OSY - Notes


V2V EDTECH LLP
Online Coaching at an Affordable Price.
OUR SERVICES:
• Diploma in All Branches, All Subjects
• Degree in All Branches, All Subjects
• BSCIT / CS
• Professional Courses
+91 93260 50669
V2V EdTech LLP
v2vedtech.com
v2vedtech

Unit - III CPU Scheduling

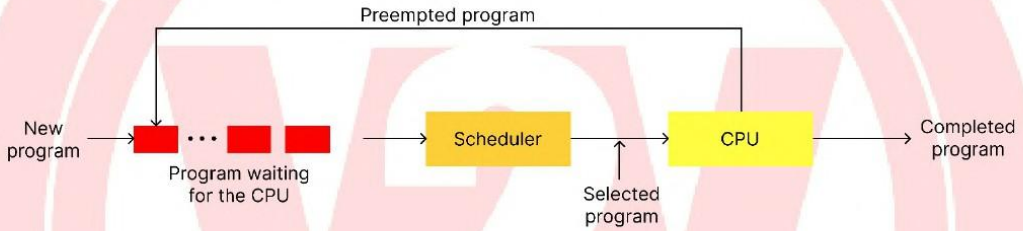
➢ 3.1 Scheduling

• Basic concept

Scheduling is a key function of an operating system. It involves the process manager handling the removal of a running process from the CPU and selecting another process to be executed based on a specific strategy.

The scheduler is the kernel component (module/program) responsible for deciding which program should be executed on the CPU. It selects a program to be executed next for execution.

When the CPU becomes free, the scheduler chooses another program from the set of programs waiting to run on the CPU. The decision of which program gets the CPU and for how long is called scheduling.



A Schematic of Scheduling

• Objectives of scheduling:

Depending on the system, the user and designer might expect the scheduler to have the following objectives for scheduling:

1) **Fairness:**
   Fairness is defined as, the degree to which each process gets an equal chance to execute
   A scheduler makes sure that each process gets its fair chance its fair share of the CPU time.

2) **Maximize Resource Utilization:**
   The scheduling techniques should keep the resources of the system busy.

3) **Avoid Indefinite Postponement:**
   A process should not experience an unbounded wait time before or while receiving service.

4) **Response Time:**
   A scheduler should minimize the response time for interactive users.

5. **Turnaround:**
   A scheduler should minimize the time so batch users must wait for an output time.

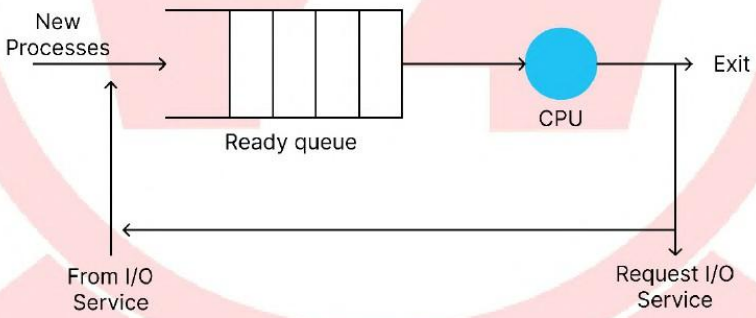
6. **Maximize Throughput:**
   A scheduling disc should maximize the number of jobs processed per unit time.

7. **Enforce Priorities:**
   If the system assign priorities to processes, the scheduling mechanism should favour the higher-priority processes.

- **Scheduling Model:**

CPU scheduling is the process of selecting a process from the ready queue and assigning the CPU to it. CPU scheduling algorithms decides which of the processes in the ready queue is to be allocated to the CPU. The CPU scheduler is the part of the operating system that selects the next process to which the CPU will be allocated de-allocates the CPU from the process currently executing, and allocates the CPU to the newly selected process. The algorithm used by the scheduler to carry out the selection of a process for execution is known as scheduling algorithm.

Every process in OS that requests a CPU service carries out following sequence of actions.



In first action, join the ready queue and wait for CPU service. In second action, execute (receive CPU service) for the duration of the current CPU burst or for the duration of the time slice (time out). In third action, join the I/O queue to wait for I/O service or return to the ready queue to wait for more CPU service. In fourth action, terminate and exit if service is completed i.e. if there are no more CPU or I/O burst. If more service is required, return to the ready queue to wait for more CPU service.

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

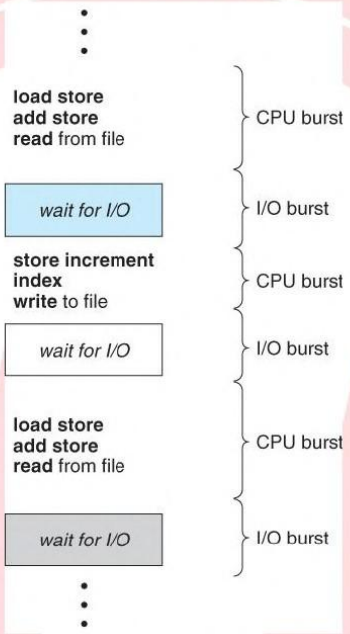
Insta : v2vedtech | App Link | v2vedtech.com

- **CPU and I/O burst cycle**

CPU burst cycle:
It is a time period when process is busy with CPU.

I/O burst cycle:
It is a time period when process is busy in working with I/O resources.

Diagram for I/O Burst and CPU Burst cycle



Execution is an Alternating sequence of CPU and I/O Burst

I/O Burst and CPU Burst Cycle:

A process execution consists of a cycle of CPU execution and I/O wait. A process starts its execution when CPU is assigned to it, so process execution begins with a CPU burst cycle. This is followed by an I/O burst cycle when a process is busy doing I/O operations. A process can frequently switch from CPU burst cycle to I/O burst cycle and vice versa. The complete execution of a process starts with CPU burst cycle, followed by I/O burst cycle, then followed by another CPU burst cycle, then followed by another I/O burst cycle and so on. The final CPU burst cycle ends with a system request to terminate execution.

