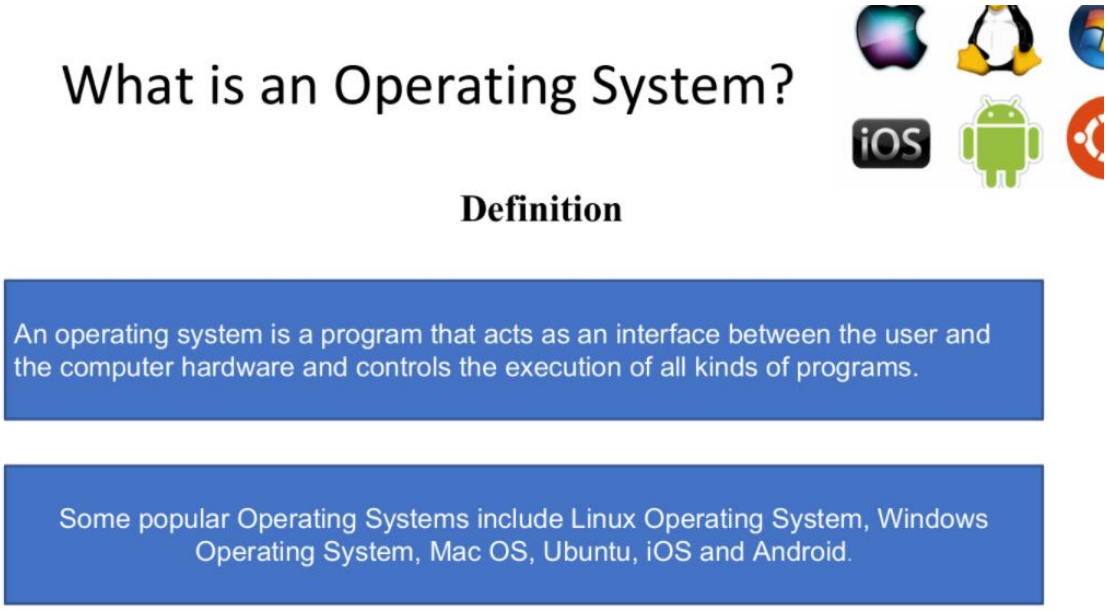
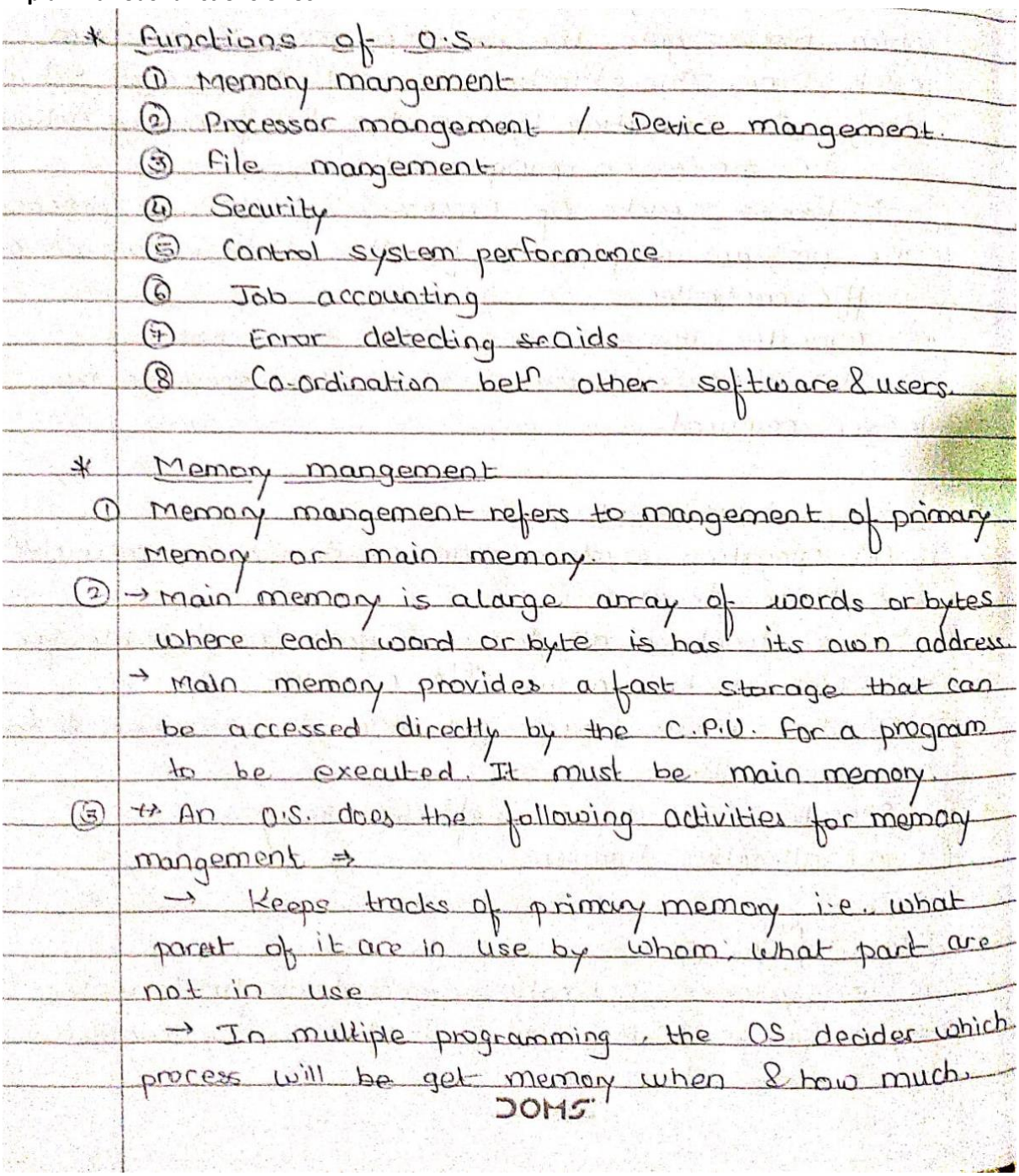
Os QB ans:

1. **Define Operating System with the help of examples:**

Ans:



2. **Explain various functions of os**



* Functions of O.S.
① Memory mangement
② Processor mangement / Device mangement.
③ File mangement
④ Security
⑤ Control system performance
⑥ Job accounting
⑦ Error detecting aids
⑧ Co-ordination bet^n other software & users.

* Memory mangement
① Memory mangement refers to mangement of primary Memory or main memory.
② → Main memory is a large array of words or bytes where each word or byte is has its own address.
→ Main memory provides a fast storage that can be accessed directly by the C.P.U. for a program to be executed. It must be main memory.
③ → An O.S. does the following activities for memory mangement ⇒
→ Keeps tracks of primary memory i.e. what parest of it are in use by whom, what part are not in use
→ In multiple programming, the OS decider which process will be get memory when & how much.
DOMS

→ Allocates the memory when a process request it to do so.
→ De allocates the memory when a process no longer needs it or has been terminated.

* Processor management.
• In multiple programming environment, the OS decides which process gets the processor when & for how much time. This function is called process scheduling. An operating system does the following activites for processor mangement.
→ keeps tracks of processor & status of process. The program responsible for this task is known as traffic controller.
→ Allocates the processor (CPU) to a process.
→ De-allocates processor when a process is no longer required.

* Device management.
① An operating system manages devices communicatn via their drivers.
② Keeps tracks of all devices. Program responsible for this task is known as I/O controller.
③ Decides which process gets the device when & for how much time
④ Allocates the device in efficient way.
⑤ De-allocates devices.

* File management
① A file system is normally organized into directories for easy navigation & usage. These directionies may contain files & other directories

② Keeps track of information, location, uses, status etc
The collective facilities are often known as file sys
③ Decides who get resources.
④ Allocates the resources.
⑤ De-allocates the resources.

