COURSE NAME : OPERATING SYSTEMS

(COURSE CODE :22516)

COURSE TEACHER : MRS.T.H.GAVHANE

UNIT NO 4 (14Marks)

CPU SCHDULING AND ALGORITHMS

Course Outcome: **Apply scheduling algorithms to calculate turnaround time and average waiting time**

- **UNIT OUTCOME**

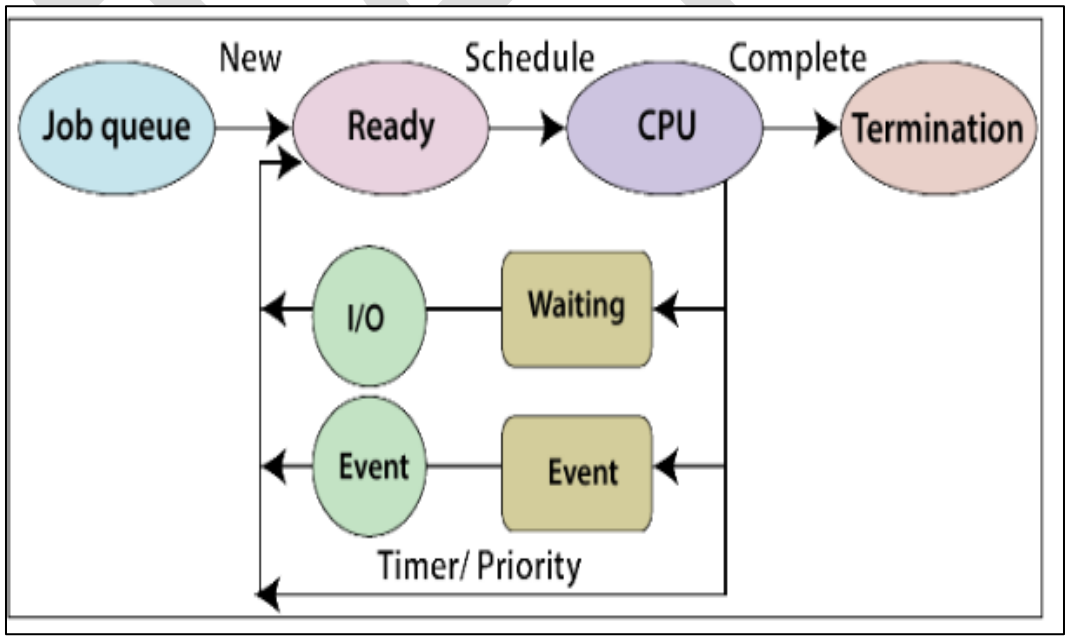
4a	**Justify the need and objective of given job scheduling criteria with relevant example**
4b	**Explain with example the procedure of allocating CPU to the given process using the specified OS**
4c	**Calculate turnaround time and average waiting time of the given scheduling algorithm**
4d	**Explain functioning of the given necessary condition leading to deadlock**

4.1 Scheduling types –Scheduling Objectives, CPU and I/O burst cycles, Pre-emptive, Non-pre-emptive scheduling, scheduling criteria

❖ What is CPU Scheduling :

CPU Scheduling is a process that allows one process to use the CPU while another process is delayed (in standby) due to unavailability of any resources such as I / O etc, thus making full use of the CPU. The purpose of CPU Scheduling is to make the system more efficient, faster, and fairer.

Whenever the CPU becomes idle, the operating system must select one of the processes in the line ready for launch. The selection process is done by a temporary (CPU) scheduler. The Scheduler selects between memory processes ready to launch and assigns the CPU to one of them.



❖ What are the types of CPU Scheduling:

CPU scheduling decisions may take place under the following four circumstances:

1. When a process switches from the **running state** to the **waiting state**(for I/O request or invocation of wait for the termination of one of the child processes).

2. When a process switches from the **running state** to the **ready state** (for example, when an interrupt occurs).

3. When a process switches from the **waiting state** to the **ready state**(for example, completion of I/O).

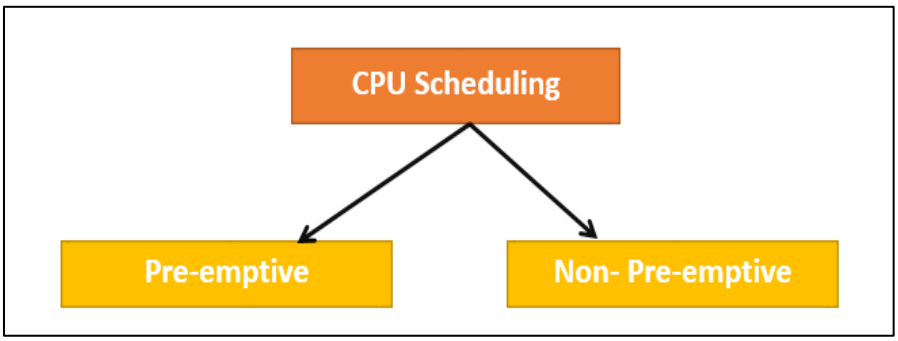
4. When a process switches from **running** to **terminates.**

So in the case of **conditions 1 and 4**, the **CPU does not really have a choice of scheduling**, if a process exists in the ready queue the CPU's response to this would be to select it for execution. In **cases 2 and 3,** the **CPU has a choice of selecting a particular process for executing next.**

When Scheduling takes place only under circumstances 1 and 4, we say the scheduling scheme is non-preemptive; otherwise the scheduling scheme is preemptive.