➢ 3.2 Preemptive and Non - preemptive scheduling, scheduling criteria

- **Preemptive scheduling:**

In preemptive scheduling if a higher priority process is entering the system, the currently running process is stopped and the CPU transfers the control to the higher priority process. The currently running process may be interrupted and moved to the ready state by the operating system. The preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

Circumstances of pre-emptive:

- Process switch from running to ready state
- Process switch from waiting to ready state

Example:

- SJF (Shortest Job First)
- Priority RR (Round Robin)

- **Non-pre-emptive Scheduling:**

Once the CPU has been allocated to a process, the process keeps the CPU until it releases CPU either by terminating or by switching to waiting state. Non-pre-emptive algorithms are designed so that once a process enters the running state, it cannot be pre-empted until it completes its allotted time.

Circumstances of non pre-emptive:

- When process switches from running to waiting state
- When process terminates.

Example:

- FCFS (First come first serve)

State difference between preemptive scheduling and non-preemptive scheduling (S – 22, S - 23)

Preemptive scheduling	Non-preemptive scheduling
Even if CPU is allocated to one process, CPU can be preempted to other process if other process is having higher priority.	once, the CPU has been allocated to a process the process keeps the CPU until releases either by terminating or by switching to waiting state.
Preemptive scheduling is more complex.	Non-preemptive scheduling is simple.
Throughput is less	Throughput is high.

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

Insta : v2vedtech | App Link | v2vedtech.com

It is suitable for real time system	It is not suitable for real time system
Algorithm design is Complex	Algorithm design is simple
The system resources are used efficiently	The system resources are not used efficiently
Only the processes having higher priority are scheduled.	Processes having any priority can get scheduled.
It does not treat all processes as equal.	It treats all process as equal.
It has high overhead.	It has low overhead.
Circumstances for Pre-emptive:	**Circumstances for Non-preemptive:**
• process switch from running to ready state	• process switches from running to waiting state
• process switch from waiting to ready state	• process terminates
Example: Round Robin Example: Priority algorithms.	**Example:** FCFS algorithm.

Explain any four scheduling criteria. (W – 19, S – 22, W – 22, S - 23)

- **Scheduling criteria**

  - ✓ CPU Utilization
  - ✓ Throughput
  - ✓ Turnaround Time (TAT)
  - ✓ Waiting Time
  - ✓ Balanced Utilization
  - ✓ Response Time

1) CPU Utilization:

CPU utilization is defined as, the percentage of time the CPU is busy in executing processes. For higher utilization, CPU must be kept as busy as possible, that is, there must be some process running at all times. CPU utilization may range from 0 to 100 percent. In real system it should range from 40 percent to 90 percent.

2) Throughput:

Throughput is defined as, the total number of processes that a system can execute per unit of time. If the CPU is busy executing processes then work is being done. One measure of work is the number of processes that are completed per unit of time called throughput. For long processes, this rate may be one process per hour, for short transactions throughput might be 10 processes per second.

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

Insta : v2vedtech | App Link | v2vedtech.com

3) Turnaround Time (TAT):

The interval from the time of submission of a process to the time of completion is the turnaround time. From the point of view of a particular process, the important criterion is how long it takes to execute that process. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O.

4) Waiting Time:

Waiting time is defined as, the time spent by a process while waiting in the ready queue. A process waits in ready queue till CPU is allocated to it. Once the CPU is allocated the process, it starts its execution and if required requests for resources. When I/O request completes, it goes back to ready queue. In ready queue again it waits for CPU allocation.

5) Balanced Utilization:

Balanced utilization is defined as, the percentage of time all the system resources are busy. It considers not only the CPU utilization but the utilization of I/O devices, memory, and all other resources. To get more work done by the system, the CPU and I/O devices must be kept running simultaneously.

6) Response Time:

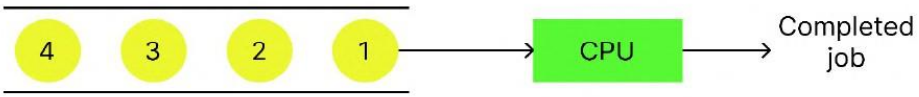
Response time is defined as, the time elapsed between the moment when a user initiates a request and the instant when the system starts responding to this request. It is the amount of time it takes to start responding, but not the time that it takes to output that response.

➢ 3.3 Types of Scheduling algorithms:

- First Come First Serve (FCFS)
- Shortest Job First (SJF)
- Shortest Remaining Time Next (SRTN)
- Round Robin (RR)
- Priority Scheduling
- Multilevel Queue Scheduling

1. First Come First Serve (FCFS)

It is the simplest type of algorithm in which the process that requests the CPU first is allocated the CPU first. In FCFS scheduling algorithm processes are scheduled in the order they are received. The FCFS algorithm is easily implemented by using a queue data structure for the ready queue.



It can be implemented with FIFO (First In First Out (FIFO)) queue. The FCFS scheduling algorithm is non-pre-emptive. Once the CPU has been allocated to a process, that process keeps the CPU until it wants to release the CPU, either by terminating or by requesting I/O.

Advantages of FCFS:

- FCFS is easier to understand and implement as processes are simply to be added at the end and removed from the front of queue.
- FCFS is well suited for batch systems.

Disadvantages of FCFS:

- Average waiting time is very large.
- FCFS is not an attractive alternative on its own for a single processor system.