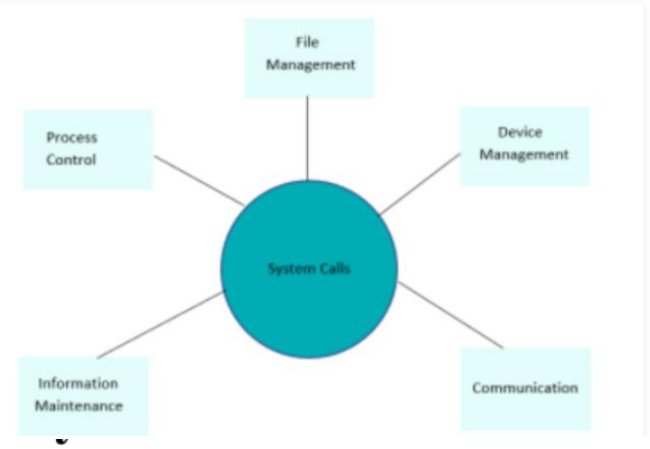
* Other Functions

- Security - By means of password & similar other techniques, it prevents unathorized access to programs & data.
- Control over System performance :- Recording delays between request for a service & responce from the system.
- Job accounting :- Keeping track of time & resource used by various jobs & users.
- Error detecting aids :- production of dumps, traces, error messages, & other debugging & error detectin aids.
- Coordination between other softwares & users :- Coordination and assignment of compilers, interprete assemblers and other software to the various user of the computer systems.

3. Explain different types of system calls with the help of an example

System call

- **System calls are divided into 5 categories mainly :**
- Process Control
- File Management
- Device Management
- Information Maintenance
- Communication



1. Process Control

- This system calls perform the task of process creation, process termination, etc.
- **Functions:**
- End and Abort
- Load and Execute
- Create Process and Terminate Process
- Wait and Signal Event
- Allocate and free memory

2. File Management

- File management system calls handle file manipulation jobs like creating a file, reading, and writing, etc.
- **Functions:**
- Create a file
- Delete file
- Open and close file
- Read, write, and reposition
- Get and set file attributes

3. Device Management

- Device management does the job of device manipulation like reading from device buffers, writing into device buffers, etc.
- **Functions:**
- Request and release device
- Logically attach/ detach devices
- Get and Set device attributes

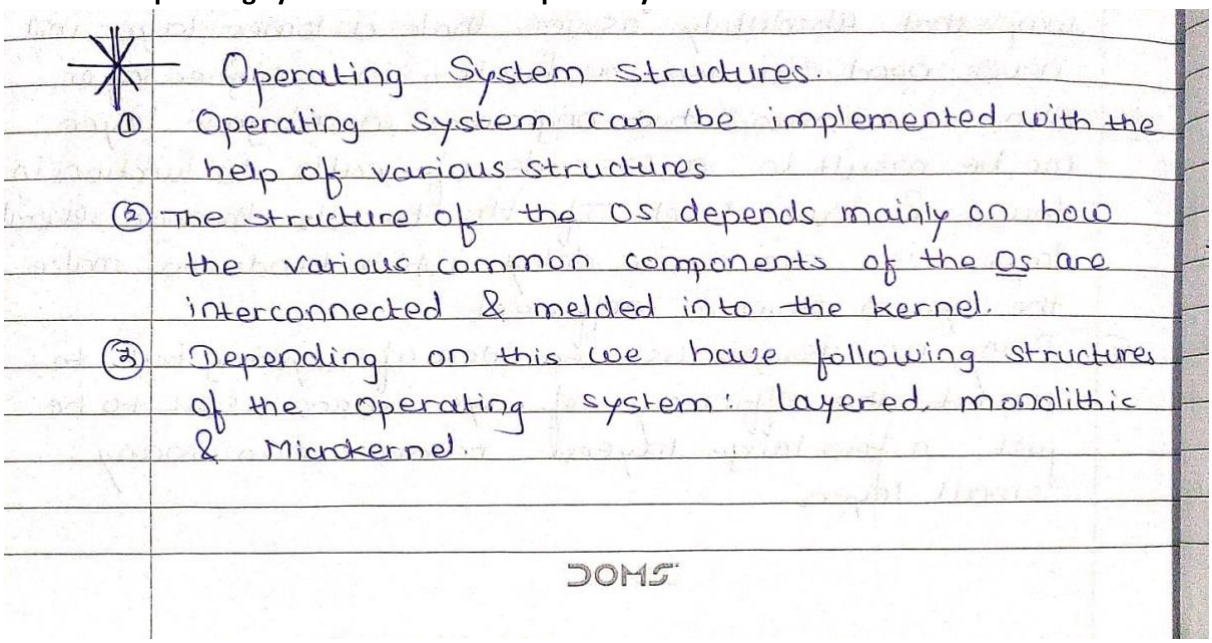
4. **Information Maintenance**

- It handles information and its transfer between the OS and the user program.
- **Functions:**
- Get or set time and date
- Get process and device attributes

5. Communication:

- These types of system calls are specially used for interprocess communications.
- **Functions:**
- Create, delete communications connections
- Send, receive message
- Help OS to transfer status information
- Attach or detach remote devices

4. List various operating system structures & explain any one in brief.



Operating System structures.

① Operating system can be implemented with the help of various structures

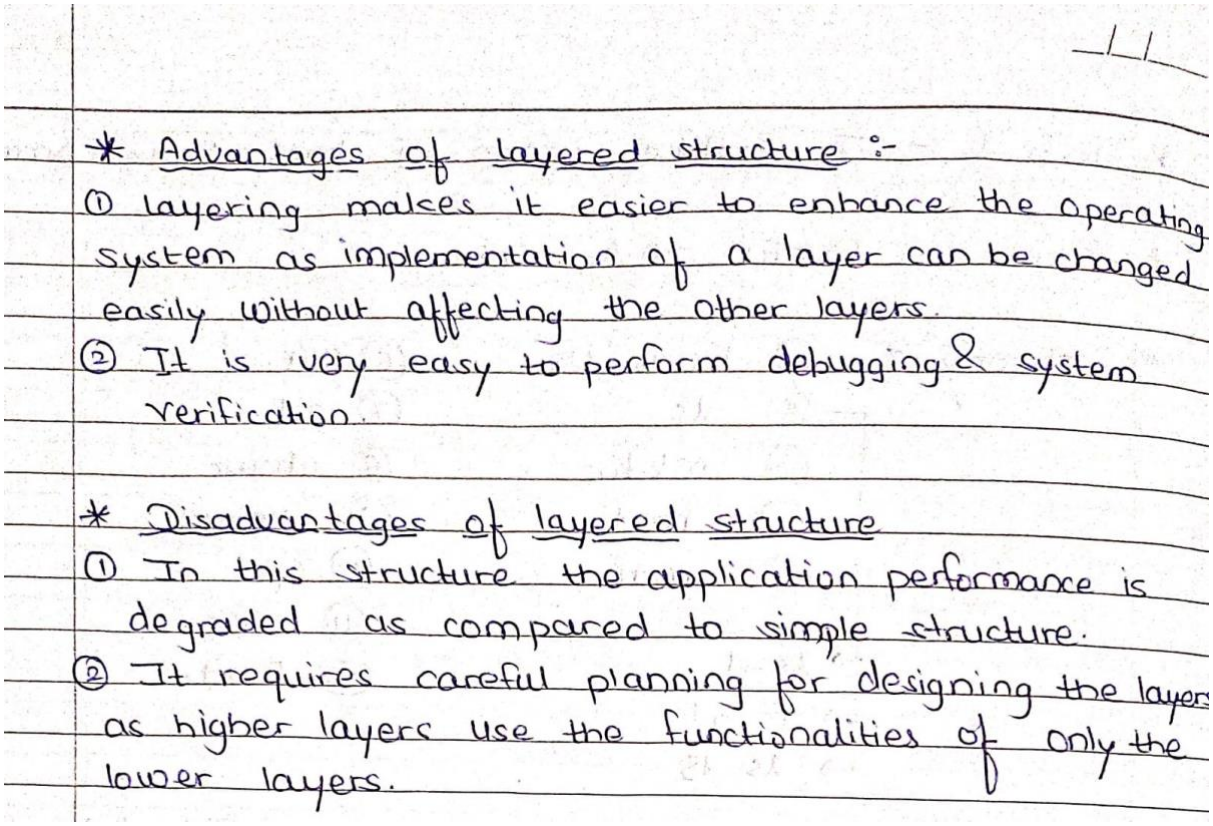
② The structure of the OS depends mainly on how the various common components of the OS are interconnected & melded into the kernel.

