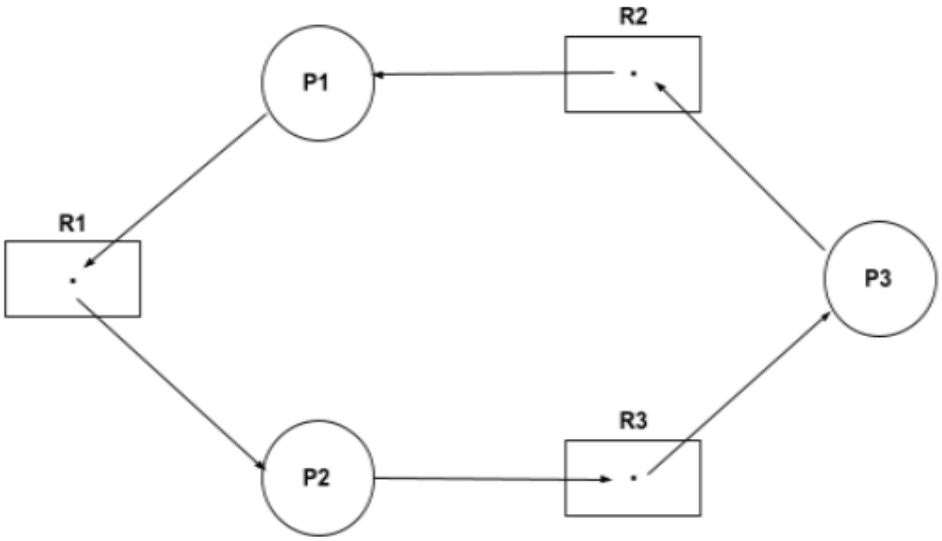
- Edges also can be of two types:

1. **Assign Edge** - This edge directing from the resource towards the process represents the allocation of the resource to the process.

2. **Request Edge** - This edge directing from the process towards the resource represents the request of the resource by the process.

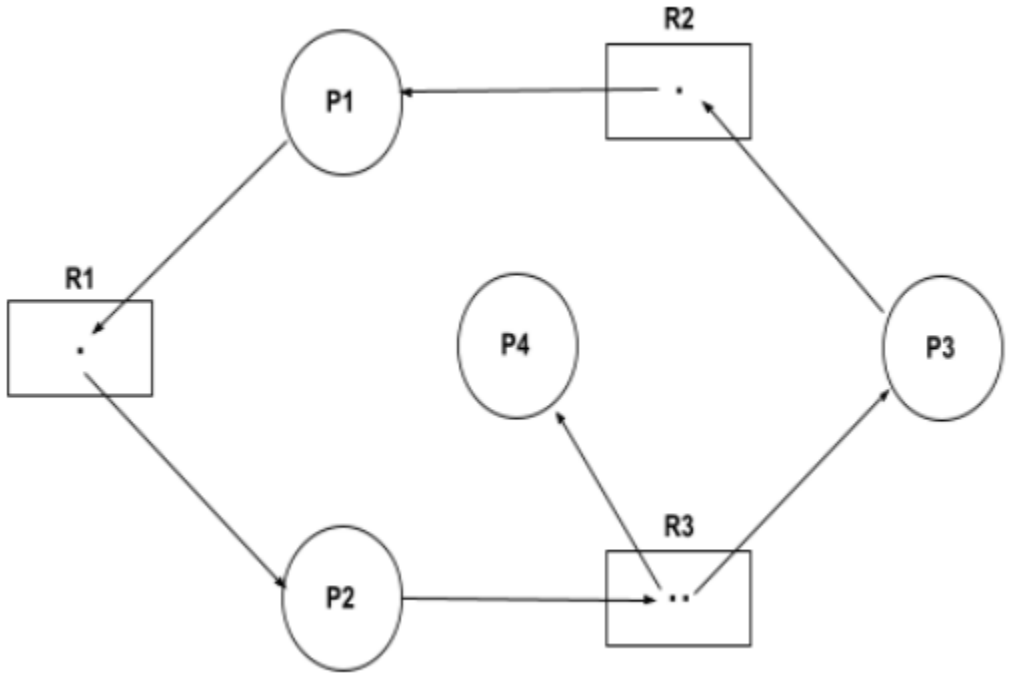


Example of Single Instances RAG



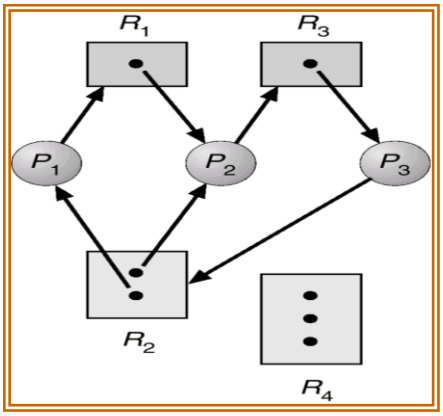
In the above single instances RAG example, it can be seen that P2 is holding the R1 and waiting for R3. P1 is waiting for R1 and holding R2 and, P3 is holding the R3 and waiting for R2. So, it can be concluded that none of the processes will get executed.

Example of Multiple Instances RAG



In the case of Multiple Instances RAG, it becomes difficult to analyze from the RAG that the system is in a safe state or not.

4. Consider following resource allocation graph and justify if / if not deadlock occurs.



The resource allocation graph contains a **deadlock** .

The reason for the deadlock is

Process P1 holding one instance of resource R2 requests for resource R1 held by process P2.

Process P2 holding resource R1 and one instance of resource R2 is requesting for resource R3 held by process P3.

Process P3 holding resource R3 is requesting resource R2 held by process P1 and P2.

Hence there is a cycle and there exists a deadlock as one process holding one resource is waiting for another resource held by another process which in turn is waiting for a resource held by another process (cycle goes on).

5) Differentiate between deadlock prevention and deadlock avoidance.

SR.NO	FACTORS	DEADLOCK PREVENTION	DEADLOCK AVOIDANCE
1.	Concept	It blocks at least one of the conditions necessary for deadlock to occur.	It ensures that system does not go in unsafe state
2.	Resource Request	All the resources are requested together.	Resource requests are done according to the available safe path.
3	Information required	It does not requires information about existing resources, available resources and resource requests	It requires information about existing resources, available resources and resource requests
4	Procedure	It prevents deadlock by constraining resource request process and handling of resources	It automatically considers requests and check whether it is safe for system or not.
5	Preemption	Sometimes, preemption occurs more frequently.	In deadlock avoidance there is no preemption.
6	Resource allocation strategy	Resource allocation strategy for deadlock prevention is conservative.	Resource allocation strategy for deadlock prevention is not conservative.
7	Future resource requests	It doesn't requires knowledge of future process resource requests.	It requires knowledge of future process resource requests.
8	Advantage	It doesn't have any cost involved because it has to just make one of the conditions false so that deadlock doesn't occur.	There is no system under-utilization as this method works dynamically to allocate the resources.
9	Disadvantage	Deadlock prevention has low device utilization.	Deadlock avoidance can block processes for too long.
10	Example	Spooling and non-blocking synchronization algorithms are used.	Banker's and safety algorithm is used.