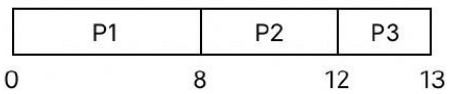
Example:

Find average waiting time and turn around time.

process	Arrival time	Burst time
P1	0	8
P2	1	4
P3	2	1

Process	Arrival time	Burst time	Completion time	Turnaround time	Waiting time
P1	0	8	8	8	0
P2	1	4	12	11	7
P3	2	1	13	11	10

Gantt chart:



Completion time:

Completion time is calculated by using Gantt chart

Turnaround time:

Turnaround time = Completion time - Arrival time

P1 -> 8 - 0 = 8 ms

P2 -> 12 - 1 = 11 ms

P3 -> 13 - 2 = 11 ms

Average turn around time:

Average turn around time = Turnaround time of all processes / No. of processes

= (8 + 11 + 11) / 3

= 10 ms

Waiting time:

Waiting time = Turnaround time - Burst time

P1 -> 8 - 8 = 0 ms

P2 -> 11 - 4 = 7 ms

P3 -> 11 - 1 = 10 ms

Average waiting time:

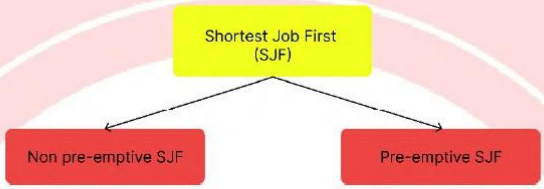
Average waiting time = waiting time of all processes / No. of processes

= (0 + 7 + 10) / 3

= 5.67 ms

2. Shortest Job First (SJF)

The SJF scheduling algorithm is also known as Shortest Process Next (SPN) scheduling algorithm or Shortest Request Next (SRN) scheduling algorithm that schedul the processes according to the length of the CPU burst they require. In SJF algorithm, the process with the shortest expected burst time is assigned to CPU. Hence, the name is shortest Job First. Jobs or processes are processed in the ascending order of their CPU burst times. Every time the job with smallest CPU burst-time is selected from the ready queue. If the two processes having same CPU burst time then they will be scheduled according to FCFS algorithm.



1. Non pre-emptive SJF:

In this method, if CPU is executing one job, it is not stopped in between before completion.

2. Pre-emptive SJF:

In this method, while CPU is executing a job, if a new job arrives with smaller burst time, then the current job is pre-empted and the new job is executed. It is also called Shortest Remaining Time First (SRTF).

Example:

Calculate average Turnaround time and waiting time from following:

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

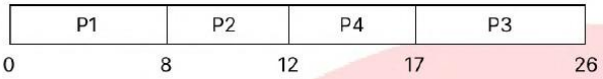
Answer:

Using non pre-emptive SJF

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P1	0	8	8	8	0

P2	1	4	12	11	7
P3	2	9	26	24	15
P4	3	5	17	14	9

Gantt chart:



Completion Time:

Compile completion time is calculated by using Gantt chart.

Turnaround time:

Turnaround time = completion time - arrival time

P1 -> 8 - 0 = 8 ms

P2 -> 12 - 1 = 11 ms

P3 -> 26 - 2 = 24 ms

P4 -> 17 - 3 = 14 ms

Average Turnaround time = Turnaround time of all processes / No. of processes

= (8 + 11 + 24 + 14) / 4

= 57 / 4

= 14.25 ms

Waiting Time:

Waiting Time = Turnaround Time - Burst Time

P1 -> 8 - 8 = 0 ms

P2 -> 11 - 4 = 7 ms

P3 -> 24 - 9 = 15 ms

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

Insta : v2vedtech | App Link | v2vedtech.com

P4 -> 14 - 5 = 9 ms

Average waiting time = Waiting time of all processes / No. of processes

= (0 + 7 + 15 + 9) / 4

= 31 / 4

= 7.75 ms

Using pre-emptive scheduling

process	Arrival Time	Burst Time	completion Time	Turnaround Time	waiting Time
P1	0	8	17	17	
P2	1	4	5	4	
P3	2	9	26	24	
P4	3	5	10	7	

Gantt chart:

Completion time:

Completion time is calculated by using Gantt chart.

Turnaround Time:

Turnaround Time = completion Time - Arrival Time

P1 -> 17 - 0 = 17 ms

P2 -> 5 - 1 = 4 ms

P3 -> 26 - 2 = 24 ms

P4 -> 10 - 3 = 7 ms

Average Turnaround time = Turnaround time of all processes / No. of processes

= (17 + 4 + 24 + 7) / 4

= 13 ms

Waiting time:

waiting time = Turnaround Time - Burst Time

P1 -> 17 - 8 = 9 ms

P2 -> 4 - 4 = 0 ms

P3 -> 24 - 9 = 15 ms

P4 -> 7 - 5 = 2 ms

Average waiting Time = Waiting time of all processes / Number of processes

= (9 + 0 + 15 + 2) / 4

= 26 / 4

= 6.5 ms

Attempt:

How pre-emptive scheduling is better than non-pre-emptive scheduling by solving your problem using SJF (Solve it by using the preemptive SJF and non-pre-emptive SJF also). (W - 23)

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Following are the reason, why pre-emptive scheduling is better than non-preemptive scheduling.

Pre-emptive scheduling is quite flexible because critical processes are allowed to access the CPU because they come in the ready queue and no matter which process is currently running.

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

Insta : v2vedtech | App Link | v2vedtech.com

Non-pre-emptive scheduling is tough because if an essential process is assigned to the ready queue, the CPU process is not be interrupted.

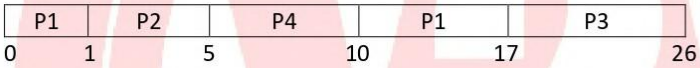
Pre-emptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process.

Non-pre-emptive scheduling ensures that a process relinquishes control of the CPU only when its finishes with its current CPU burst.