③ Depending on this we have following structures of the operating system: layered, monolithic & Microkernel.

DOMS

* Layered Approach

1. In a layered approach, we make implementation & modification easier by designing the OS as a set of very distinct modules called layers.
2. Each layer is a well-defined abstract data structure (aka. object) with specific data members & operations (methods)
3. The design of the layers requires that they be numbered from lowest to highest, with the lowest being the hardware & the highest being the user interface.
4. Code in one layer, can make calls to functions in other layers but only if those funct^n's are in same or lower levels.
5. Because of the rules, once the system is designed, it's relatively easy to build without ~~letys~~ errors. starting with the hardware & continuing with the next highest layer only after the current layer has been implemented and tested.
6. It can be quite difficult to design OS layers in a way that absolutely assures that a lower layer will never need to use a function in a higher layer
7. Also, a command that originates in a higher layer can be result in a cascade of calls to functions in lower & lower levels. If this travels through several layers, it can cause delay which tends to make the system slow & inefficient.
8. Many OS designs use the idea of layering but, to protect the efficiency of sys. there tend to be just a few large layers, rather than many small layers.



* Advantages of layered structure :-

① layering makes it easier to enhance the operating system as implementation of a layer can be changed easily without affecting the other layers.

② It is very easy to perform debugging & system verification.

* Disadvantages of layered structure

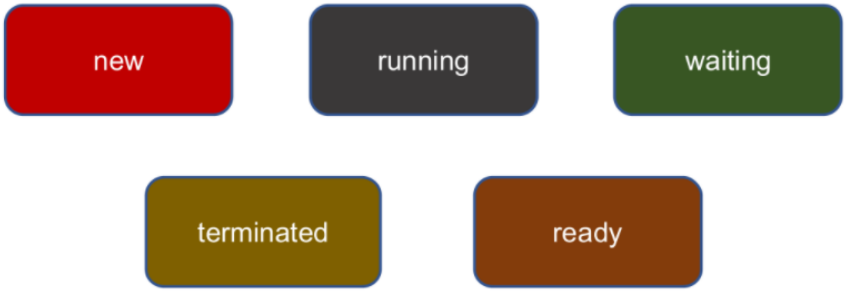
① In this structure the application performance is degraded as compared to simple structure.

② It requires careful planning for designing the layers as higher layers use the functionalities of only the lower layers.

5. Explain various states of a process

Process States

- As a process executes, it changes state. The state of a process refers to what the process currently does. A process can be in one of the following states during its lifetime:



- new: The process is being created.
- running: Instructions are being executed.
- waiting: The process is waiting for some event to occur.
- ready: The process is waiting to be assigned to a processor.
- terminated: The process has finished execution.

6. Explain semaphores and signals with respect to IPC

- The following are a few important terms used in IPC:
- **Semaphores:** A semaphore is a signaling mechanism technique. This OS method either allows or disallows access to the resource, which depends on how it is set up.
- **Signals:** It is a method to communicate between multiple processes by way of signaling. The source process will send a signal which is recognized by number, and the destination process will handle it

7. Write a short note on queues, mailboxes

Message queues :

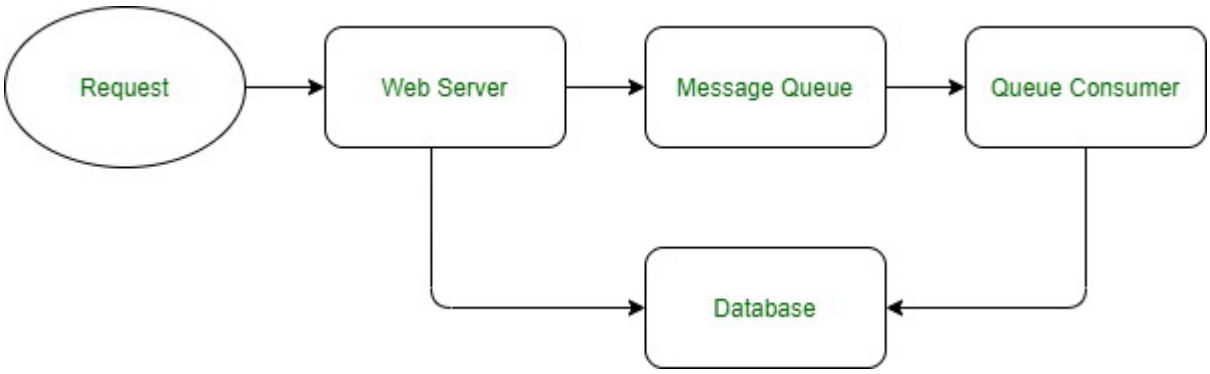
1. Message queuing makes it feasible for programs to talk asynchronously, through sending messages to every different through a queue. A message queue presents brief storage among the sender and the receiver in order that the sender can preserve running without interruption while the vacation spot application is busy or now no longer connected. Asynchronous processing lets in a mission to name a provider, and flow directly to the subsequent mission whilst the provider processes the request at its very own pace.

2. A queue is a line of factors ready to be handled — in sequential order beginning at the start of the line. A message queue is a queue of messages sent amongst applications. It consists of a series of labor items that are ready to be processed.

3. Data is transferring between sender and receiver application, and this is called a message. And we can say that it is a byte array with some headers on top.

4. The fundamental structure of a message queue is simple: there are purchaser programs known as manufacturers that create messages and supply them to the message queue. Another application, known as a customer, connects to the queue and receives the messages to be processed. Messages positioned onto the queue are saved till the customer retrieves them. The queue can offer safety from carrier outages and failures.

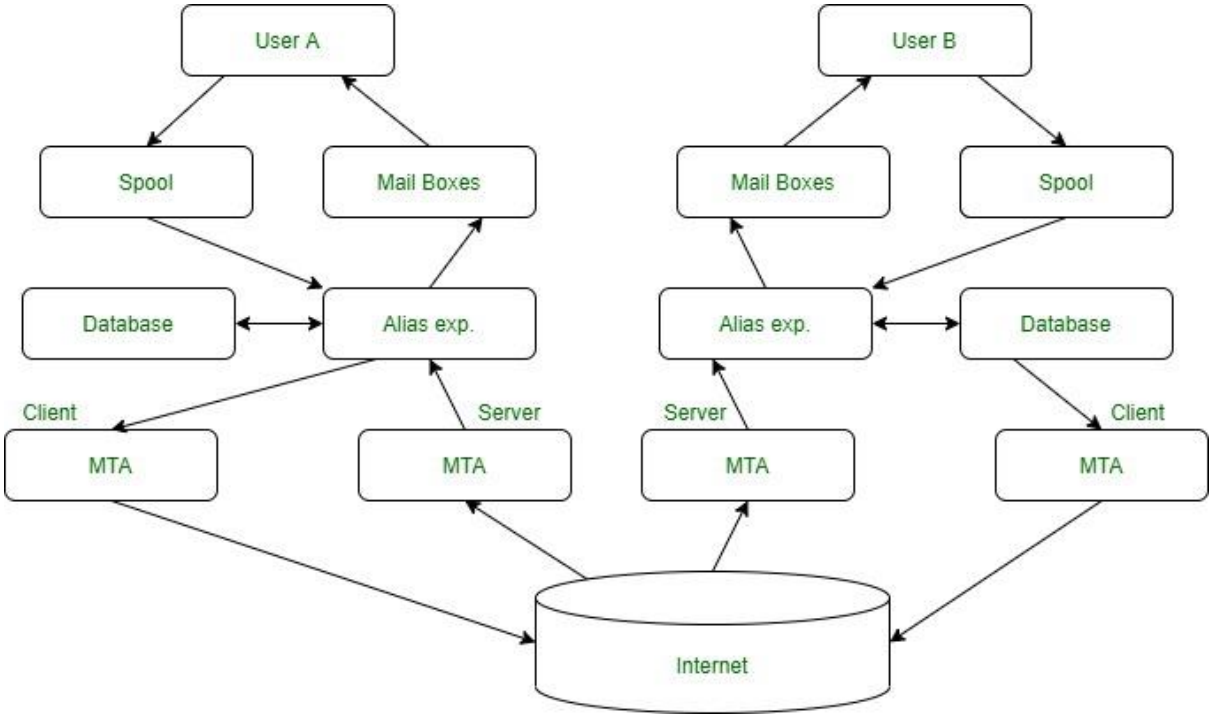
5. Some common examples of queues are Kafka, Heron, real-time streaming, Amazon SQS, and RabbitMQ.



