NAME- SUSNATO BARUA

ROLL- 1854044

DEPT- IT

CHAPTER- Process Scheduling.

| CPU Burst | IO Burst |
|---|---|
| a) When it is executing instructions, it is a CPU burst. | a) When it services requests to fetch information, it is I/O burst. |
| b) CPU can ~~ritten~~ instructions from cache until it needs to fetch more instructions or data from memory. Here, CPU burst ends & IO burst starts | b) The IO burst reads or writes data until the requested data is read/written or the space to store it cache runs out. That ends on IO burst. |
| c) CPU Bursts are relatively short. | c) IO bursts are relatively long. |

| 2. Pre-emptive | Non Pre-emptive |
|---|---|
| a) Processes can be interrupted | a) Processes can't be interrupted during execution. |
| b) If a process having high priority frequently arrives in ready queue, low priority process may starve. | b) If a process with long burst time is running in CPU, then later coming processes with lower burst time may starve. |

| | |
|---|---|
| c) It has overheads of scheduling the processes. | c) It doesn't have overheads. |
| d) CPU utilization is high. | d) Low CPU utilization. |
| e) High cost | e) No cost. |
| f) Flexible in nature. | f) Rigid. |

3. A dispatcher is a special program which comes into play after the ~~short~~ scheduler when the scheduler completes its job of selecting a process, it is the dispatcher which takes that process to the desired state/queue. The dispatcher is the module that gives a process control over the CPU after it has been selected by the short-term scheduler. This function involves the following major tasks:-

a) Switching context.
b) switching to user mode.
c) jumping to the proper location in the user program to restart the program.

4. The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selecting of another process on the basis of a particular strategy.
The process scheduling criteria are following:-

i) CPU utilisation:- The best scheduling algorithm or a good one would be considered that one which will keep the CPU busy as much as possible. In a real-time system, it varies from 40 to 90% depending on the load upon length or duration of process.

iii) Turn-around time:- The time interval from the time of submission of a process to the time of completion, is known as turnaround time. It is the sum of times spent waiting to get into memory, waiting in ready queue, executing in CPU & waiting for I/O.

iv) Waiting Time:- A scheduling algorithm doesn't affect the time required to complete the process once it starts execution. It only affects the waiting in the ready queue.

v) Response Time:- In an interactive system, a process may produce some o/p fairly early and continue computing new results while previous results are being output to the user. Thus, another criteria is the time taken from submission of the process of request until the first response is produced. This is called response time.

5 a) FCFS:- It stands for First Come First Serve. It is a scheduling algorithm that automatically executes queued requests and processes in order of their arrival.

i) It uses queue data structure.
ii) It is easy to understand and implement.
iii) It is poor in performance as waiting time is high.
iv) It supports both pre-emptive and non pre-emptive scheduling algorithm.

Eg:- In real life, when we go to a railway ticket counter, one who arrives in the queue first gets the ticket. CPU uses the same algorithm in FCFS.

b) SJF scheduling: This algorithm executes the process having the smallest execution time.

i) It is associated with each job as a unit of time to complete.

ii) It is frequently used for long-term scheduling.

iii) It reduces the avg. waiting time over FCFS algorithm.

iv) Job completion time must be known earlier, but it's hard to predict.

Eg:- Non pre-emptive SJF

| Process | BT | AT |
|---|---|---|
| $P_1$ | 3 | 0 |
| $P_2$ | 8 | 1 |
| $P_3$ | 6 | 2 |

→ At time 0, $P_1$ starts executing. Till its completion, both $P_2$ and $P_3$ will be in ready queue. So, $P_3$ will start execution having low burst time & then $P_2$ will execute.

c) SRTF Scheduling: It is the pre-emptive version of shortest job first algorithm.

i) Process with the least remaining execution time is executed first.

ii) It is claimed to be better than SJF scheduling algorithm.

iii) There is no ~~spec~~ specific method to predict the length of the upcoming CPU burst.

Eg:-

| Pid | BT | AT |
|---|---|---|
| $P_1$ | 8 | 0 |
| $P_2$ | 2 | 1 |
| $P_3$ | 3 | 4 |

$P_1$ will start executing. After 1 ms, $P_1 \rightarrow BT(7)$ & $P_2 \rightarrow BT(2)$. Hence, $P_2$ will execute till its completion then $P_1$ will execute for another $m_3$, then $P_1 \rightarrow BT(6)$ & $P_3 \rightarrow BT(3)$. $P_3$ will start execution till its completion and then $P_1$ will complete.

d) Priority Scheduling :- It is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the task to work as per the priority.

a) i) It is used in Os for performing batch processes.