Pre-emptive SJF:-

Process	Arrival Time	Burst Time	Waiting Time (ms)	Turnaround time (ms)
P1	0	8	10 - 1 = 9	17 - 0 = 17
P2	1	4	1 - 1 = 0	5 - 1 = 4
P3	2	9	17 - 2 = 15	26 - 2 = 24
P4	3	5	5 - 3 = 2	10 - 3 = 7

Gantt chart:



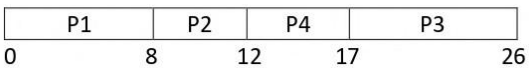
Average waiting time = Waiting time of all processes / Number of processes
= (9 + 0 + 15 + 2) / 4
= 26 / 4
= 6.5 ms

Average Turn-Around Time = Turn-Around time of all processes / Number of processes
= (17 + 4 + 24 + 7) / 4
= 52 / 4
= 13 ms

Non-pre-emptive SJF:

Process	Arrival Time	Burst Time	Waiting Time (ms)	Turnaround Time (ms)
P1	0	8	0	8 - 0 = 8
P2	1	4	8 - 1 = 7	12 - 1 = 11
P3	2	9	17 - 2 = 15	26 - 2 = 24
P4	3	5	12 - 3 = 9	17 - 3 = 14

Gantt chart:



Average waiting time = waiting time of all processes / Number of process
= (0 + 7 + 15 + 9) / 4
= 31 / 4
= 7.75 ms

Average Turn-Around time = Turn-Around time of all processes / Number of process
= (8 + 11 + 24 + 14) / 4
= 57 / 4
= 14.25 ms

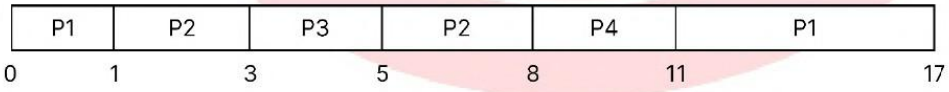
3. Shortest Remaining Time Next (SRTN)

SRTN scheduling algorithm is a pre-emptive form of Shortest Job First (SJF) scheduling algorithm. The shortest remaining time next also known as shortest Time to Go (STG) scheduling algorithm. In this algorithm a job is chosen whose burst time is the shortest. For a new job, its burst time is compared with burst time of current job. If new job needs less time to complete than the current job, then, the current job is blocked and the new job is run. It is used for batch systems.

Example:

Processes	Arrival Time	CPU Burst
P1	0	7
P2	1	5
P3	3	2
P4	4	3

Gantt chart:



Job	Arrival Time	Burst time	Finish Time	Turnaround Time	Waiting Time
P1	0	7	17	17	10
P2	1	5	8	7	2
P3	3	2	5	2	0
P4	4	3	11	7	4

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

Insta : v2vedtech | App Link | v2vedtech.com

			Average	33/4 = 8.25	16/4 = 4

Advantages :

1. It minimizes average waiting time.
2. More efficient for short processes.

Disadvantages :

1. It may cause starvation risk.
2. It may cause high overhead.
3. Difficult to implement.

Explain Round Robin algorithm with suitable example. (W - 19)

4. Round Robin (RR)

Round Robin is the pre-emptive process scheduling algorithm. The Round Robin scheduling algorithm is designed especially for time sharing systems. Processes are dispatched in a First In First Out (FIFO) sequence but each process is allowed to run for only a limited amount of time. A small unit of time called a Time Quantum or Time Slice is defined. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating CPU to each process for a time interval of up to 1 time quantum. New processes are added to the tail of the ready queue.

Example:

Process	Burst Time
P1	24
P2	3
P3	3

Time quantum : 4ms

The resulting Round Robin schedule is as follows:

P1	P2	P3	P1	P1	P1	P1	P1

0 4 7 10 14 18 22 26 30

CPU is allocated to process P1 for 4ms. Since it requires another 20 milliseconds, it is preempted after the first time quantum and the CPU is given to the next process in the queue, process P2. Process P2 does not need 4 milliseconds, so it quits before its time quantum expires. The CPU is then given to the next process, process P3. Once each process has received 1 time quantum, the CPU returns to process P1 for an additional time quantum.

Advantages of Round Robin scheduling algorithm

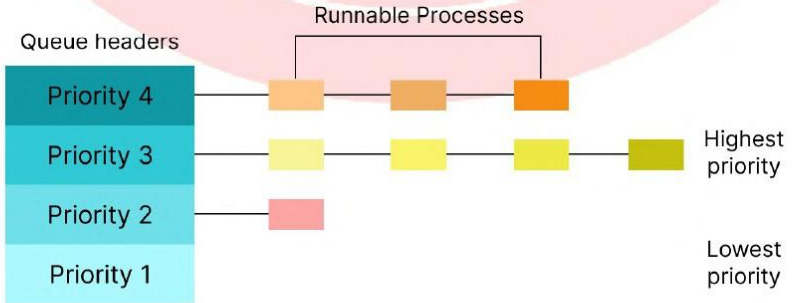
1. It is efficient for time sharing systems.
2. Round Robin increases the fairness among the processes.
3. In Round Robin scheduling algorithm overhead on processor is low.

Disadvantages of Round Robin scheduling algorithm

1. In Round Robin the processes may take long time to execute.
2. In Round Robin scheduling care must be taken in choosing quantum value.
3. Throughput in Round Robin scheduling algorithm is low if time quantum is too small.

5. Priority Scheduling

A priority is associated with each process and the CPU is allocated to the process with the highest priority, hence it is called Priority scheduling. Equal priority processes are scheduled in FCFS order. The priority scheduling can be either pre-emptive or non-pre-emptive. When process enters into the ready queue. Its priority is compared with the priority of the current running process. In a pre-emptive priority scheduling algorithm, at any time. The CPU is allocated to process if the priority of the newly arrived process is higher than the priority of the currently running process. In non-preemptive priority scheduling algorithm if priority of newly arrived process is higher than also currently running process with low priority is not get interrupted



Advantages of Priority scheduling Algorithm:

- It is simple to use
- Important processes are never made to wait because of the execution of less important processes.

Disadvantages of Priority scheduling Algorithm:

- It suffers from the problem of starvation of lower priority processes.
- Apriority scheduling can leave some low priority waiting processes indefinitely for CPU.