Mailboxes :

- The mailbox is an easy mechanism for asynchronous communication. Some architectures outline mailbox registers. These mailboxes have a hard and fast range of bits and may be used for small messages. We also can enforce a mailbox the usage of P() and V() the usage of essential reminiscence for the mailbox storage. A quite simple model of a mailbox, one which holds the simplest one message at a time, illustrates a few crucial standards in interprocess communication.

- In order for the mailbox to be maximum useful, we need it to incorporate items: the message itself and a mail prepared flag. The flag is real at the same time as a message has been located in the mailbox and cleared at the same time as the message is removed.



8. Explain Round Robin with the help of an example

➜ 2.16.2(D) Round Robin Scheduling

Q. With the help of example, explain Round Robin Scheduling algorithm.

- Round Robin Scheduling is designed especially for time-shari ng systems where many processes get CPU on time sharing basis.
- In this algorithm, a small unit of time called time quantum is defined.
- CPU is allocated to each process for this time quantum period of time. To implement this scheduling, ready queue treated as FIFO queue.
- The new processes go to the tail of queue and each time CPU chooses the process from head of queue.
- When time quantum expires, context switch occurs and CPU switches to other process which is scheduled next.
- The time quantum is fixed and then processes are scheduled such that no process get CPU time more than one time quantum.
- If process is executing and request for I/O the process goes in waiting (blocked) state.
- After the completion of I/O, process again gets added at the tail of ready queue. The time quantum should not be very small or very large.
- If the time quantum is very large, the algorithm will behave just like FCFS.
- A smaller time quantum increases context switches leading to performance degradation (less throughput) as context switches elapses more time.

operating System (MU - Sem -

- Consider the following example ... If we ... quantum of 4 ms ... seconds then the Gantt chart shows the result.

proce**	gural lime
PI	11
P2	7
P3	15
P4	4

PJ	P2	P3	P4	PI	P2	P3	PI	P3	P3

0 4 8 J2 1* 20 23 27 30 34 3?

Turnaround time for P1 = (30-0) = 30

Turnaround lime for P2 = (23-0) = 23

Turnaround rime for P3 = (37 - 0) = 37

Tur7iarou „d time for P4 = (16-0) = 16

Average Turnaround time = (∘ + 23 + 37 +]6) / 4 = 26.5

Waiting time for PI - 0 + ,16 – 4) + (27 – 20) = 19

Waiting time for P2 = 4 + (20-8) = 16

Waiting time for P3 = 8 + (23- 12)*(3O-27)-22

Waiting time for P4 = 1 2

Average Waiting time = (19 + 4 + 22 + 12)/4 - 14.25

8. Explain the concept of a process:

A process is a program in execution. The execution of a process progresses in a sequential fashion. A program is a passive entity while a process is an active entity. A process includes much more than just the program code.

A process is the unit of work in modern time-sharing systems. A system has a collection of processes — user processes as well as system processes. All these processes can execute concurrently with the CPU multiplexed among them.

A process includes the text section, stack, data section, program counter, register contents and so on. The text section consists of the set of instructions to be executed for the process.

The data section contains the values of initialized and uninitialized global variables in the program. The stack is used whenever there is a function call in the program. A layer is pushed into the stack when a function is called.

The arguments to the function and the local variables used in the function are put into the layer of the stack. Once the function call returns to the calling program, the layer of the stack is popped.

- The text, data and stack sections comprise the address space of the process. The program counter has the address of the next instruction to be executed in the process.
- It is possible to have two processes associated with the same program. For example, consider an editor program, say Microsoft Word.
- The program has the same text section. But, the data section will be different for each file that is opened in Microsoft Word, that is, each file has a different data section

9. **Explain Uniprocessor scheduling**:

- How does an operating system determine how much processing time a single process receives? There are a number of fundamental "scheduling policies" that an architect of an operating system may consider implementing:

Multiprogramming system

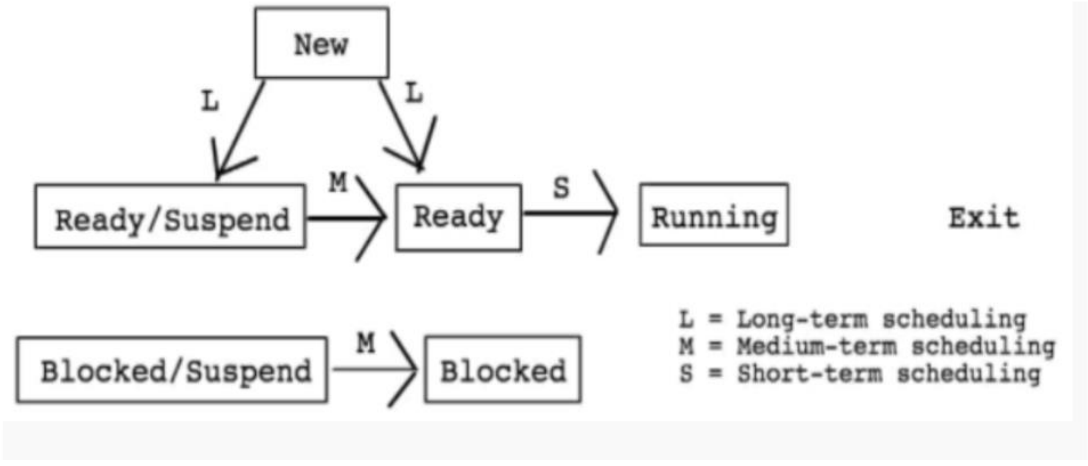
- A multiprogramming system is a basic form of parallel processing in which multiple programs are run at the same time on a uniprocessor. As a result of having only a single processor, concurrent execution of multiple programs is impossible. Rather, the operating system must alternate between the programs, e.g. giving some processing time to one process, then switching to a different process.

