CPU Scheduling and Algorithm

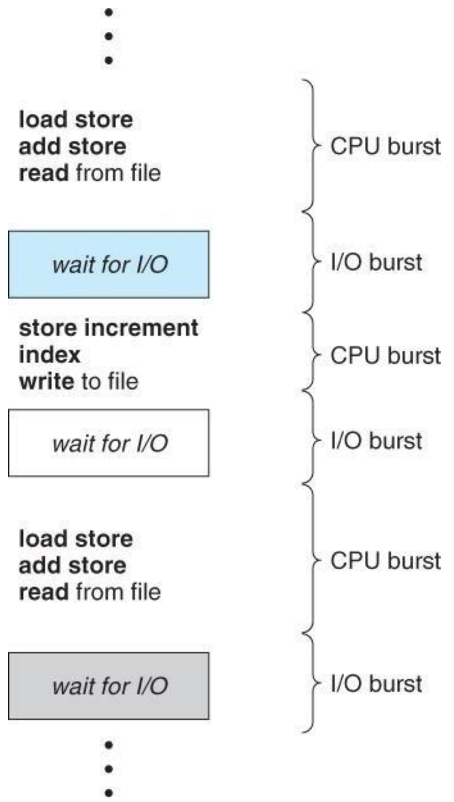
Scheduling types

Scheduling Objectives

- Be Fair while allocating resources to the processes
- Maximize throughput of the system
- Maximize number of users receiving acceptable response times.
- Be predictable
- Balance resource use
- Avoid indefinite postponement
- Enforce Priorities
- Give preference to processes holding key resources
- Give better service to processes that have desirable behaviour patterns

CPU and I/O Burst Cycle:

- 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, followed by an I/O burst, then another CPU burst ... etc
- The last CPU burst will end with a system request to terminate execution rather than with another I/O burst.
- The duration of these CPU burst have been measured.
- An I/O-bound program would typically have many short CPU bursts, A CPU-bound program might have a few very long CPU bursts.
- This can help to select an appropriate CPU-scheduling algorithm.



Preemptive Scheduling:

- Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state.
- The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining.
- That process stays in ready queue till it gets next chance to execute.

Non-Preemptive Scheduling:

- Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state.
- In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state.
- In case of non-preemptive scheduling does not interrupt a process running CPU in middle of the execution.
- Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.

Basis for Comparison	Preemptive Scheduling	Non Preemptive Scheduling
Basic	The resources are allocated to a process for a limited time.	Once resources are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Interrupt	Process can be interrupted in between.	Process can not be interrupted till it terminates or switches to waiting state.
Starvation	If a high priority process frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then another process with less CPU burst time may starve.
Overhead	Preemptive scheduling has overheads of scheduling the processes.	Non-preemptive scheduling does not have overheads.
Flexibility	Preemptive scheduling is flexible.	Non-preemptive scheduling is rigid.
Cost	Preemptive scheduling is cost associated.	Non-preemptive scheduling is not cost associative.

Scheduling Criteria

- There are several different criteria to consider when trying to select the "best" scheduling algorithm for a particular situation and environment, including:
    - **CPU utilization -** Ideally the CPU would be busy 100% of the time, so as to waste 0 CPU cycles. On a real system CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded.)
    - **Throughput -** Number of processes completed per unit time. May range from 10 / second to 1 / hour depending on the specific processes.

- **Turnaround time -** Time required for a particular process to complete, from submission time to completion.
- **Waiting time -** How much time processes spend in the ready queue waiting their turn to get on the CPU.
- **Response time -** The time taken in an interactive program from the issuance of a command to the ***commence*** of a response to that command.

In brief:
Arrival Time: Time at which the process arrives in the ready queue. **Completion Time:** Time at which process completes its execution. **Burst Time:** Time required by a process for CPU execution. **Turn Around Time:** Time Difference between completion time and arrival time. Turn Around Time = Completion Time – Arrival Time
Waiting Time(W.T): Time Difference between turnaround time and burst time. Waiting Time = Turn Around Time – Burst Time

4.2 Types of Scheduling Algorithm

(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 1:

Q. Consider the following processes with burst time (CPU Execution time). Calculate the average waiting time and average turnaround time?

Process id	Arrival time	Burst time/CPU execution time
P1	0	2
P2	1	3
P3	2	5
P4	3	4
P5	4	6

Sol.