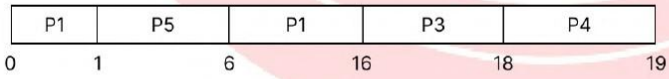
Example :

Consider the following set of processes assumed to have arrived at time 0 in the order P1, P2, P3, ---- Ps, with the length of the CPU burst time given in milliseconds. Calculate the average TAT and waiting Time

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Process	Burst Time	Priority	Completion Time	Turnaround Time	Waiting Time
P1	10	3	16	16	6
P2	1	1	1	1	0
P3	2	3	18	18	16
P4	1	4	19	19	18
P5	5	2	6	6	1

Gantt chart:



Completion Time:

Completion time is calculated by using Gantt chart

Turnaround Time:

Turnaround Time = Completion Time - Arrival Time

P1 = 16 - 0 = 16 ms

P2 = 1 - 0 = 1 ms

P3 = 18 - 0 = 18 ms

P4 = 19 - 0 = 19 ms

P5 = 6 - 0 = 6 ms

Average Turnaround Time = Turnaround time of all processes / Number of Processes

= (16 + 1 + 18 + 19 + 6) / 5

= 12ms

Waiting Time:

Waiting Time = Turnaround Time - Burst Time

P1 = 16 - 10 = 6 ms

P2 = 1 - 1 = 0 ms

P3 = 18 - 2 = 16 ms

P4 = 19 - 1 = 18 ms

P5 = 6 - 5 = 1 ms

Average Waiting Time = Waiting Time of all processes / Number of processes

= (6 + 0 + 16 + 18 + 1) / 5

= 8.2ms

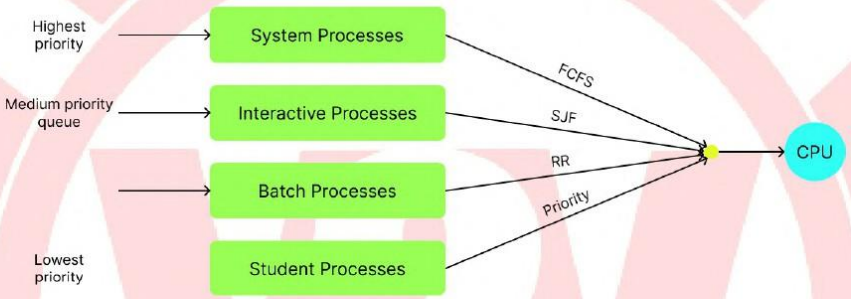
With neat diagram explain multilevel queue scheduling. (S - 23)

6. Multilevel Queue Scheduling

Multilevel queue scheduling classifies processes into different groups. It partitions the ready queue into several separate queues. The processes are permanently assigned to one queue based on some properties such as memory size, priority, process type, etc. Each queue has its own scheduling algorithm. In a system there are foreground processes and background processes. So system can divide processes into two queues: one for background and other for foreground. The interactive processes are known as foreground processes and the batch processes are known as background processes. Foreground queue can be scheduled with Round Robin algorithm where as background queue can be scheduled by First Come First Serve algorithm.

Example:-

Consider all the processes in the system are divided into four groups system, interactive, batch and student processes queue. Each queue contains processes. CPU based on scheduled queues may be priority, total burst time or process types.



Multilevel priority queue scheduling with scheduling algorithm

Advantages MLQ Scheduling Algorithm:

1. In MLQ the processes are permanently assigned to their respective queues and do not move between queues. This results is low scheduling overhead.
2. In MLQ one can apply different scheduling algorithms to different processes.
3. There are many processes which we are not able to put them in the one single queue which is solved by MLQ scheduling as we can now put them in different queues.

Disadvantages MLQ Scheduling Algorithm:

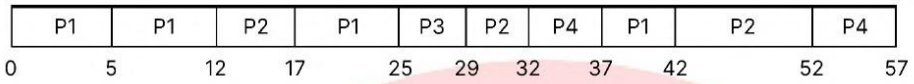
1. The processes in lower priority queues may have to starve for CPU in case processes are continuously arriving in higher priority queues.
2. In MLQ the process does not move from one queue to another queue.

Consider following table, which contains four processes P1, P2, P3, and P4 with their arrival times and required CPU burst (in milliseconds):

Process	Arrival Time	CPU Burst
P1	0	25

P2	12	18
P3	25	4
P4	32	10

Gantt Chart:



Waiting Time

- Waiting time of Process P1 = 5 + 12 = 17
- Waiting time of Process P2 = 12 + 10 = 22
- Waiting time of Process P3 = 0
- Waiting time of Process P4 = 52 - 37 = 15
- Average Waiting Time = (17 + 22 + 0 + 15)/4 = 54/4 = 13.5 msec.

Turnaround Time

- TAT of process P1 = 42 - 0 = 42
- TAT of process P2 = 52 - 12 = 40
- TAT of process P3 = 29 - 25 = 4
- TAT of process P4 = 57 - 32 = 25
- Average TAT = (42 + 40 + 4 + 25)/4 = 111/4 = 27.75 msec.

Compare Short Job First (SJF) and Shortest Remaining Time (SRTN) Scheduling (S - 24)

Short Job First (SJF)	Shortest Remaining Time (SRTN)
1. It is a non-preemptive algorithm	1. It is a preemptive algorithm
2. It involves less overhead than SRTN.	2. It involves more overheads than SJF.
3. It lead to comparatively lower throughput.	3. It leads to increased throughput as execution time is less.
4. It minimizes the average waiting time for each process.	4. It may or may not minimize the average waiting time for each process.

5. It involves lesser number of context switching.	5. It involves higher number of context switching.
6. Short processes are executed first and then following by longer processes.	6. Shorter processes run fast and longer processes show poor response time.

Following are different questions asked in exam regarding different types of scheduling :

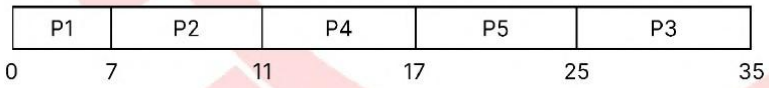
The jobs are scheduled for execution as follows:

Process	Arrival Time	Burst Time
P1	0	7
P2	1	4
P3	2	10
P4	3	6
P5	4	8

i) SJF ii) FCFS
Also find average waiting time using Ganatt chart.