Types of CPU Scheduling :

There are two types of CPU scheduling ->



a. Pre-emptive Scheduling :

- In this type of Scheduling, the tasks are usually assigned with priorities. At times it is necessary to run a certain task that has a higher priority before another task although it is running. Therefore, the running task is interrupted for some time and resumed later when the priority task has finished its execution.
- Thus this type of scheduling is used mainly when a process switches either from running state to ready state or from waiting state to ready state. The resources (that is CPU cycles) are mainly allocated to the process for a limited amount of time and then are taken away, and after that, the process is again placed back in the ready queue in the case if that process still has a CPU burst time remaining. That process stays in the ready queue until it gets the next chance to execute.
- Some Algorithms that are based on preemptive scheduling are

  Round Robin Scheduling (RR),

  Shortest Remaining Time First (SRTF),

  Priority (preemptive version) Scheduling, etc.

Process	Arrival time	CPU Burst Time (in millisecond
P0	2	3
P1	3	5
P2	0	6
P3	1	5

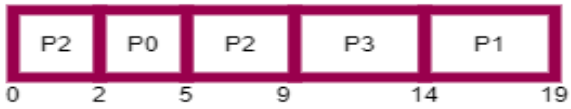


Figure: Preemptive Scheduling

b. Non- Pre-emptive Scheduling :

- Under non-preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.
- It is the only method that can be used on certain hardware platforms because It does not require the special hardware(for example a timer) needed for preemptive scheduling.
- In non-preemptive scheduling, it does not interrupt a process running CPU in the middle of the execution. Instead, it waits till the process completes its CPU burst time, and then after that it can allocate the CPU to any other process.
- Some Algorithms based on non-preemptive scheduling are:

  Shortest Job First (SJF basically non-preemptive) Scheduling

  Priority (non- preemptive version) Scheduling, etc.

Process	Arrival time	CPU Burst Time (in millisecond
P0	2	8
P1	3	6
P2	0	9
P3	1	4

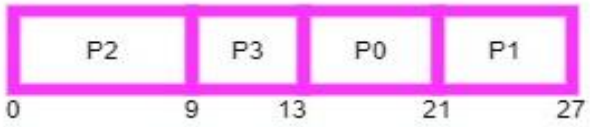


Figure: Non-Preemptive Scheduling

❖ What are the Scheduling Criteria:

Now if the CPU needs to schedule these processes, it must definitely do it wisely. What are the wise decisions it should make to create the "best" scheduling?

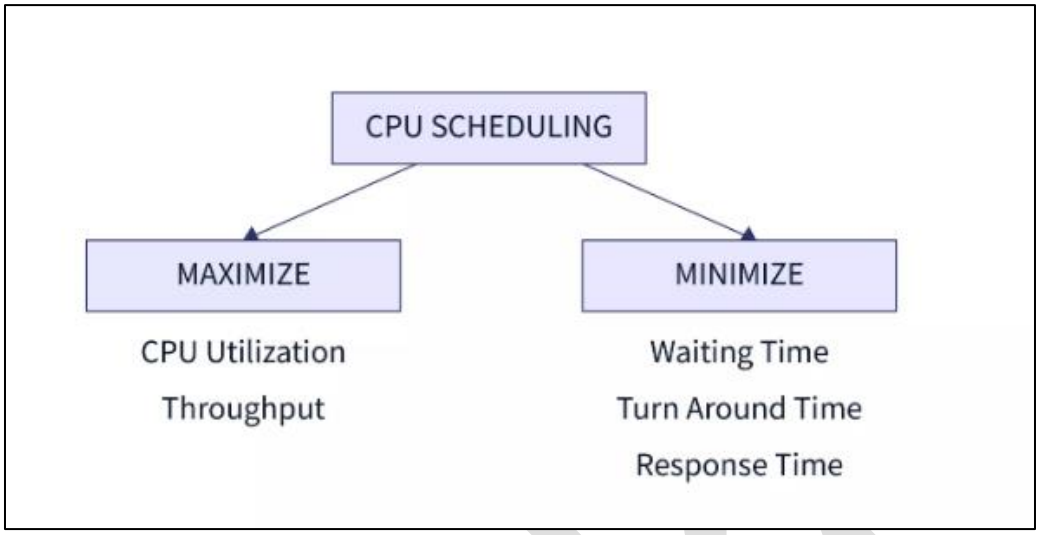
- **CPU Utilization**: It would make sense if the scheduling was done in such a way that the CPU is utilized to its maximum. If a scheduling algorithm is not wasting any CPU cycle or makes the CPU work most of the time (100% of the time, ideally), then the scheduling algorithm can be considered as good.
- **Throughput:** Throughput by definition is the total number of processes that are completed (executed) per unit time or, in simpler terms, it is the total work done by the CPU in a unit of time. Now of course, an algorithm must work to maximize throughput.

- **Turn Around Time:** The turnaround time is essentially the total time that it took for a process to arrive in the ready queue and complete. A good CPU scheduling criteria would be able to minimize this time taken.
- **Waiting Time:** A scheduling algorithm obviously cannot change the time that is required by a process to complete its execution, however, it can minimize the waiting time of the process.
- **Response Time:** If your system is interactive, then taking into consideration simply the turnaround time to judge a scheduling algorithm is not good enough. A process might produce results quicker, and then continue to compute new results while the outputs of previous operations are being shown to the user. Hence, we have another CPU scheduling criteria, which is the response time (time taken from submission of the process until its first 'response' is produced). The goal of the CPU would be to minimize this time.