Gantt chart

P1	P1	P2	P2	P2	P3	P3	P3	P3	P3	P4	P4	P4	P4	P5	P5	P5	P5	P5	P5

Time: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Completion time of P1 (2); Completion time of P2 (5); Completion time of P3 (10); Completion time of P4 (14); Completion time of P5 (20)

Turnaround time= Completion time – Arrival time

Waiting time= Turnaround time – Burst time

Process id	Arrival time	Burst time	Completion time	Turnaround time	Waiting time
P1	0	2	2	2-0=2	2-2=0
P2	1	3	5	5-1=4	4-3=1
P3	2	5	10	10-2=8	8-5=3
P4	3	4	14	14-3=11	11-4=7
P5	4	6	20	20-4=16	16-6=10

Average turnaround time= $\sum_{i=0}^{n} \text{Turnaround time(i)/n}$ where, n= no. of process

Average waiting time= $\sum_{i=0}^{n} \text{Wating time(i)/n}$ where, n= no. of process

Average turnaround time= 2+4+8+11+16/5 =41/5 =8.2

Average waiting time= 0+1+3+7+10/5 = 21/5 =4.2

Example 2:

Consider the processes P1, P2, P3 given in the below table, arrives for execution in the same order, with Arrival Time 0, and given Burst Time,

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	24
P2	0	3
P3	0	3

Gantt chart

P_1	P_2	P_3

0 24 27 30

PROCESS	WAIT TIME	TURN AROUND TIME
P1	0	24
P2	24	27
P3	27	30

Total Wait Time = 0 + 24 + 27 = 51 ms

Average Waiting Time = (Total Wait Time) / (Total number of processes) = 51/3 = 17 ms

Total Turn Around Time: 24 + 27 + 30 = 81 ms

Average Turn Around time = (Total Turn Around Time) / (Total number of processes) = 81 / 3 = 27 ms

Throughput = 3 jobs/30 sec = 0.1 jobs/sec

Example 3:

Consider the processes P1, P2, P3, P4 given in the below table, arrives for execution in the same order, with given Arrival Time and Burst Time.

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Gantt chart

P_1	P_2	P_3	P_4

0 8 12 21 26

PROCESS	WAIT TIME	TURN AROUND TIME
P1	0	8 – 0 = 8
P2	8 – 1 = 7	12 – 1 = 11
P3	12 – 2 = 10	21 – 2 = 19
P4	21 – 3 = 18	26 – 3 = 23

Total Wait Time:= 0 + 7 + 10 + 18 = 35 ms

Average Waiting Time = (Total Wait Time) / (Total number of processes)= 35/4 = 8.75 ms

Total Turn Around Time: 8 + 11 + 19 + 23 = 61 ms

Average Turn Around time = (Total Turn Around Time) / (Total number of processes) 61/4 = 15.25 ms

Throughput: 4 jobs/26 sec = 0.15385 jobs/sec

(b) Shortest Job First (SJF)

- Process which have the shortest burst time are scheduled first.
- If two processes have the same bust time, then FCFS is used to break the tie.
- This is a non-pre-emptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.
- Pre-emptive mode of Shortest Job First is called as Shortest Remaining Time First (SRTF).

Advantages-

- SRTF is optimal and guarantees the minimum average waiting time.
- It provides a standard for other algorithms since no other algorithm performs better than it.

Disadvantages-

- It can not be implemented practically since burst time of the processes can not be known in advance.
- It leads to starvation for processes with larger burst time.
- Priorities can not be set for the processes.
- Processes with larger burst time have poor response time.

Example-01:

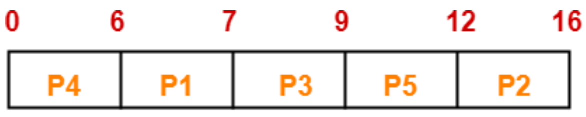
Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

Solution-

If the CPU scheduling policy is SJF non-preemptive, calculate the average waiting time and average turnaround time.