Answer :

SJF Non-Preemptive SJF:
Gantt chart:

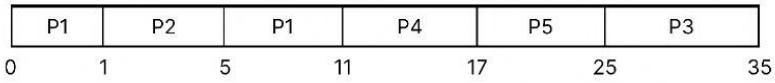


Process	Arrival Time	Burst Time	Waiting Time
P1	0	7	0
P2	1	4	7-1 = 6
P3	2	10	25-2 = 23
P4	3	6	11-3 = 8
P5	4	8	17-4 = 13

Average waiting time = waiting time of all process / total number of process
= (0+6+23+8+13)/5
= 50/5
= 10

Preemptive SJF

Gantt Chart:



Process	Arrival Time	Burst Time	Waiting Time
P1	0	7	0 + (5 - 1) = 4
P2	1	4	1 - 1 = 0
P3	2	10	25 - 2 = 23
P4	3	6	11 - 3 = 8
P5	4	8	17 - 4 = 13

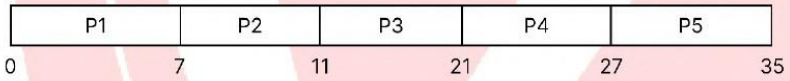
Average waiting Time = waiting time of all processes / Total number of processes

= (4+0+23+8+13) / 5

= 9.6

ii) FCFS:

Gantt chart:



Process	Arrival Time	Burst Time	Waiting Time
P1	0	7	0 - 0 = 0
P2	1	4	7 - 1 = 6
P3	2	10	11 - 2 = 9
P4	3	6	21 - 3 = 18
P5	4	8	27 - 4 = 23

Average waiting time = waiting time of all processes / Total number of processes

= (0 + 6 + 9 + 18 + 23) / 5

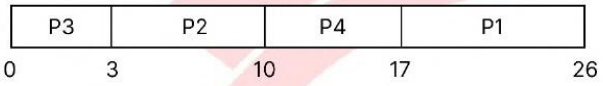
= 56 / 5

= 11.2

Solve given problem by using SJF and FCFS scheduling algorithm using Gantt chart. Calculate the average waiting time for each algorithm.

Process	Burst time (in ms)
P1	9
P2	7
P3	3
P4	7

Gantt Chart SJF:



waiting Time:

P1 = 17

P2 = 3

P3 = 0

P4 = 10

Average waiting time = waiting time of all processes / Number of processes

= (17+3+0+10) / 4

= 30/4

= 7.5 milliseconds (ms)

Gantt chart FCFS:

Waiting Time:

P1 = 0

P2 = 9

P3 = 16

P4 = 19

Average waiting time = waiting time of all processes / Number of processes

= (0 + 9 + 16 + 19) / 4

= 44 / 4

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

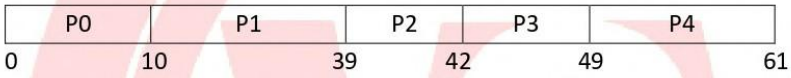
Insta : v2vedtech | App Link | v2vedtech.com

= 11 milliseconds (ms)

Solve given problem by using FCFS Scheduling algorithm. Draw Correct Grantt chart & Calculate average waiting time and average turnaround time.

Process	Arrival Time	Burst Time
P0	0	10
P1	1	29
P2	2	3
P3	3	7
P4	4	12

Gantt Chart



FCFS
waiting time
P0 = 0
P1 = (10 - 1) = 9
P2 = (39 - 2) = 37
P3 = (42 - 3) = 39
P4 = (49 - 4) = 45

Average waiting time = Waiting time of all process / Number of process
= (0 + 9 + 37 + 39 + 45) / 5
= 26 ms

Turnaround Time
P0 = 10
P1 = 39 - 1 = 38
P2 = 42 - 2 = 40
P3 = 49 - 3 = 46
P4 = 61 - 4 = 57

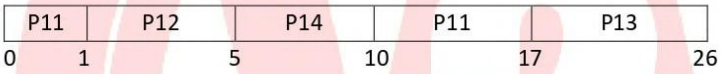
Average Turnaround time = Turnaround time of all process / Number of process

= (10 + 38 + 40 + 46 + 57) / 5
= 191 / 5
= 38.2 ms

Attempt Solve given problem by using i) Pre-emptive SJF ii) Round Robin (Time Slice = 3 ms)
Calculate average waiting time using Gannt chart.

Process	A.T.	B.T.(in ms)
P11	0	8
P12	1	4
P13	2	9
P14	3	5

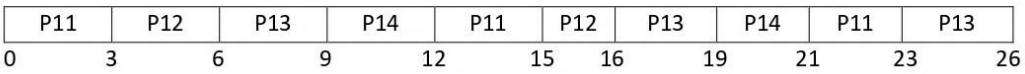
i) Pre-emptive SJF:



Waiting Time = (Total Completion time - Burst time) - Arrival time
P11 = (17 - 8) - 0 = 9 ms.
P12 = (5 - 4) - 1 = 0 ms.
P13 = (26 - 9) - 2 = 15 ms.
P14 = (10 - 5) - 3 = 2 ms.

Average waiting time :
=(9 + 0 + 15 + 2) / 4
= 26 / 4
= 6.5 ms

ii) Round Robin (Time slice = 3 ms)



Waiting time :
P11 = (23 - 8) - 0 = 15 ms.

P12 = (16 - 4) - 1 = 11 ms.
P13 = (26 - 9) - 2 = 15 ms.
P14 = (21 - 5) - 3 = 13 ms.

Average waiting time:
=(15 + 11 + 15 + 13) / 4
= 54 / 4
= 13.5 ms

What is the average turnaround time for the following process using:

1. FCFS scheduling algorithm.
2. SJF non-preemptive scheduling algorithm.
3. Round Robin Scheduling algorithm.

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	1

i) Using FCFS:
Gantt Chart:

P1	P2	P3

0 8 12 13

Process	Arrival time	Burst time	Completion time	Turnaround time
P1	0	8	8	8
P2	1	4	12	11
P3	2	1	13	11

Completion Time:
Completion Time is calculated by using Gantt chart.