- Scheduling on most systems can be broken down into three separate tasks:

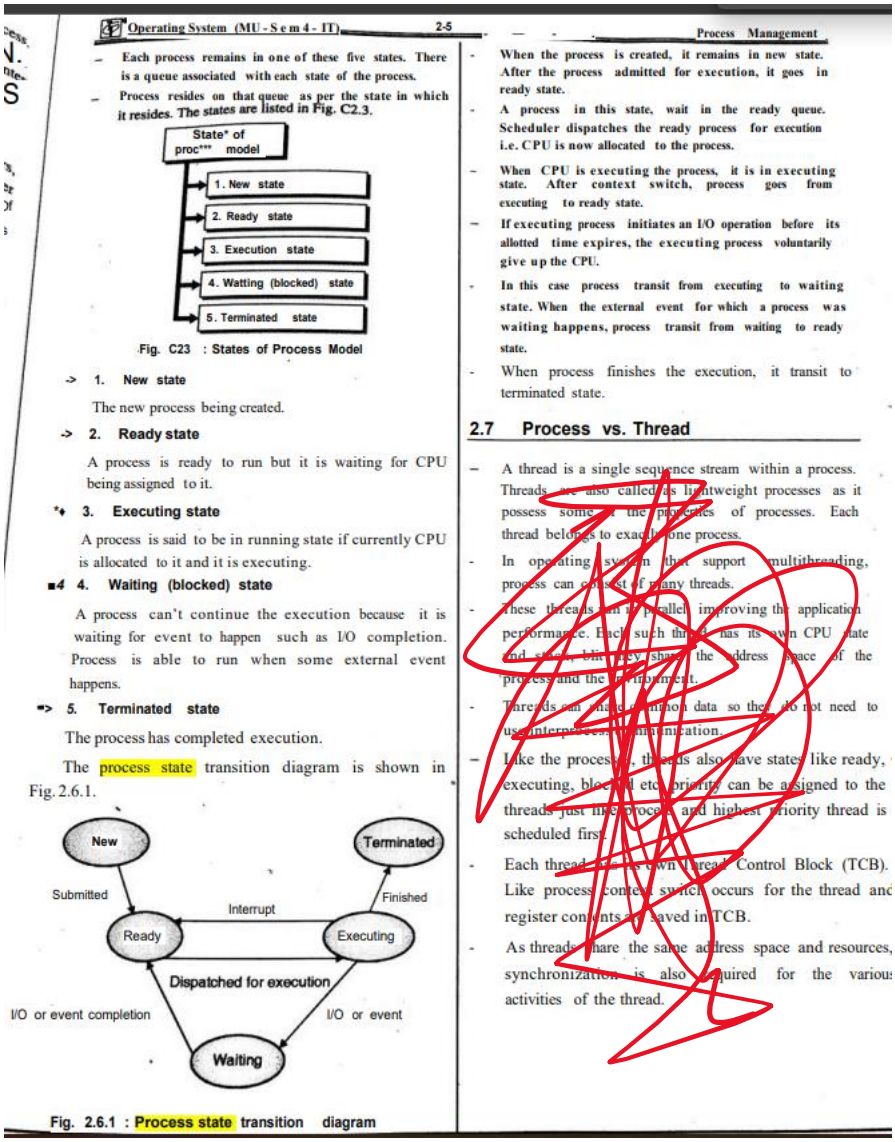
Long-term Scheduling – The decision to add a new process to the group of processes to be executed. The greater the number of processes in the system, the smaller amount of time an individual process can be executed; as a consequence, competition for the processor increases.

Medium-term Scheduling – The decision to add a process to the group of processes that are partially or fully in main memory. Part of the *swapping* function which exchanges the contents of an area of main memory with the contents of an area of secondary memory

Short-term Scheduling – The decision of which available process will be executed next by the processor. Invoked when the current process is at risk for being blocked or preempted which can be caused by clock interrupts, I/O interrupts, OS calls, or signals.



11. Draw and explain Process state transition diagram



Operating System (MU - Sem 4 - IT) 2-5

Process Management

- Each process remains in one of these five states. There is a queue associated with each state of the process.
- Process resides on that queue as per the state in which it resides. The states are listed in Fig. C2.3.

Fig. C23 : States of Process Model

-> 1. New state

The new process being created.

-> 2. Ready state

A process is ready to run but it is waiting for CPU being assigned to it.

3. Executing state

A process is said to be in running state if currently CPU is allocated to it and it is executing.

4. Waiting (blocked) state

A process can't continue the execution because it is waiting for event to happen such as I/O completion. Process is able to run when some external event happens.

=> 5. Terminated state

The process has completed execution.

The process state transition diagram is shown in Fig. 2.6.1.

Fig. 2.6.1 : Process state transition diagram

- When the process is created, it remains in new state. After the process admitted for execution, it goes in ready state.
- A process in this state, wait in the ready queue. Scheduler dispatches the ready process for execution i.e. CPU is now allocated to the process.
- When CPU is executing the process, it is in executing state. After context switch, process goes from executing to ready state.
- If executing process initiates an I/O operation before its allotted time expires, the executing process voluntarily give up the CPU.
- In this case process transit from executing to waiting state. When the external event for which a process was waiting happens, process transit from waiting to ready state.
- When process finishes the execution, it transit to terminated state.

2.7 Process vs. Thread

- A thread is a single sequence stream within a process. Threads are also called as lightweight processes as it possess some of the properties of processes. Each thread belongs to exactly one process.
- In operating system that support multithreading, process can consist of many threads.
- These threads run in parallel improving the application performance. Each such thread has its own CPU state and stack, but they share the address space of the process and the environment.
- Threads can share common data so they do not need to use interprocess communication.
- Like the processes, threads also have states like ready, executing, blocked etc. priority can be assigned to the threads just like process and highest priority thread is scheduled first.
- Each thread has its own Thread Control Block (TCB). Like process context switch occurs for the thread and register contents are saved in TCB.
- As threads share the same address space and resources, synchronization is also required for the various activities of the thread.

12. **List various scheduling policies. Explain Preemptive SJF with the help of an example**

Ans: 1. First-come-first-served (FCFS)

2.Round robin(RR)

- Shortest Job First (SJF) ...
- Multilevel Feedback Queues. ...
- Lottery Scheduling.

Scheduling Policies

Exactly when the selection function is run depends on whether or not the scheduling policy is ***preemptive*** **or** ***nonpreemptive***.

For a nonpreemptive policy, once a process is running, it continues to execute until it terminates or it blocks itself; it is not interrupted

In preemptive policies, a running process can be switched out for a different process before it finishes based on a particular scheduling policy's unique preemptive condition. A preemptive condition can be things like a clock interval or the arrival of a new process to the ready pool.

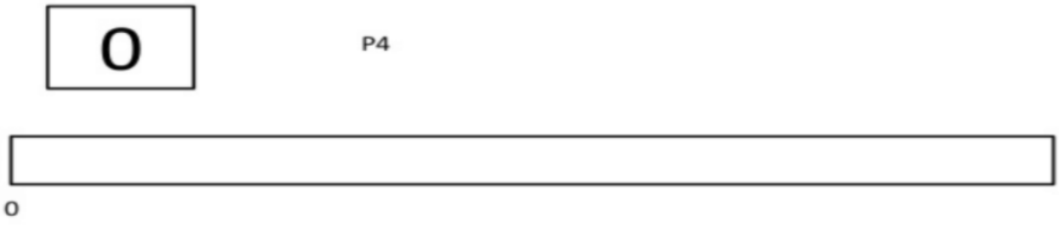
Preemptive SJF

- In Preemptive SJF Scheduling, jobs are put into the ready queue as they come. A process with shortest burst time begins execution. If a process with even a shorter burst time arrives, the current process is removed or preempted from execution, and the shorter job is allocated CPU cycle.
- Consider the following five process:

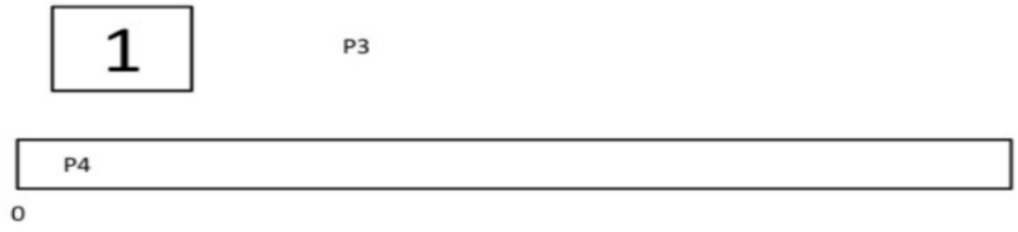
Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

Step 0) At time=0, P4 arrives and starts execution.