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	7	7 – 3 = 4	4 – 1 = 3
P2	16	16 – 1 = 15	15 – 4 = 11
P3	9	9 – 4 = 5	5 – 2 = 3
P4	6	6 – 0 = 6	6 – 6 = 0
P5	12	12 – 2 = 10	10 – 3 = 7

Now,

- Average Turn Around time = (4 + 15 + 5 + 6 + 10) / 5 = 40 / 5 = 8 unit
- Average waiting time = (3 + 11 + 3 + 0 + 7) / 5 = 24 / 5 = 4.8 unit

Example-02:

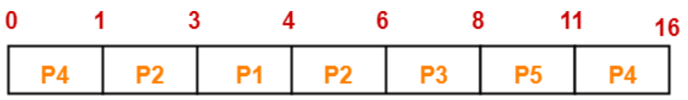
Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

If the CPU scheduling policy is SJF pre-emptive, calculate the average waiting time and average turnaround time.

Solution-

Gantt Chart-



Gantt Chart

Process Id	Exit time	Turn Around time	Waiting time
P1	4	4 – 3 = 1	1 – 1 = 0
P2	6	6 – 1 = 5	5 – 4 = 1
P3	8	8 – 4 = 4	4 – 2 = 2
P4	16	16 – 0 = 16	16 – 6 = 10
P5	11	11 – 2 = 9	9 – 3 = 6

Now,

- Average Turn Around time = (1 + 5 + 4 + 16 + 9) / 5 = 35 / 5 = 7 unit
- Average waiting time = (0 + 1 + 2 + 10 + 6) / 5 = 19 / 5 = 3.8 unit

<u>Example-03:</u>

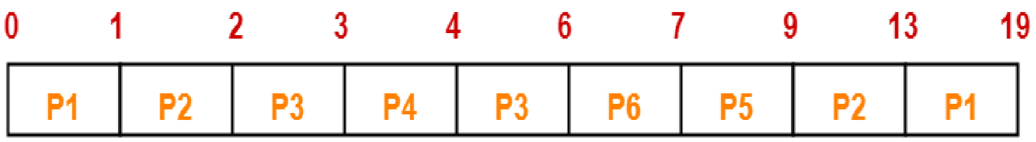
Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	7
P2	1	5
P3	2	3
P4	3	1
P5	4	2
P6	5	1

If the CPU scheduling policy is shortest remaining time first, calculate the average waiting time and average turnaround time.

<u>Solution-</u>

<u>Gantt Chart-</u>



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	19	19 – 0 = 19	19 – 7 = 12
P2	13	13 – 1 = 12	12 – 5 = 7
P3	6	6 – 2 = 4	4 – 3 = 1
P4	4	4 – 3 = 1	1 – 1 = 0
P5	9	9 – 4 = 5	5 – 2 = 3
P6	7	7 – 5 = 2	2 – 1 = 1

Now,

- Average Turn Around time = (19 + 12 + 4 + 1 + 5 + 2) / 6 = 43 / 6 = 7.17 unit
- Average waiting time = (12 + 7 + 1 + 0 + 3 + 1) / 6 = 24 / 6 = 4 unit

Example -04:

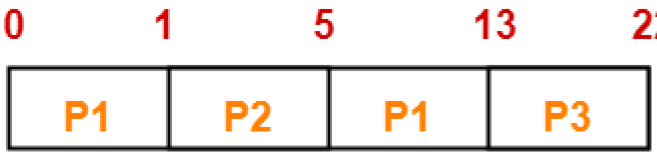
Consider the set of 3 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	9
P2	1	4
P3	2	9

If the CPU scheduling policy is SRTF, calculate the average waiting time and average turn around time.

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	13	13 – 0 = 13	13 – 9 = 4
P2	5	5 – 1 = 4	4 – 4 = 0
P3	22	22- 2 = 20	20 – 9 = 11

Now,

- Average Turn Around time = (13 + 4 + 20) / 3 = 37 / 3 = 12.33 unit
- Average waiting time = (4 + 0 + 11) / 3 = 15 / 3 = 5 unit

Example-05:

Consider the set of 4 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	20
P2	15	25
P3	30	10
P4	45	15

If the CPU scheduling policy is SRTF, calculate the waiting time of process P2.