Turnaround Time:
Turnaround Time = Completion Time - Arrival Time

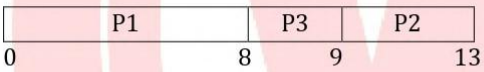
P1 = 8-0 = 8 ms
P2 = 12-1 = 11 ms
P3 = 13-2 = 11 ms

Average turnaround time = Turnaround time of all processes / Number of processes
= (8 + 11 + 11) / 3
= 10 ms / units

ii) using SJF (non-preemptive):

Process	Arrival time	Burst time	Completion time	Turnaround time
P1	0	8	8	8
P2	1	4	13	12
P3	2	1	9	7

Gantt chart:



Completion Time:
completion time is calculated by using Gantt chart

Turnaround Time:
Turnaround time = completion time - Arrival time
P1 = 8-0 = 8 ms
P2 = 13-1 = 12 ms
P3 = 9-2 = 7 ms

Average turnaround time = Turnaround time of all processes / Number of processes
= (8 + 12 + 7) / 3
= 9 ms / units

iii) using Round Robin: consider time Quantum = 2 units

Process	Arrival time	Burst time	Completion time	Turnaround time
P1	0	8	13	13
P2	1	4	9	8
P3	2	1	5	3

Gantt chart:

Completion Time:
completion time is calculated by using Gantt chart.

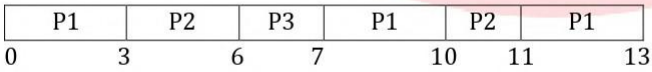
Turnaround Time:
Turnaround Time = Completion Time - Arrival Time
P1 = 13 - 0 = 13 ms
P2 = 11 - 1 = 10 ms
P3 = 7 - 2 = 5 ms

Average Turnaround Time = Turnaround time of all processes / Number of processes
= (13 + 8 + 3) / 3
= 8 ms / units

- **Considering time quantum as 3 units:**

Process	Arrival time	Burst time	Completion time	Turnaround time
P1	0	8	13	13
P2	1	4	11	10
P3	2	1	7	5

Gantt chart:



Completion time:
Completion time is calculated by using gantt chart.

Turnaround time:
Turnaround time = completion time - Arrival time
P1 = 13 - 0 = 13 ms
P2 = 11 - 1 = 10 ms
P3 = 7 - 2 = 5 ms

Average Turnaround time = Turnaround time of all processes / Number of processes
= (13 + 10 + 5) / 3
= 9.33 ms / unit

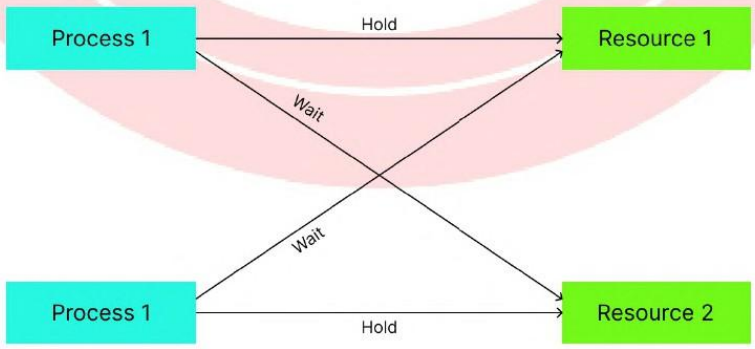
➢ 3.4 Deadlock:

Deadlock is a situation when two or more processes get locked and cannot processed further because of inter-dependability. Deadlock is defined as 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. Processes involved in a deadlock remain blocked permanently and this affect OS performance. A dead deadlock arises when the four conditions hold true simultaneously in a system.

These four conditions are as follows:

i) Mutual Exclusion
ii) Hold and Wait
iii) No Pre-emption
iv) Circular Wait

Example of Deadlock :



①Process 1 is holding Resource 1 and is waiting for Resource 2.

② Process 2 is holding Resource 2 and is waiting for Resource 1.

③ Neither process can proceed because each one is waiting for a resource that the other holds.

Following are the conditions for deadlock:

1. Mutual exclusion:
At least one resource must be held in a non-sharable mode. That is, only one process at a time can use the resource.

2. Hold and wait:
A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.

3. No pre-emption:
Resource cannot be pre-empted, i.e. resource can only be released voluntarily by the process holding it, after the process has completed its task. Resources previously granted cannot be forcibly taken away from a process.

4. Circular wait:
A set of waiting processes must exist such that P_0 is waiting for a resource held by P_1, P_1 is waiting for a resource held by P_2, ..., Pn-1 is waiting for a resource held by Pn and Pn is waiting for a resource held by P_0. Each process is waiting for the resource held by other waiting processes in circular form.

Following are the methods of deadlock prevention :

1. Eliminate Mutual Exclusion:
The mutual exclusion condition must hold for non-sharable type of resources. Shareable resources do not require mutually exclusive access, and thus cannot be involved in deadlock. For example, a file shared on network of computers, with read-only access can be open by multiple users at a time. It is not possible to prevent deadlock by denying the mutual exclusion condition.

2. Eliminate Hold and Wait:
One way to avoid this Hold and wait is when a process requests a resource it does not hold any other resources. One protocol that can be used requires each process to request and be allocated all its resources before it begin execution. Another protocol that can be used is, to allow a process to request resources only when the process has none. A process may request some resources and use them. Before it requests any additional resources, it must release all the resources that are currently allocated to it.

3. Eliminate No Preemption:
If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are preempted. That is, this resources are implicitly released. The preempted resources are added to the list of resources for which the process

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

Insta : v2vedtech | App Link | v2vedtech.com

is waiting. Process will be restarted only when all the resources, i.e. its old resources, as well as the new ones that it is requesting will be available.

4. Eliminating Circular Wait condition:
The circular wait condition of deadlock can be eliminated by assigning a priority number to each available resource and a process can request resources only in increasing order. If the priority number of a requested resource is greater than that of all the currently held resources, the request is granted. If the priority number of a requested resource is less than that of all the currently held resources, all the resources with greater priority number must be released first, before acquiring the new resource.