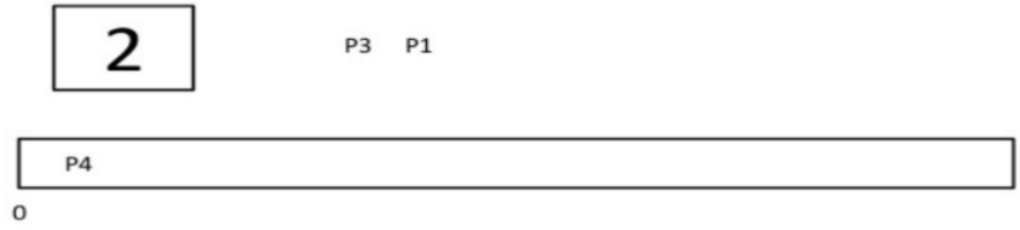
Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4



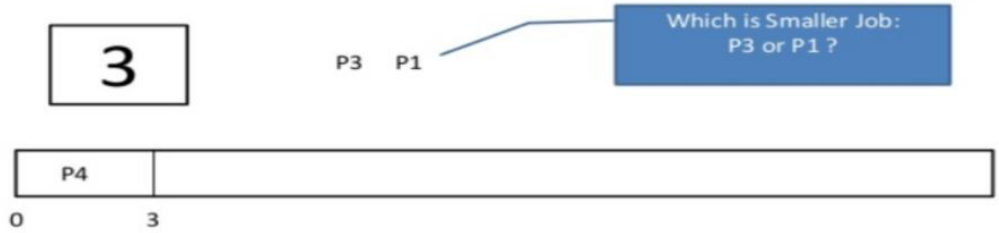
Step 1) At time= 1, Process P3 arrives. But, P4 has a shorter burst time. It will continue execution.



Step 2) At time = 2, process P1 arrives with burst time = 6. The burst time is more than that of P4. Hence, P4 wil continue execution.

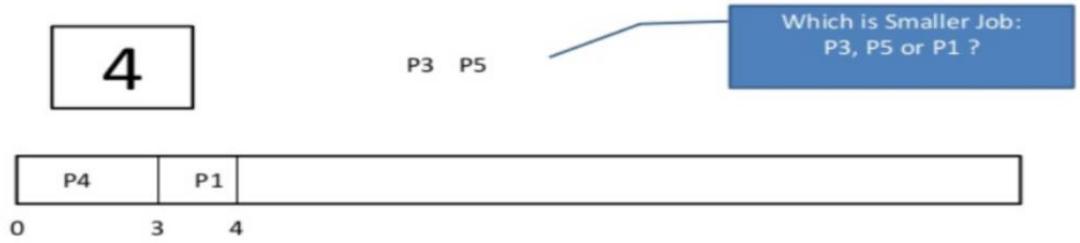


Step 3) At time = 3, process P4 will finish its execution. The burst time of P3 and P1 is compared. Process P1 is executed because its burst time is lower.



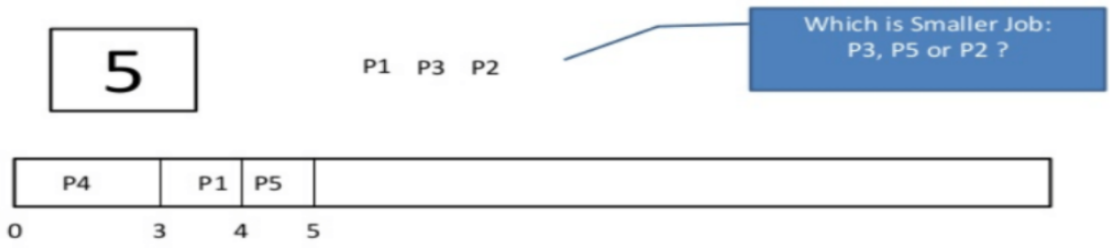
Step 4) At time = 4, process P5 will arrive. The burst time of P3, P5, and P1 is compared. Process P5 is executed because its burst time is lowest. Process P1 is preempted.

Process Queue	Burst time	Arrival time
P1	5 out of 6 is remaining	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

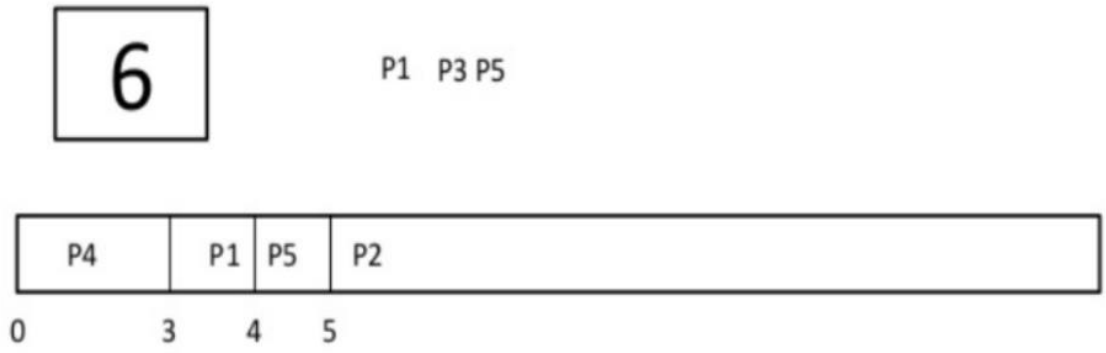


Step 5) At time = 5, process P2 will arrive. The burst time of P1, P2, P3, and P5 is compared. Process P2 is executed because its burst time is least. Process P5 is preempted.

Process Queue	Burst time	Arrival time
P1	5 out of 6 is remaining	2
P2	2	5
P3	8	1
P4	3	0
P5	3 out of 4 is remaining	4

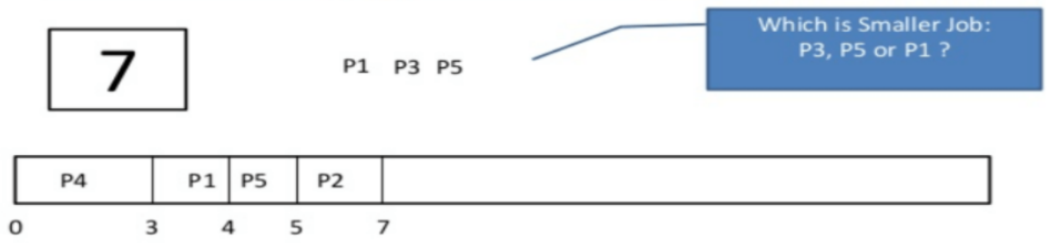


Step 6) At time =6, P2 is executing.

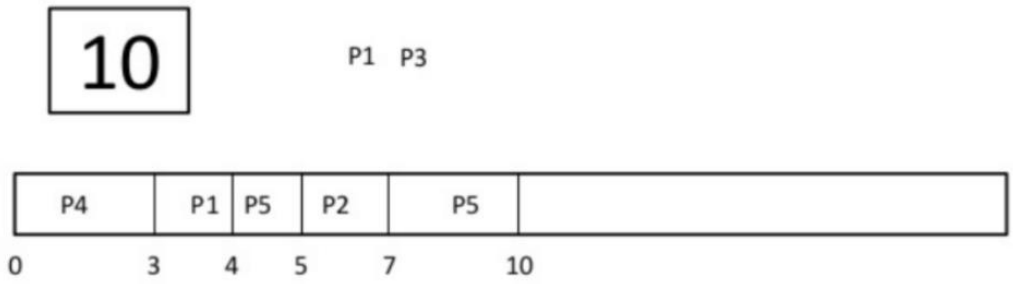


Step 7) At time =7, P2 finishes its execution. The burst time of P1, P3, and P5 is compared. Process P5 is executed because its burst time is lesser.