Solution-

Gantt Chart-

Gantt Chart

Now, we know-

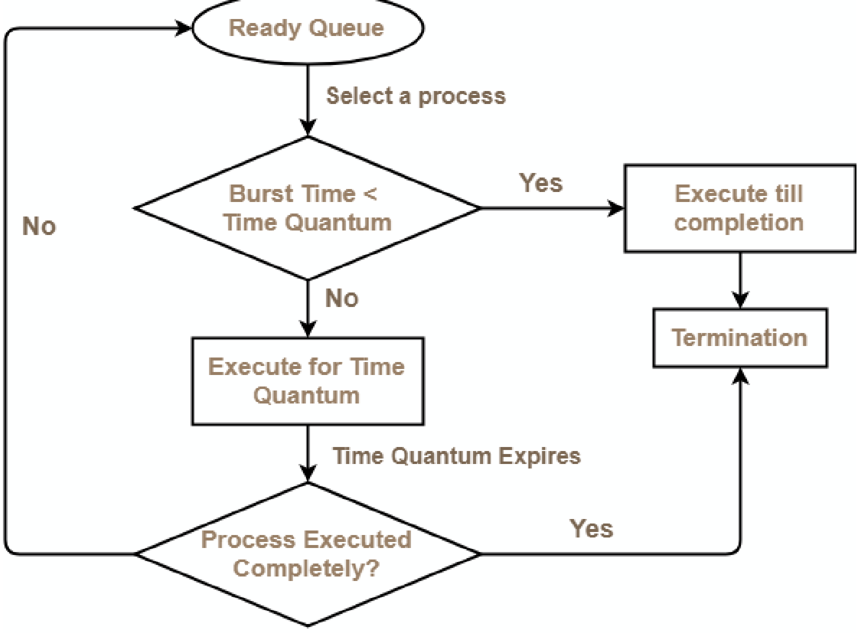
- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Thus,

- Turn Around Time of process P2 = 55 – 15 = 40 unit
- Waiting time of process P2 = 40 – 25 = 15 unit

(c) Round Robin Scheduling

- CPU is assigned to the process on the basis of FCFS for a fixed amount of time.
- This fixed amount of time is called as time quantum or time slice.
- After the time quantum expires, the running process is preempted and sent to the ready queue.
- Then, the processor is assigned to the next arrived process.
- It is always preemptive in nature.



Round Robin Scheduling

Advantages-

- It gives the best performance in terms of average response time.
- It is best suited for time sharing system, client server architecture and interactive system.

Disadvantages-

- It leads to starvation for processes with larger burst time as they have to repeat the cycle many times.
- Its performance heavily depends on time quantum.
- Priorities can not be set for the processes.

With decreasing value of time quantum,

- Number of context switch increases
- Response time decreases
- Chances of starvation decreases

Thus, smaller value of time quantum is better in terms of response time.

With increasing value of time quantum,

- Number of context switch decreases
- Response time increases
- Chances of starvation increases

Thus, higher value of time quantum is better in terms of number of context switch.

- With increasing value of time quantum, Round Robin Scheduling tends to become FCFS Scheduling.
- When time quantum tends to infinity, Round Robin Scheduling becomes FCFS Scheduling.
- The performance of Round Robin scheduling heavily depends on the value of time quantum.
- The value of time quantum should be such that it is neither too big nor too small.

Example-01:

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

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

If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turnaround time.

Solution-

Ready Queue- P5, P1, P2, P5, P4, P1, P3, P2, P1

Gantt Chart-

0 2 4 5 7 9 11 12 13 14

P1	P2	P3	P1	P4	P5	P2	P1	P5

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	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	10 – 3 = 7

Now,

- Average Turn Around time = (13 + 11 + 3 + 6 + 10) / 5 = 43 / 5 = 8.6 unit
- Average waiting time = (8 + 8 + 2 + 4 + 7) / 5 = 29 / 5 = 5.8 unit

Problem-02:

Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	4
P2	1	5
P3	2	2
P4	3	1
P5	4	6
P6	6	3

If the CPU scheduling policy is Round Robin with time quantum = 2, calculate the *average waiting time* and *average turnaround time.*

Solution-

Ready Queue- P5, P6, P2, P5, P6, P2, P5, P4, P1, P3, P2, P1

Gantt chart-

0 2 4 6 8 9 11 13 15 17 18 19 21

P1	P2	P3	P1	P4	P5	P2	P6	P5	P2	P6	P5

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	8	8 – 0 = 8	8 – 4 = 4
P2	18	18 – 1 = 17	17 – 5 = 12
P3	6	6 – 2 = 4	4 – 2 = 2
P4	9	9 – 3 = 6	6 – 1 = 5
P5	21	21 – 4 = 17	17 – 6 = 11
P6	19	19 – 6 = 13	13 – 3 = 10

Now,

- Average Turn Around time = (8 + 17 + 4 + 6 + 17 + 13) / 6 = 65 / 6 = 10.84 unit
- Average waiting time = (4 + 12 + 2 + 5 + 11 + 10) / 6 = 44 / 6 = 7.33 unit