Number	Resource Name
0	Tape Drive
1	Printer
2	Plotter
3	Card Reader
4	Card Punch

System Models :

Resources and Processes: A system has a limited number of resources (like CPU, memory, printers, files)
which are divided into types, each with multiple identical instances. Various processes compete for these resources.

Resource Requests and Release: Processes must request a resource before using it and release it after use. If a requested resource is not immediately available, the process must wait.

Resource Limitations: The total number of resources requested by processes cannot exceed the total number available in the system.

Resource Management Cycle: The typical sequence for a process using a resource is:

- **Request:** The process asks for a resource.
- **Use:** If granted, the process operates on the resource.
- **Release:** The process relinquishes the resource.

System Calls and Management: Resource requests and releases are handled through system calls provided by the operating system (e.g., request(), release(), open(), close(), allocate(), free()).

Synchronization Mechanisms: Resource management, particularly for resources not directly managed by the OS, can be achieved using synchronization mechanisms like semaphores (through wait() and signal() operations) or mutex locks.

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

Insta : v2vedtech | App Link | v2vedtech.com

Operating System's Role: The operating system is responsible for verifying that a process has indeed requested and been allocated a resource before allowing it to use it.

System Table for Resource Tracking: The OS maintains a system table that records the status of each resource:

- Whether the resource is free or allocated.
- If allocated, it also records which process the resource is allocated to.

Handling Unavailable Resources: If a process requests a resource that is currently allocated to another process, the requesting process is added to a queue of processes waiting for that particular resource.

Deadlock Handling :

Deadlock avoidance methods :

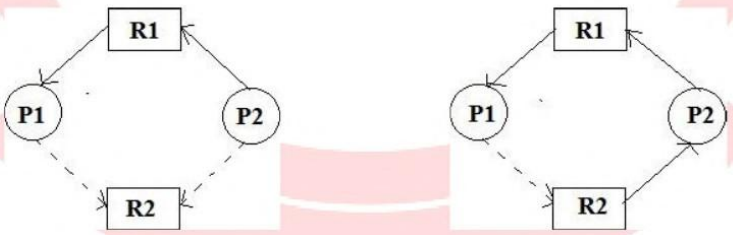
1. Safe State :

If a system is already in a safe state, we can try to stay away from an unsafe state and avoid deadlock. Deadlocks cannot be avoided in an unsafe state. A safe sequence of processes and allocation of resources ensures a safe state. Deadlock avoidance algorithms try not to allocate resources to a process if it will make the system in an unsafe state.

2. Resource Allocation Graph :-

A resource allocation graph is generally used to avoid deadlocks. If there are no cycles in the resource allocation graph, then there are no deadlocks. If there are cycles, there may be a deadlock. The resource allocation graph has request edges and assignment edges. Based on calm edges we can see if these is a chance for a cycle and then grant requests if the system will again be in a safe state.

Example :



If R2 is allocated to P2 and if P2 is request for R2, there will be a deadlock.

3. Banker's Algorithm :

In this algorithm, every process must tell upfront the maximum resource of each type it need. The Banker's algorithm can be divided into two parts: safety algorithm if a system is in a safe state or not. The resource request algorithm make an assumption of allocation and see if the system will be in a safe state. If the new state is unsafe, the resources are not allocated and the data structures are restored to their previous state. In this case the processes must wait for the resource.

Example :

5 Processes P_0 through P_4 :

3 Resources types: A (10 Instances), B (5 Instances) and C (7 Instances)

Process	Allocation	Max	Available	Remaining
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2	5 3 2	1 2 2
P_2	3 0 2	9 0 2	7 4 3	6 0 0
P_3	2 1 1	2 2 2	7 4 5	0 1 1
P_4	0 0 2	4 3 3	10 4 7	4 3 1
			10 5 7	

The system in a safe state since the sequence <P_1, P_3, P_4, P_2, P_0> satisfies safety criteria.

Deadlock avoidance- Banker's Algorithm

This algorithm calculates allocated, required and available resources before allocating resources to any process to avoid deadlock. It contains two matrices on a dynamic basis. Matrix A contains resources allocated to different process at a given time. Matrix B maintains the resources which are still required by different processes at the same time.

Following are the steps of Banker's algorithm to avoid deadlock.

Step 1: When a process requests for a resource, the OS allocates it on a trial basis.

Step 2: After trial allocation, the os updates all the matrices and vectors. This updating can be done by the OS in a separate work area in the memory.

Step 3: It compares free resources Vector with each row of matrix B on a vector to vector basis.

Step 4: If free resources is smaller than each of the row in Matrix B i.e. even if all free resources are allocated to any process in Matrix B and not a single process can complete its task then as concludes that the System is in unstable State.

Step 5: If free resources is greater than any row for a process in Matrix B the os allocates all required resources for that process on a trial basis. It assumes that after completion of process, it will release all the resources allocated to it. These resources can be added to the free vector.

Mob No : 9326050669 / 9372072139 | Youtube : @v2vedtechllp|

Insta : v2vedtech | App Link | v2vedtech.com

Step 6: After execution of a process, it removes the row indicating executed process from both matrices.

Step 7: This algorithm will repeat the procedure Step 3 for each process from the matrices and finds that
all processes can complete execution without entering unsafe state. For each request for any resource by a process os goes through all these trials of imaginary allocation and updates on After
this if the system remains in the safe state, and then changes can be made in actual matrices.

Example:
Find out the safe sequence of processes using bankers algorithm for following problem

Process	Allocation	Max Need	Available	Remaining Need
	A B C	A B C	A B C	A B C
P1	0 0 1	2 2 2	4 2 2	2 2 1
P2	0 1 0	3 3 3	4 2 3	3 2 3
P3	1 0 0	1 1 1	4 3 3	0 1 1
P4	0 0 2	2 2 2	5 3 3	2 2 0
P5	0 0 0	3 3 3	0 3 5	3 1 3
			5 5 5	

A, B, C are representing resources.
Total Resources: A = 5
B = 5
C = 5

so, the safe sequence is
P1 -> P2 -> P3 -> P4 -> P5