CPU Scheduling Terminologies :

- **Arrival time:** Arrival time (AT) is the time at which a process arrives at the ready queue.
- **Burst Time:** As you may have seen the third column being 'burst time', it is the time required by the CPU to complete the execution of a process, or the amount of **time required** for the execution of a process. It is also sometimes called the execution time or running time.
- **Completion Time:** As the name suggests, completion time is the time at which a process completes its execution. It is not to be confused with burst time.
- **Turn-Around Time:** Also written as TAT, turnaround time is simply the difference between completion time and arrival time (Completion time - arrival time).

- **Waiting Time:** Waiting time (WT) of a process is the difference between turnaround time and burst time (TAT - BT), i.e. the amount of time a process waits for getting CPU resources in the ready queue.
- **Response Time:** Response time (RT) of a process is the time after which any process gets CPU resources allocated after entering the ready queue.



❖ **Difference Between Preemptive Scheduling and Non-preemptive Scheduling**

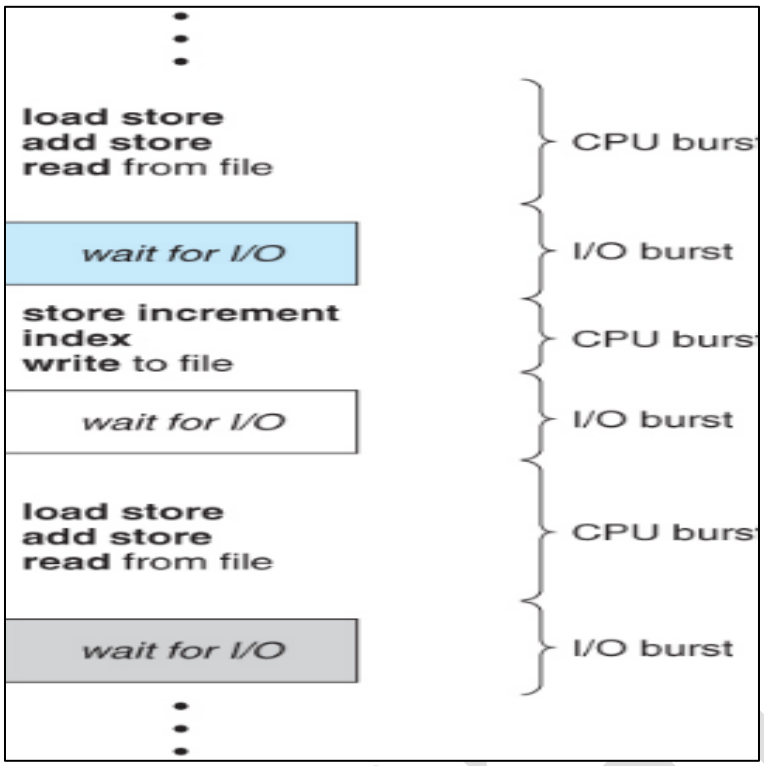
Sr. No.	Preemptive Scheduling	Non-preemptive Scheduling
1	A processor can be preempted to execute the different processes in the middle of any current process execution.	Once the processor starts its execution, it must finish it before executing the other. It can't be paused in the middle.
2	CPU utilization is more efficient compared to Non-Preemptive Scheduling.	CPU utilization is less efficient compared to preemptive Scheduling.
3	Waiting and response time of preemptive Scheduling is less.	Waiting and response time of the non-preemptive Scheduling method is higher.

4	Preemptive Scheduling is prioritized. The highest priority process is a process that is currently utilized.	When any process enters the state of running, the state of that process is never deleted from the scheduler until it finishes its job.
5	Preemptive Scheduling is flexible.	Non-preemptive Scheduling is rigid.
6	Examples: – Shortest Remaining Time First, Round Robin, etc.	Examples: First Come First Serve, Shortest Job First, Priority Scheduling, etc.
7	Preemptive Scheduling algorithm can be pre-empted that is the process can be Scheduled	In non-preemptive scheduling process cannot be Scheduled
8	In this process, the CPU is allocated to the processes for a specific time period.	In this process, CPU is allocated to the process until it terminates or switches to the waiting state.
9	Preemptive algorithm has the overhead of switching the process from the ready state to the running state and vice-versa.	Non-preemptive Scheduling has no such overhead of switching the process from running into the ready state.

❖ CPU and I/O Burst Cycles :

Process execution consists of a cycle of CPU execution and I/O wait. Processes alternate between these two states. Process execution begins with a CPU burst. That is followed by an I/O burst, then another CPU burst, then another I/O burst, and so on. Eventually, the last CPU burst will end with a system request to terminate execution, rather than with another I/O burst.

CPU burst is when the process is being executed in the CPU. I/O burst is when the CPU is waiting for I/O for further execution. After I/O burst, the process goes into the ready queue for the next CPU burst.



CPU Burst

It is the amount of time, a process uses the CPU until it starts waiting for some input or interrupted by some other process

I/O Burst or Input Output burst

It is the amount of time, a process waits for input-output before needing CPU time.