Problem-03: Consider the set of 6 processes whose arrival time and burst time are given below-

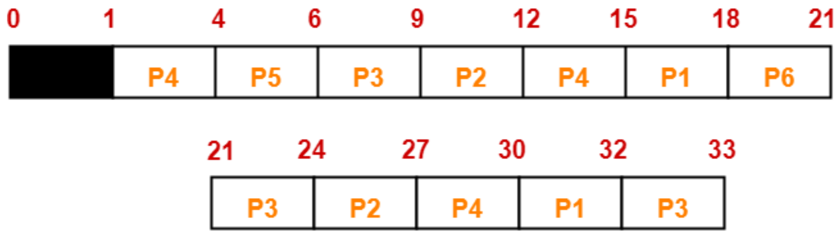
Process Id	Arrival time	Burst time
P1	5	5
P2	4	6
P3	3	7
P4	1	9
P5	2	2
P6	6	3

If the CPU scheduling policy is Round Robin with time quantum = 3, calculate the average waiting time and average turnaround time.

Solution-

Ready Queue- P3, P1, P4, P2, P3, P6, P1, P4, P2, P3, P5, P4

Gantt chart-



Now, we know-

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

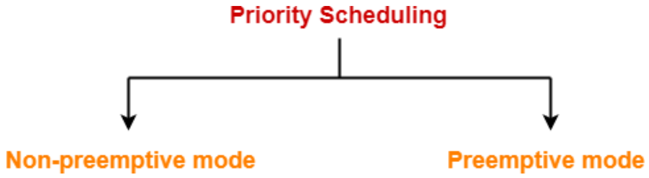
Process Id	Exit time	Turn Around time	Waiting time
P1	32	32 – 5 = 27	27 – 5 = 22
P2	27	27 – 4 = 23	23 – 6 = 17
P3	33	33 – 3 = 30	30 – 7 = 23
P4	30	30 – 1 = 29	29 – 9 = 20
P5	6	6 – 2 = 4	4 – 2 = 2
P6	21	21 – 6 = 15	15 – 3 = 12

Now,

- Average Turn Around time = (27 + 23 + 30 + 29 + 4 + 15) / 6 = 128 / 6 = 21.33 unit
- Average waiting time = (22 + 17 + 23 + 20 + 2 + 12) / 6 = 96 / 6 = 16 unit

(d) Priority Scheduling

- Out of all the available processes, CPU is assigned to the process having the highest priority.
- In case of a tie, it is broken by **FCFS Scheduling**.
- Priority Scheduling can be used in both preemptive and non-preemptive mode.



- The waiting time for the process having the highest priority will always be zero in preemptive mode.
- The waiting time for the process having the highest priority may not be zero in non-preemptive mode.

Priority scheduling in preemptive and non-preemptive mode behaves exactly same under following conditions-

- The arrival time of all the processes is same
- All the processes become available

Advantages-

- It considers the priority of the processes and allows the important processes to run first.
- Priority scheduling in pre-emptive mode is best suited for real time operating system.

Disadvantages-

- Processes with lesser priority may starve for CPU.
- There is no idea of response time and waiting time.

Problem-01:

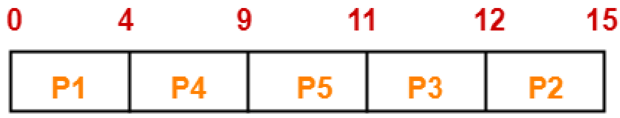
Consider the set of 5 processes whose arrival time and burst time are given below-

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 turnaround time. (*Higher number represents higher priority*)

Solution-

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

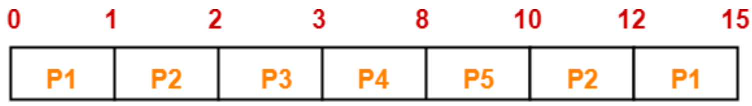
Problem-02: Consider the set of 5 processes whose arrival time and burst time are given below-

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 turn around time. (Higher number represents higher priority).