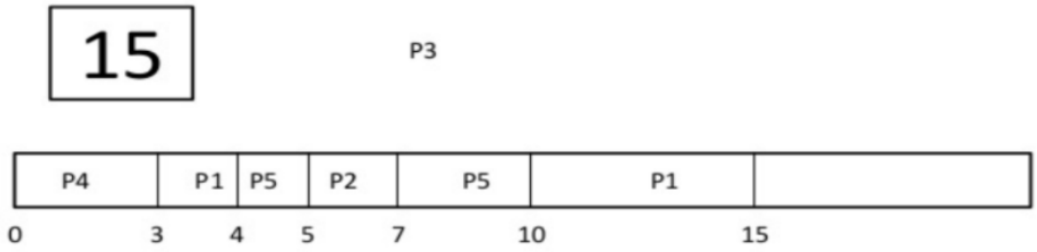
Process Queue	Burst time	Arrival time
P1	5 out of 6 is remaining	2
P2	2	5
P3	8	1
P4	3	0
P5	3 out of 4 is remaining	4



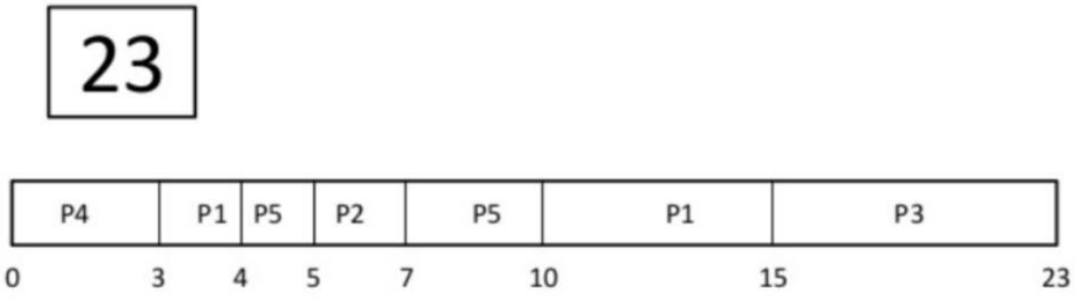
Step 8) At time =10, P5 will finish its execution. The burst time of P1 and P3 is compared. Process P1 is execute because its burst time is less.



Step 9) At time =15, P1 finishes its execution. P3 is the only process left. It will start execution.



Step 10) At time =23, P3 finishes its execution.



Step 11) Let's calculate the average waiting time for above example.

```
Wait time
P4= 0-0=0
P1=  (3-2) + 6 =7
P2= 5-5 = 0
P5= 4-4+2 =2
P3= 15-1 = 14
```

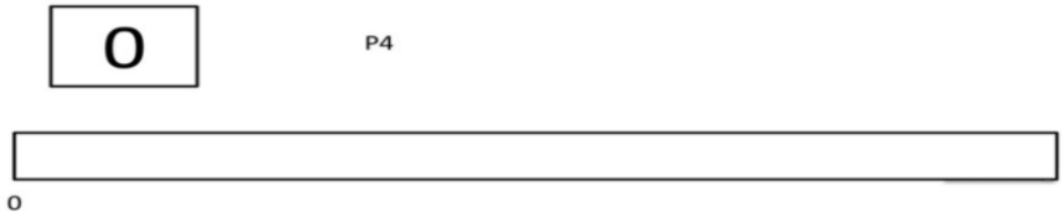
```
Average Waiting Time = 0+7+0+2+14/5 = 23/5 =4.6
```

13]List various scheduling policies. Explain Non Preemptive SJF with the help of an example

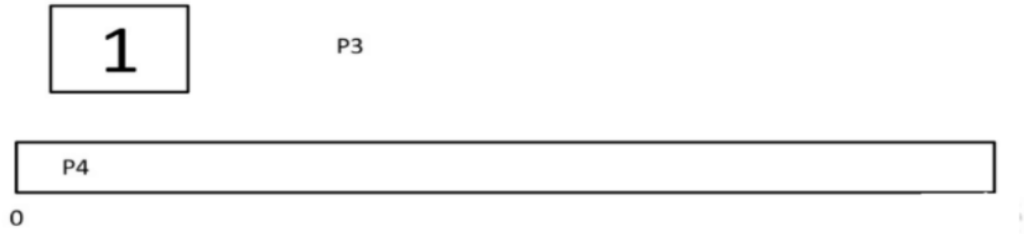
- **Non-Preemptive SJF**
- In non-preemptive scheduling, once the CPU cycle is allocated to process, the process holds it till it reaches a waiting state or terminated.
- Consider the following five processes each having its own unique burst time and arrival time.

Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

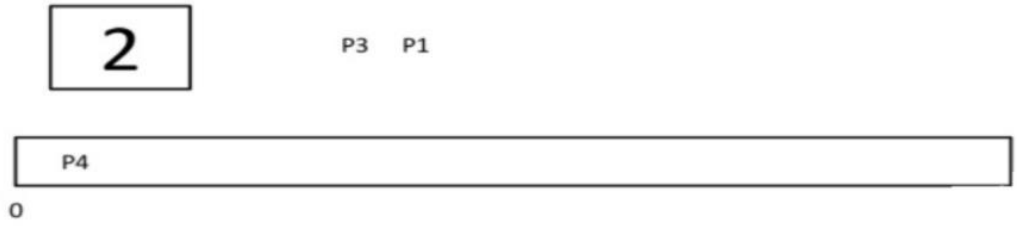
Step 0) At time=0, P4 arrives and starts execution.



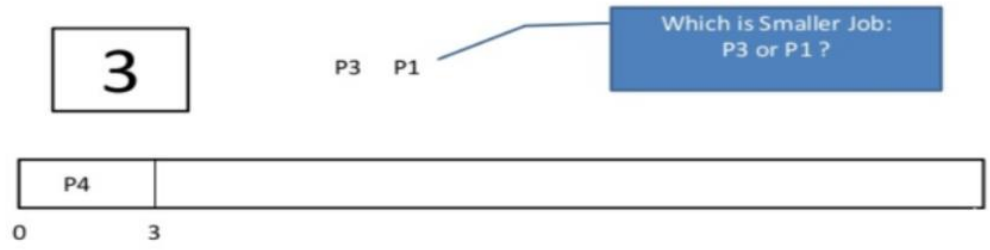
Step 1) At time= 1, Process P3 arrives. But, P4 still needs 2 execution units to complete. It will continue execution.



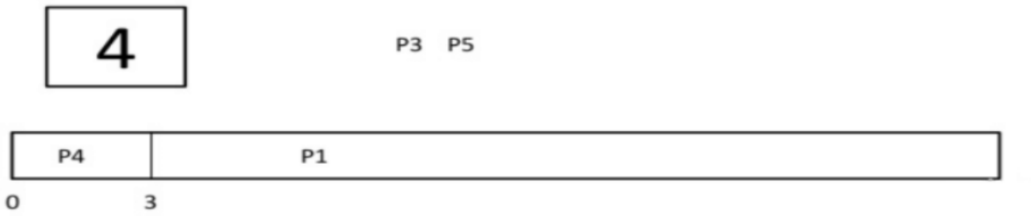
Step 2) At time =2, process P1 arrives and is added to the waiting queue. P4 will continue execution.



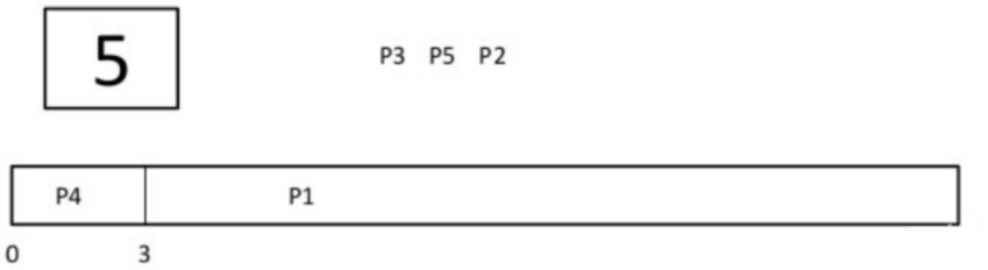
Step 3) At time = 3, process P4 will finish its execution. The burst time of P3 and P1 is compared. Process P1 is executed because its burst time is less compared to P3.