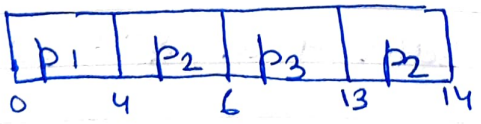
b) ii) If two jobs are having the same priority & ready, it works on FCFS basis.

iii) Lower the number, higher is the priority.

iv) If newer process arrives having a higher priority than the currently running process, then running process is pre-emptive.

Eg :-

| Pid | BT | Priority | AT |
|---|---|---|---|
| $P_1$ | 4 | 1 | 0 |
| $P_2$ | 3 | 2 | 0 |
| $P_3$ | 7 | 1 | 6 |




e) RR Scheduling :- It is the oldest, simplest scheduling algorithm, which is mostly used for multi-tasking.

i) It is a pre-emptive algorithm.

ii) The CPU is shifted to the next process after time

quantum.

iii) It is a real time algorithm which responds to the event within a specific time limit.

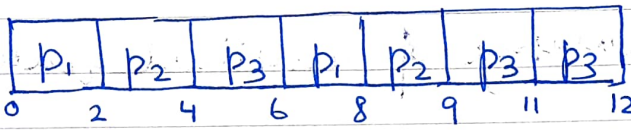
iv) The process that is ~~pre-emptive~~ pre-empted is added to the end of the queue.

Eg:-

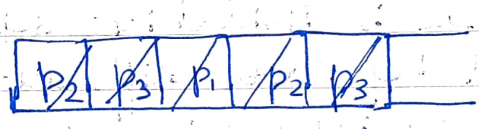
| Pid | BT |
|---|---|
| $P_1$ | 4 |
| $P_2$ | 3 |
| $P_3$ | 5 |

TQ = 2.

Gantt Chart




Waiting queue




6. Convoy effect is a phenomenon in which the whole operating system slows down due to few slow processes. FCFS scheduling algorithm is effected with it as the 1st process may have the highest burst time and shorter burst time jobs are waiting behind resulting in the slowing of the system. This is known as convoy effect in FCFS.

7. The major problem of priority scheduling algorithm is indefinite block or starvation. If the scene comes as high priority processes keep coming in ready queue and low priority processes may starve in ready queue waiting for their turn.

→ The problem of priority scheduling is solved by aging. It is a technique of gradually increasing the priority of processes that wait in the system for a long period of time.

8. Selection of quantum time is the most important to determined efficiency of round-robin scheduling because if quantum time would be too large, it will behave as a FCFS algorithm. If quantum time would be too small, then, it will result to the frequently switching of the processes leading to the high system overheads and eventually, the efficiency of the system would be low. Hence, a descent quantum time is necessary.

9. The advantage of multilevel queue scheduling is:-

i) Ready queue is divided into separate queues which are owned by three different types of processes- System processes, interactive processes and Batch processes.

ii) Two queues follow Round Robin scheduling algorithm and the least priority queue follows FCFS.

iii) For scheduling among the queues, fixed priority pre-emptive scheduling method and time slicing method is used.

iv) The processes are permanently assigned to the queue, so it has advantage of low scheduling overhead.

The disadvantages are:-

i) Some processes may starve for CPU if some higher priority queues are never becoming empty.

ii) It is inflexible in nature.

This problem can be solved by multilevel feedback queue scheduling. In this algorithm, it allows a process to move b/w queues.

iii) If any queue is using the CPU too much, then that queue is demoted.

iv) If any queue is not getting the chance to use the CPU, that queue is promoted.

v) It is very complex but most general CPU-scheduling algorithm.

10.

| Process | AT | BT |
|---|---|---|
| $P_0$ | 0ms | 9ms |
| $P_1$ | 1ms | 4ms |
| $P_2$ | 2ms | 9ms |


0ms 1ms 2ms 5ms 13ms 22ms

$P_0 = 8ms$ $P_0 = 8ms$

$P_1 = 4ms$ $P_1 = 3ms$

$P_2 = 9ms$

∴

| | Process completion time | Turn Around Time | Waiting Time (TAT − BT) |
|---|---|---|---|
| $P_0$ | 13ms | 13 ms | 4ms |
| $P_1$ | 5ms | 4ms | 0ms |
| $P_2$ | 22ms | 20ms | 11ms |

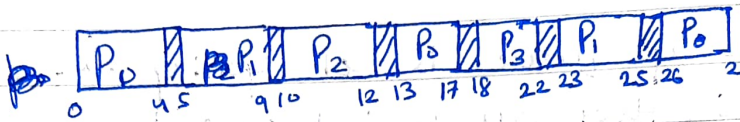
∴ Average waiting time : $\frac{(4+0+11)ms}{3} = 5\ ms$

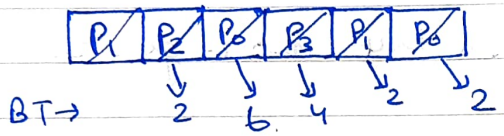
11.