Solution-

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

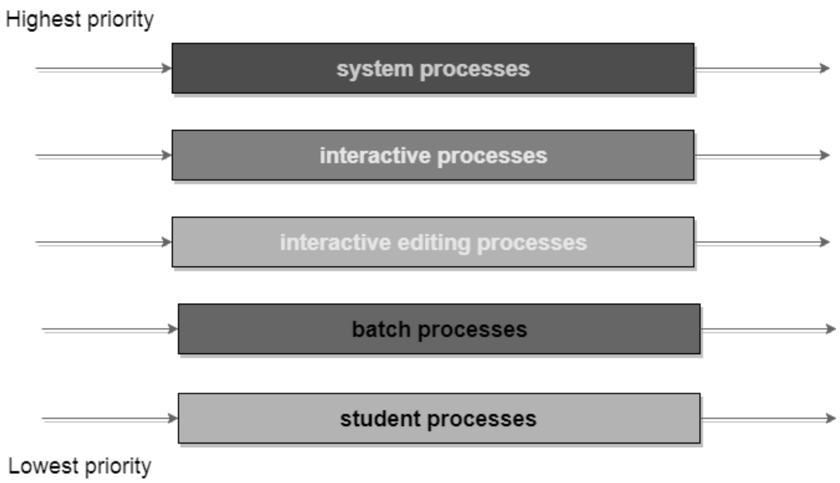
(d) Multilevel Queue Scheduling

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.

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

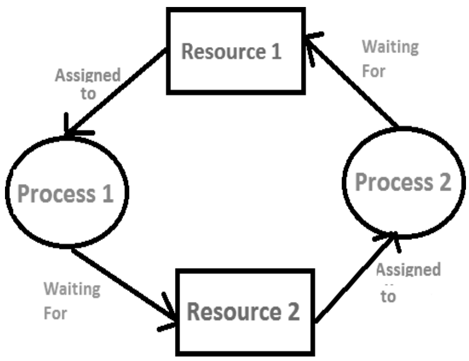
1. System Processes
2. Interactive Processes
3. Interactive Editing Processes
4. Batch Processes
5. Student Processes

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 pre-empted.



4.3 Deadlock

- **Deadlock** is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.
- For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.



Deadlock can arise if following four necessary conditions hold simultaneously.

1. **Mutual Exclusion:** One or more than one resource are non-sharable means Only one process can use at a time.
2. **Hold and Wait:** A process is holding at least one resource and waiting for another resources.
3. **No Pre-emption:** A resource cannot be taken from a process unless the process releases the resource means the process which once scheduled will be executed till the completion and no other process can be scheduled by the scheduler meanwhile.
4. **Circular Wait:** A set of processes are waiting for each other in circular form means All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

Difference between Starvation and Deadlock

Sr.	Deadlock	Starvation
1	Deadlock is a situation where no process got blocked and no process proceeds	Starvation is a situation where the low priority process got blocked and the high priority processes proceed.
2	Deadlock is an infinite waiting.	Starvation is a long waiting but not infinite.
3	Every Deadlock is always a starvation.	Every starvation need not be deadlock.
4	The requested resource is blocked by the other process.	The requested resource is continuously be used by the higher priority processes.
5	Deadlock happens when Mutual exclusion, hold and wait, No preemption and circular wait occurs simultaneously.	It occurs due to the uncontrolled priority and resource management.

Deadlock Handling

The various strategies for handling deadlock are-

1. Deadlock Prevention
2. Deadlock Avoidance
3. *Deadlock Detection and Recovery*
4. *Deadlock Ignorance*

1. Deadlock Prevention

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

Mutual Exclusion

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

Hold and Wait

- To prevent this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.

No Preemption

- Preemption of process resource allocations can prevent this condition of deadlocks, when it is possible.

Circular Wait

- One way to avoid circular wait is to number all resources, and to require that processes request resources only in strictly increasing (or decreasing) order.

2. Deadlock Avoidance

- In deadlock avoidance, the operating system checks whether the system is in safe state or in unsafe state at every step which the operating system performs.
- The process continues until the system is in safe state.
- Once the system moves to unsafe state, the OS has to backtrack one step.
- In simple words, The OS reviews each allocation so that the allocation doesn't cause the deadlock in the system.

3. Deadlock detection and recovery

- This strategy involves waiting until a deadlock occurs.
- After deadlock occurs, the system state is recovered.
- The main challenge with this approach is detecting the deadlock.

4. Deadlock Ignorance

- This strategy involves ignoring the concept of deadlock and assuming as if it does not exist.
- This strategy helps to avoid the extra overhead of handling deadlock.
- Windows and Linux use this strategy and it is the most widely used method.