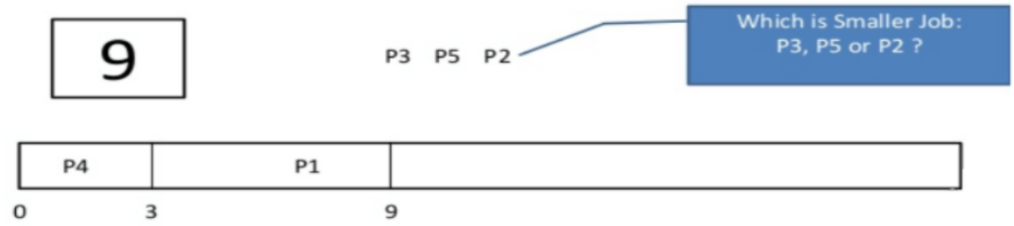
Step 4) At time = 4, process P5 arrives and is added to the waiting queue. P1 will continue execution.



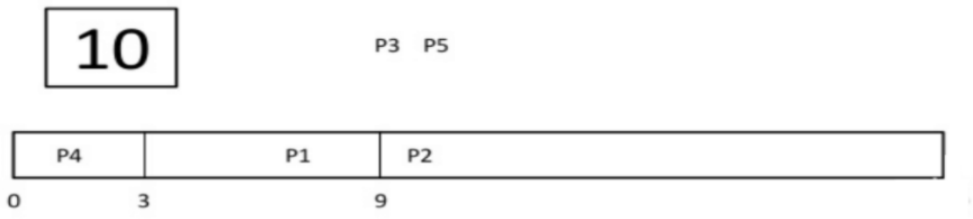
Step 5) At time = 5, process P2 arrives and is added to the waiting queue. P1 will continue execution.



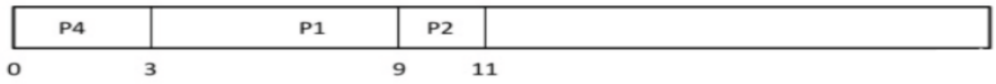
Step 6) At time = 9, process P1 will finish its execution. The burst time of P3, P5, and P2 is compared. Process P2 is executed because its burst time is the lowest.



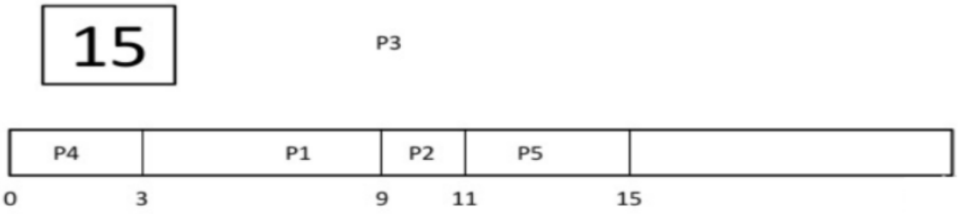
Step 7) At time=10, P2 is executing and P3 and P5 are in the waiting queue.



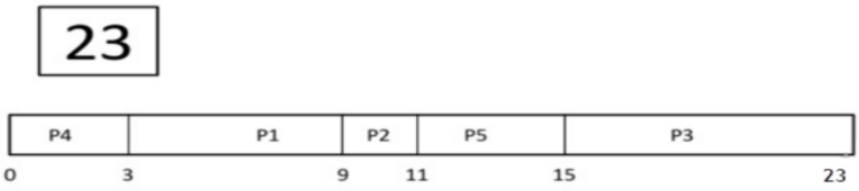
Step 8) At time = 11, process P2 will finish its execution. The burst time of P3 and P5 is compared. Process P5 is executed because its burst time is lower.



Step 9) At time = 15, process P5 will finish its execution.



Step 10) At time = 23, process P3 will finish its execution.



Step 11) Let's calculate the average waiting time for above example.

```
Wait time
P4= 0-0=0
P1=  3-2=1
P2= 9-5=4
P5= 11-4=7
P3= 15-1=14
```

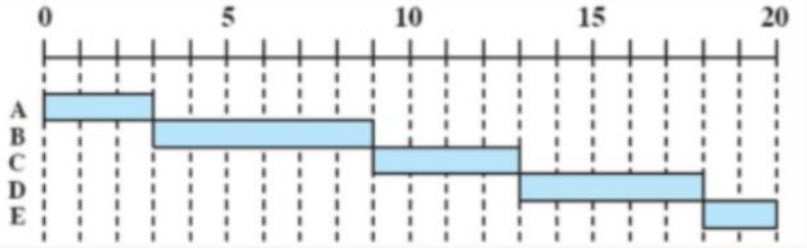
```
Average Waiting Time= 0+1+4+7+14/5 = 26/5 = 5.2
```

14. Write a short note on FCFS scheduling policy.

1. First-come-first-served (FCFS)

- First-come-first-served is nonpreemptive and characterized by the selection function **max[w]**. FCFS is just another way of saying first-in-first-out(FIFO). This means that the process that has spent the most amount of time waiting will be the one to execute.

First-Come-First Served (FCFS)



Process	Arrival Time	Execution Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

- process A starts at time slot zero, since FCFS is non-preemptive, process A will execute until it ceases to execute.
- Process B arrives at time slot 2, but it can't preempt process A, so it must wait for process A to terminate, which terminates at time slot 3.
- Once process A terminates, because it has been waiting longer than any other process, process B is selected for running (there are no other processes in the system).
- The pattern continues, and process B, will also execute until it terminates, or in an additional 6 time steps at time slot 9.

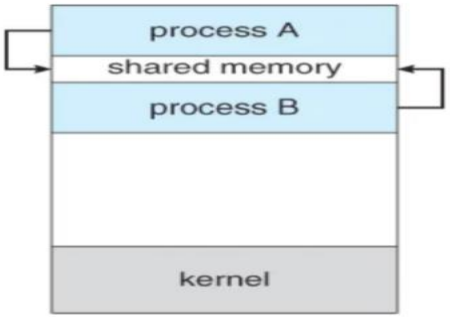
- Processes C, D, and E, all arrive while process B is executing, but at the time process B terminates, C is waiting for 5 time steps (w=5), D for 3 time steps(w=3), and E, for 1 time step(w=1). So applying our selection function max[w], process C will be selected. The pattern continues, until all processes are serviced.

15. Arrival Time and Burst time for three processes P1, P2, P3 are given in the diagram below. Calculate Turn around time, completion time, and waiting time.

Process	Burst Time	Arrival Time
P1	7	0
P2	3	1
P3	4	3

16. Analyze shared memory model of Inter Process Communication

- Communication between processes using shared memory requires processes to share some variable, and it completely depends on how the programmer will implement it.
- One way of communication using shared memory can be imagined like this: Suppose process1 and process2 are executing simultaneously, and they share some resources or use some information from another process.
- Process1 generates information about certain computations or resources being used and keeps it as a record in shared memory. When process2 needs to use the shared information, it will check in the record stored in shared memory and take note of the information generated by process1 and act accordingly.



(b) Shared Memory

- Processes can use shared memory for extracting information as a record from another process as well as for delivering any specific information to other processes.

17. Explain the principle of concurrency

Concurrency is the execution of the multiple instruction sequences at the same time. It happens in the operating system when there are several process threads running in parallel. The running process threads always communicate with each other through shared memory or message passing. Concurrency results in sharing of resources result in problems like deadlocks and resources starvation.

It helps in techniques like coordinating execution of processes, memory allocation and execution scheduling for maximizing throughput.

The relative speed of execution cannot be predicted. It depends on the following:
The activities of other processes
The way operating system handles interrupts
The scheduling policies of the operating system