Burst Cycle

The execution of process consists of a cycle of CPU burst and I/O burst. Usually it starts with CPU burst and then followed by I/O burst, another CPU burst, another I/O burst and so on. This cycle will continue until the process is terminated.

CPU bound and I/O bound

In Actual, a Process comprises of CPU bound and I/O bound programs. An I/O bound program has many short CPU bursts. A CPU bound program might have a few long CPU bursts.

4.2 Types of Scheduling Algorithms :

a. **First Come First Serve (FCFS)**
b. **Shortest Job First (SJF)**
c. **Shortest Remaining Time (SRTN)**
d. **Round Robin Scheduling**
e. **Priority Scheduling**
f. **Multilevel Queue Scheduling**

a. First Come First Serve (FCFS) :

In FCFS Scheduling

- The process which arrives first in the ready queue is firstly assigned the CPU.
- In case of a tie, process with smaller process id is executed first.
- It is always non-preemptive in nature.
- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Advantages-

- It is simple and easy to understand.
- It can be easily implemented using queue data structure.
- It does not lead to starvation.

Disadvantages-

- It does not consider the priority or burst time of the processes.
- It suffers from convoy effect i.e. processes with higher burst time arrived before the processes with smaller burst time.

Example:

Let's take an example of The FCFS scheduling algorithm. In the Following schedule, there are 5 processes with process ID **P0, P1, P2, P3 and P4**. P0 arrives at time 0, P1 at time 1, P2 at time 2, P3 arrives at time 3 and Process P4 arrives at time 4 in the ready queue. The processes and their respective Arrival and Burst time are given in the following table.

The Turnaround time and the waiting time are calculated by using the following formula.

1. **Turn Around Time = Completion Time - Arrival Time**
2. **Waiting Time = Turnaround time - Burst Time**

The average waiting Time is determined by summing the respective waiting time of all the processes and divided the sum by the total number of processes.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
0	0	2	2	2	0
1	1	6	8	7	1
2	2	4	12	10	6
3	3	9	21	18	9
4	6	12	33	29	17

Avg Waiting Time=31/5

P0	P1	P2	P3	P4

0 2 8 12 21 33

GANTT chart

b. Shortest Job First Scheduling (SJF):

We were scheduling the processes according to their arrival time (in FCFS scheduling). However, SJF scheduling algorithm, schedules the processes according to their burst time.

In SJF scheduling, the process with the lowest burst time, among the list of available processes in the ready queue, is going to be scheduled next.

However, it is very difficult to predict the burst time needed for a process hence this algorithm is very difficult to implement in the system.

Advantages of SJF

1. Maximum throughput
2. Minimum average waiting and turnaround time

Disadvantages of SJF

1. May suffer with the problem of starvation
2. It is not implementable because the exact Burst time for a process can't be known in advance.

There are different techniques available by which, the CPU burst time of the process can be determined.

Types of Shortest Job First (SJF) Scheduling

There are two types of Shortest Job First Scheduling.

1. Non-Preemptive SJF
2. Preemptive SJF

If the processor knows the Burst time of the processes in advance, the scheduling of the process can be implemented successfully. But practically it's impossible.
When all the processes are available at the same time, then the Shortest Job Scheduling algorithm becomes optimal.

Example :

Preemptive SJF Scheduling: - In this, jobs are moved into the ready queue when they arrive. Those Processes which have less burst time begins its execution first. When the process with less burst time arrives, then the current process stops execution, and the process having less burst time is allocated with the CPU first.

Example of Preemptive SJF Scheduling: In the following example, we have 4 processes with process ID P1, P2, P3, and P4. The arrival time and burst time of the processes are given in the following table.

Process	Burst time	Arrival time	Completion time	Turnaround time	Waiting time
P1	18	0	31	31	13
P2	4	1	5	4	0
P3	7	2	14	12	5
P4	2	3	7	4	2

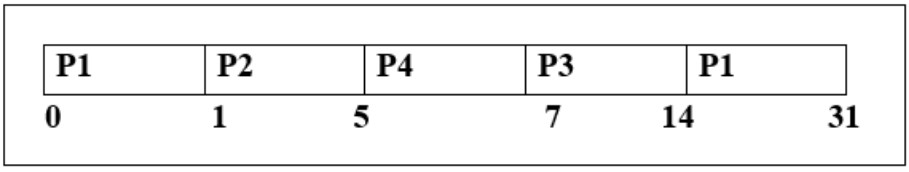
The waiting time and turnaround time are calculated with the help of the following formula.

Waiting Time = Turnaround time – Burst Time
Turnaround Time = Completion time – Arrival time

Process waiting time:
P1=31-18=13
P2=4-4=0
P3=12-7=5
P4=4-2=2
Average waiting time= 13+0+5+2/4
=20
Process Turnaround Time:
P1=31-0=31
P2=5-1=4
P3=14-2=12
P4=7-3=4

Average turnaround time= 31+4+12+4/4
=12.75

The **GANTT chart** of preemptive shortest job first scheduling is:



Non-Preemptive SJF: - In Non-Preemptive Scheduling, if a CPU is located to the process, then the process will hold the CPU until the process enters into the waiting state or terminated.

Example of Non-Preemptive SJF Scheduling:
In the following example, we have 4 processes with process Id P0, P1, P2, and P3. The arrival time and burst time of the processes are given in the following table.

Process ID	Burst Time	Arrival Time	Completion time	Waiting Time	Turnaround Time
P0	8	5	21	8	16
P1	5	0	5	0	5
P2	9	4	16	3	12
P3	2	1	7	4	6

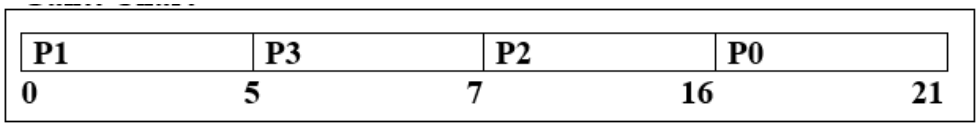
The waiting time and turnaround time are calculated with the help of the following formula.

Waiting Time = Turnaround time – Burst Time
Turnaround Time = Completion time – Arrival time

Process waiting time:
P0= 16-8=8
P1= 5-5=0
P2=12-9=3
P3=6-2=4
Average waiting time= 8+0+3+4/4
=15/4
=3.75
Process turnaround time:
P0=21-5=16
P1=5-0=5

P2=16-4=12
P3=7-1=6
Average turnaround time=16+5+12+6/4
=39/4
=9.75

The **GANTT chart** of Non- preemptive shortest job first scheduling is:



c. Shortest Remaining Time Scheduling (SRTN) :

- The Preemptive version of Shortest Job First(SJF) scheduling is known as Shortest Remaining Time First (SRTF). With the help of the SRTF algorithm, the process having the smallest amount of time remaining until completion is selected first to execute.

- So basically in SRTF, the processes are scheduled according to the shortest remaining time.

- However, the SRTF algorithm involves more overheads than the Shortest job first (SJF)scheduling, because in SRTF OS is required frequently in order to monitor the CPU time of the jobs in the **READY** queue and to perform context switching.

- In the **SRTF scheduling algorithm**, the execution of any process can be stopped after a certain amount of time. On arrival of every process, the short-term scheduler schedules those processes from the list of available processes & running processes that have the least remaining burst time.

- After all the processes are available in the ready queue, then, No preemption will be done and then the algorithm will work the same as SJF scheduling. In the Process Control Block, the context of the process is saved, when the process is removed from the execution and when the next process is scheduled. The PCB is accessed on the next execution of this process.

Advantages of SRTF

The main advantage of the SRTF algorithm is that it makes the processing of the jobs faster than the SJF algorithm, mentioned it's overhead charges are not counted.

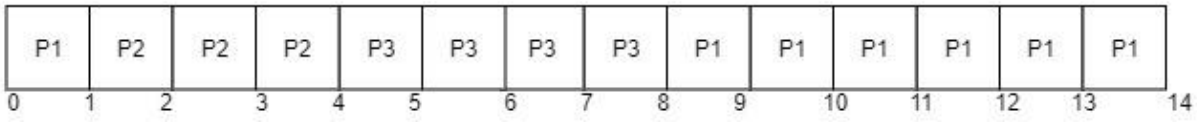
Disadvantages of SRTF

In SRTF, the context switching is done a lot more times than in SJN due to more consumption of the CPU's valuable time for processing. The consumed time of CPU then adds up to its processing time and which then diminishes the advantage of fast processing of this algorithm.

Example

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

The Gantt Chart for SRTF will be:



Explanation

- At the 0th unit of the CPU, there is only one process that is **P1**, so P1 gets executed for the 1 time unit.
- At the 1st unit of the CPU, Process **P2** arrives. Now, the **P1** needs 6 more units more to be executed, and the **P2** needs only 3 units. So, **P2** is executed first by preempting **P1**.
- At the 3rd unit of time, the process **P3** arrives, and the burst time of P3 is 4 units which is more than the completion time of P2 that is 1 unit, so P2 continues its execution.
- Now after the completion of **P2**, the burst time of **P3** is **4 units** that means it needs only 4 units for completion while P1 needs 6 units for completion.
- So, this algorithm picks **P3** above **P1 due to the reason that the completion time of P3 is less than that of P1**
- P3 gets completed at time unit 8, there are no new processes arrived.
- So again, **P1** is sent for execution, and it gets completed at the 14th unit.

As Arrival Time and Burst time for three processes P1, P2, P3 are given in the above diagram. Let us calculate Turnaround time, completion time, and waiting time.

Process	**Arrival Time**	**Burst Time**	**Completion time**	**Turnaround Time** **Turn Around Time = Completion Time – Arrival Time**	**Waiting Time** **Waiting Time = Turn Around Time – Burst Time**
P1	0	7	14	14-0=14	14-7=7
P2	1	3	4	4-1=3	3-3=0
P3	3	4	8	8-3=5	5-4=1

Average waiting time is calculated by adding the waiting time of all processes and then dividing them by no. of processes.

Average waiting time = waiting for time of all processes/ no.of processes

Average waiting time=7+0+1=8/3 = **2.66ms**

d. Round Robin Scheduling :

Round Robin(RR) scheduling algorithm is mainly designed for time-sharing systems. This algorithm is similar to FCFS scheduling, but in Round Robin(RR) scheduling, preemption is added which enables the system to switch between processes.

- A fixed time is allotted to each process, called a **quantum**, for execution.
- Once a process is executed for the given time period that process is preempted and another process executes for the given time period.
- Context switching is used to save states of preempted processes.
- This algorithm is simple and easy to implement and the most important is thing is this algorithm is starvation-free as all processes get a fair share of CPU.
- It is important to note here that the length of time quantum is generally from 10 to 100 milliseconds in length.

Some important characteristics of the Round Robin(RR) Algorithm are as follows:

1. Round Robin Scheduling algorithm resides under the category of Preemptive Algorithms.
2. This algorithm is one of the oldest, easiest, and fairest algorithm.

3. This Algorithm is a real-time algorithm because it responds to the event within a specific time limit.
4. In this algorithm, the time slice should be the minimum that is assigned to a specific task that needs to be processed. Though it may vary for different operating systems.
5. This is a hybrid model and is clock-driven in nature.
6. This is a widely used scheduling method in the traditional operating system.

Advantages :

1. It can be actually implementable in the system because it is not depending on the burst time.
2. It doesn't suffer from the problem of starvation or convoy effect.
3. All the jobs get a fare allocation of CPU.

Disadvantages :

1. The higher the time quantum, the higher the response time in the system.
2. The lower the time quantum, the higher the context switching overhead in the system.
3. Deciding a perfect time quantum is really a very difficult task in the system

Example

Assume there are 5 processes with process ID and burst time given below

PID	Burst Time
P1	6
P2	5
P3	2
P4	3
P5	7

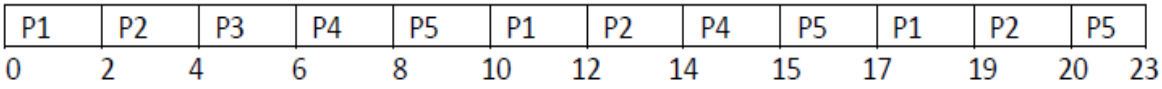
Time quantum: 2
Assume that all process arrives at 0.

Now, we will calculate average waiting time for these processes to complete.

Solution –

We can represent execution of above processes using GANTT chart as shown below –

Gantt Chart:



Round Robin Example Gantt Chart

Explanation:

– First p1 process is picked from the ready queue and executes for 2 per unit time (time slice = 2).
If arrival time is not available, it behaves like FCFS with time slice.

– After P2 is executed for 2 per unit time, P3 is picked up from the ready queue. Since P3 burst time is 2 so it will finish the process execution at once.
– Like P1 & P2 process execution, P4 and p5 will execute 2 time slices and then again it will start from P1 same as above.

Waiting time = Turn Around Time – Burst Time
P1 = 19 – 6 = 13
P2 = 20 – 5 = 15
P3 = 6 – 2 = 4
P4 = 15 – 3 = 12
P5 = 23 – 7 = 16

Average waiting time = (13+15+4+12+16) / 5 = 12

e. Priority Scheduling Algorithm :

Priority Scheduling is a process scheduling algorithm based on priority where the scheduler selects tasks according to priority. Thus, processes with higher priority execute first followed by processes with lower priorities.

If two jobs have the same priorities then the process that should execute first is chosen on the basis of round-robin or FCFS. Which process should have what priority depends on a process' memory requirements, time requirements, the ratio of I/O burst to CPU burst, etc.

Types of Priority Scheduling

Following are the two main types of priority scheduling:

1. **Preemptive Scheduling:** Tasks execute according to their priorities. In case a lower priority task is running and a higher priority task arrives in the

waiting state then the lower priority task is put on hold. The higher priority task replaces it and once done executing the lower priority task resumes execution from when it was paused. This process requires special hardware like a timer.

2. Non-Preemptive Scheduling: The OS allocates the CPU to a specific process that releases the CPU either through context switching or after termination. We can use this method on various hardware platforms because unlike preemptive scheduling it doesn't require any special hardware.

Advantages :

Following are the benefits of priority scheduling method:

- Easy to use.
- Processes with higher priority execute first which saves time.
- The importance of each process is precisely defined.
- A good algorithm for applications with fluctuating time and resource requirements.

Disadvantages :

Following are the disadvantages of priority scheduling:

- We can lose all the low-priority processes if the system crashes.
- This process can cause starvation if high-priority processes take too much CPU time. The lower priority process can also be postponed for an indefinite time.
- There is a chance that a process can't run even when it is ready as some other process is running currently.

Example 1 (Preemptive)

Consider the set of 5 processes whose arrival time and burst time are given below-

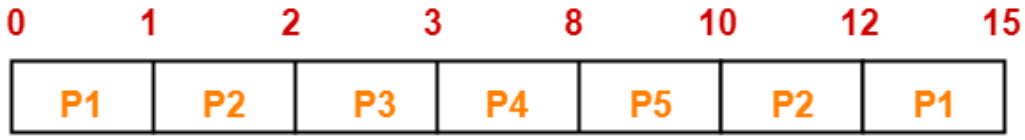
Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5

P5	4	2	5

If the CPU scheduling policy is priority preemptive, calculate the average waiting time and average turnaround time. (Higher number represents higher priority)

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	15	15 – 0 = 15	15 – 4 = 11
P2	12	12 – 1 = 11	11 – 3 = 8
P3	3	3 – 2 = 1	1 – 1 = 0
P4	8	8 – 3 = 5	5 – 5 = 0
P5	10	10 – 4 = 6	6 – 2 = 4

Now,

- Average Turn Around time = (15 + 11 + 1 + 5 + 6) / 5 = 38 / 5 = 7.6 unit
- Average waiting time = (11 + 8 + 0 + 0 + 4) / 5 = 23 / 5 = 4.6 unit

Example 2 (Non-Preemptive)

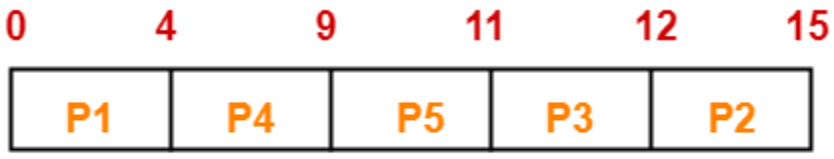
Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

If the CPU scheduling policy is priority non-preemptive, calculate the average waiting time and average turn around time. (Higher number represents higher priority)

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	4	4 – 0 = 4	4 – 4 = 0
P2	15	15 – 1 = 14	14 – 3 = 11
P3	12	12 – 2 = 10	10 – 1 = 9
P4	9	9 – 3 = 6	6 – 5 = 1
P5	11	11 – 4 = 7	7 – 2 = 5

Now,

- Average Turn Around time = (4 + 14 + 10 + 6 + 7) / 5 = 41 / 5 = 8.2 unit
- Average waiting time = (0 + 11 + 9 + 1 + 5) / 5 = 26 / 5 = 5.2 unit

f. Multilevel Queue Scheduling :

A multi-level queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type. Each queue has its own scheduling algorithm.

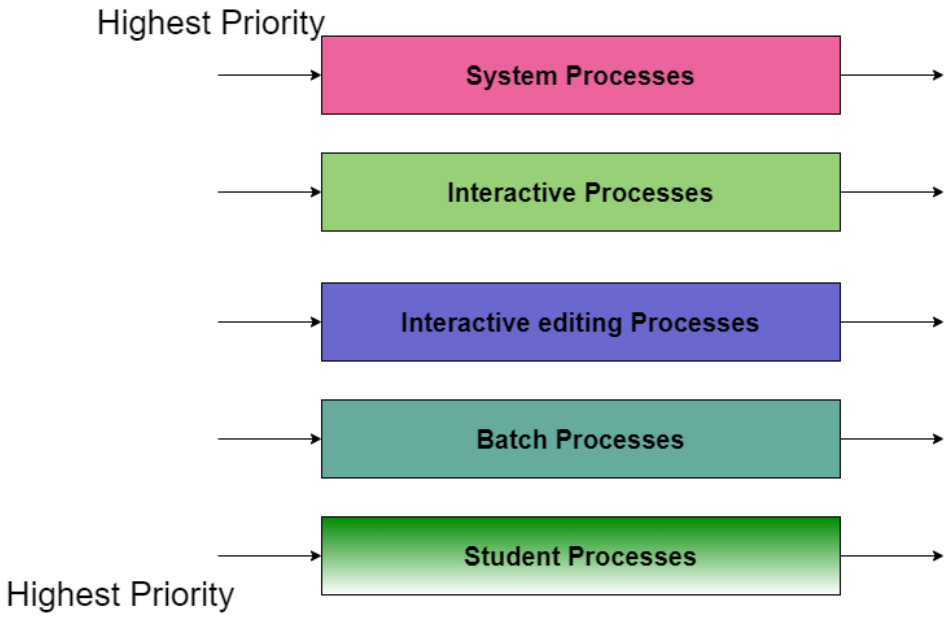
For example, separate queues might be used for foreground and background processes. The foreground queue might be scheduled by the Round Robin algorithm, while the background queue is scheduled by an FCFS algorithm.

In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling. **For example,** The foreground queue may have absolute priority over the background queue.

Let us consider an example of a multilevel queue-scheduling algorithm with five queues:

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

Each queue has absolute priority over lower-priority queues. No process in the batch queue, for example, could run unless the queues for system processes, interactive processes, and interactive editing processes were all empty. If an interactive editing process entered the ready queue while a batch process was running, the batch process will be preempted.



In this case, if there are no processes on the higher priority queue only then the processes on the low priority queues will run. For **Example:** Once processes on the system queue, the Interactive queue, and Interactive editing queue become empty, only then the processes on the batch queue will run.

The Description of the processes in the above diagram is as follows:

- **System Process** The Operating system itself has its own process to run and is termed as System Process.
- **Interactive Process** The Interactive Process is a process in which there should be the same kind of interaction (basically an online game).

- **Batch Processes** Batch processing is basically a technique in the Operating system that collects the programs and data together in the form of the **batch** before the **processing** starts.
- **Student Process** The system process always gets the highest priority while the student processes always get the lowest priority.

In an operating system, there are many processes, in order to obtain the result we cannot put all processes in a queue; thus this process is solved by multilevel queue scheduling.

Advantages

With the help of this scheduling we can apply various kind of scheduling for different kind of processes:

For System Processes: First Come First Serve(FCFS) Scheduling.

For Interactive Processes: Shortest Job First (SJF) Scheduling.

For Batch Processes: Round Robin(RR) Scheduling

For Student Processes: Priority Scheduling

Disadvantages

The main disadvantage of Multilevel Queue Scheduling is the problem of starvation for lower-level processes.

what is starvation?

Starvation:

Due to starvation lower-level processes either never execute or have to wait for a long amount of time because of lower priority or higher priority process taking a large amount of time.

4.3 Deadlock : System Models, necessary conditions leading to deadlocks, Deadlock Handling: Prevention ,avoidance

What is Deadlock ?

All the processes in a system require some resources such as central processing unit(CPU), file storage, input/output devices, etc to execute it. Once the execution is finished, the process releases the resource it was holding. However, when many processes run on a

system they also compete for these resources they require for execution. This may arise a deadlock situation.

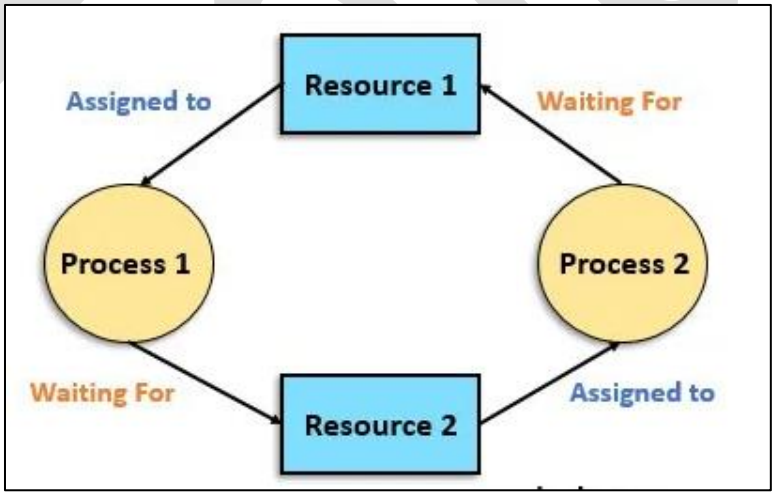
A **deadlock** is a situation in which more than one process is blocked because it is holding a resource and also requires some resource that is acquired by some other process. Therefore, none of the processes gets executed.

Necessary Conditions for Deadlock

The four necessary conditions for a deadlock to arise are as follows.

- **Mutual Exclusion**: Only one process can use a resource at any given time i.e. the resources are non-sharable.
- **Hold and wait**: A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.
- **No preemption**: The resource can be released by a process voluntarily i.e. after execution of the process.
- **Circular Wait**: A set of processes are waiting for each other in a circular fashion. For example, let's say there are a set of processes {P0,P1,P2,P3} such that P0 depends on P1, P1 depends on P2, P2 depends on P3 and P3 depends on P0. This creates a circular relation between all these processes and they have to wait forever to be executed.

Example



Deadlock State

In the above figure, there are two processes and two resources. Process 1 holds "Resource 1" and needs "Resource 2" while Process 2 holds "Resource 2" and requires "Resource 1". This creates a situation of deadlock because none of the two processes can be executed. Since the resources are non-shareable they can only be used by one process at a

time(Mutual Exclusion). Each process is holding a resource and waiting for the other process the release the resource it requires. None of the two processes releases their resources before their execution and this creates a circular wait. Therefore, all four conditions are satisfied.

Methods of Handling Deadlocks in Operating System

The first two methods are used to ensure the system never enters a deadlock.

Deadlock Prevention

This is done by restraining the ways a request can be made. Since deadlock occurs when all the above four conditions are met, we try to prevent any one of them, thus preventing a deadlock.

Deadlock Avoidance

When a process requests a resource, the deadlock avoidance algorithm examines the resource-allocation state. If allocating that resource sends the system into an unsafe state, the request is got granted.

Therefore, it requires additional information such as how many resources of each type is required by a process. If the system enters into an unsafe state, it has to take a step back to avoid deadlock.

Deadlock Detection and Recovery

We let the system fall into a deadlock and if it happens, we detect it using a detection algorithm and try to recover.

Some ways of recovery are as follows.

- Aborting all the deadlocked processes.
- Abort one process at a time until the system recovers from the deadlock.
- Resource Preemption: Resources are taken one by one from a process and assigned to higher priority processes until the deadlock is resolved.

Deadlock Ignorance

In the method, the system assumes that deadlock never occurs. Since the problem of deadlock situation is not frequent, some systems simply ignore it. Operating systems such as UNIX and Windows follow this approach. However, if a deadlock occurs we can reboot our system and the deadlock is resolved automatically.

Difference between Starvation and Deadlocks

Deadlock	Starvation
A deadlock is a situation in which more than one process is blocked because it is holding a resource and also requires some resource that is acquired by some other process.	Starvation is a process in which the low priority processes are postponed indefinitely because the resources are never allocated.
Resources are blocked by a set of processes in a circular fashion.	Resources are continuously used by high-priority resources.
It is prevented by avoiding anyone necessary condition required for a deadlock or recovered using a recovery algorithm.	It can be prevented by aging.
In a deadlock, none of the processes get executed.	In starvation, higher priority processes execute while lower priority processes are postponed.
Deadlock is also called circular wait.	Starvation is also called lived lock.

Advantage of Deadlock Method

- No preemption is needed for deadlocks.
- It is a good method if the state of the resource can be saved and restored easily.
- It is good for activities that perform a single burst of activity.
- It does not need run-time computations because the problem is solved in system design.

Disadvantages of Deadlock Method

- The processes must know the maximum resource of each type required to execute it.
- Preemptions are frequently encountered.
- It delays the process initiation.
- There are inherent pre-emption losses.
- It does not support incremental request of resources.