| Process | Arrival time | Burst time |
|---|---|---|
| $P_0$ | 0 | 10 |
| $P_1$ | 1 | 6 |
| $P_2$ | 3 | 2 |
| $P_3$ | 5 | 4 |

Time quantum = 4ms
Context switch overhead → 1 ms

Gantt chart




| Process | Completion time | Turn Around time | Waiting time | Response time |
|---|---|---|---|---|
| $P_0$ | 28 | 28 | 18 | 0 |
| $P_1$ | 25 | 24 | 18 | 5 |
| $P_2$ | 12 | 9 | 7 | 10 |
| $P_3$ | 22 | 17 | 13 | 18 |

Average turn around time = $\frac{(28+24+9+17)}{4}$ ms =

$\left(\frac{78}{4}\right)$ ms = 19.5 ms

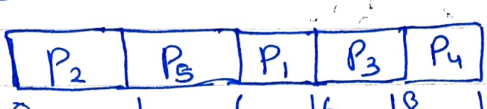
Avg waiting time = $\frac{(18+18+7+13)}{4}$ ms = $\left(\frac{56}{4}\right)$ ms = 14 ms

Avg response time = $\frac{(0+5+10+18)}{4}$ ms = $\left(\frac{33}{4}\right)$ ms = 8.25 ms

12. As the execution of the process in a non-preemptive process is not interrupted until its completion, their response time, i.e, when the process is provided the CPU, for the first time, become its waiting time because they have to wait only to get the CPU only.

13. Since type of priority scheduling isn't given, I am considering it non pre-emptive & arrival time as 0.

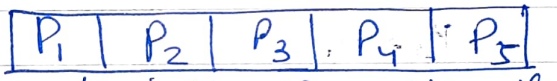
Gantt Chart:




→ Priority scheduling.

| Process | Completion time (ms) | Turn Around time (ms) | Waiting time (ms) | Response time (ms) |
|---|---|---|---|---|
| P1 | 16 | 16 | 6 | 6 |
| P2 | 1 | 1 | 0 | 0 |
| P3 | 18 | 18 | 16 | 16 |
| P4 | 19 | 19 | 18 | 18 |
| P5 | 6 | 6 | 1 | 1 |
| Avg → | | 12 ms | 8.2 ms | 8.2 ms |

FCFS Gantt chart:



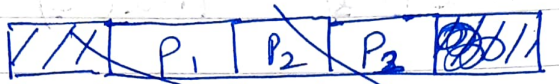

| Process | Completion time (ms) | Turn Around time (ms) | Waiting time (ms) |
|---|---|---|---|
| P1 | 10 | 10 | 0 |
| P2 | 11 | 11 | 10 |
| P3 | 13 | 13 | 11 |
| P4 | 14 | 14 | 13 |
| P5 | 19 | 19 | 14 |
| Avg → | | 13.4 ms | 9.6 ms |

→ Here, priority scheduling is more efficient than FCFS.

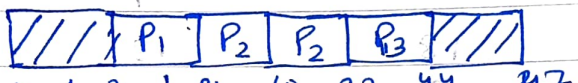
14. First 20 % of $P_1 \rightarrow 2$ sec
$P_2 \rightarrow 4$ sec
$P_3 \rightarrow 6$ sec

Last 10 % of $P_1 \rightarrow 1$ sec
$P_2 \rightarrow 2$ sec
$P_3 \rightarrow 3$ sec

Gantt Chart




Gantt Chart



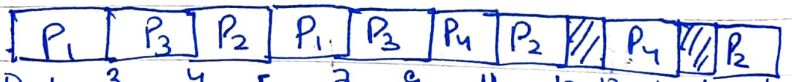

i/o process

$P_1 \rightarrow 1$ sec – I/O
$P_2 \Rightarrow$ 26
$P_3 \rightarrow 24$

Here, total for 5 sec, our CPU remain idle.

$\therefore \% = \left(\frac{5}{47} \times 100\right) = 10.638\%$

15. Gantt chart:




4 sec

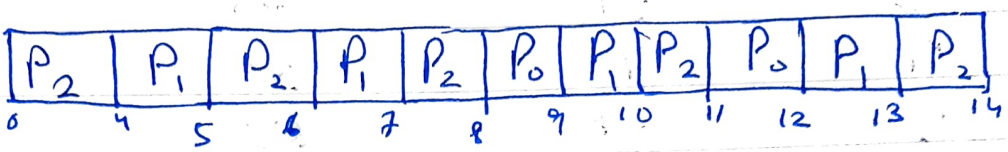
| | | | | |
|---|---|---|---|---|
| $P_2 \rightarrow$ 7 | $P_2 \rightarrow 7$ | $P_2 \rightarrow 6$ | $P_2 \rightarrow 6$ | $P_2 \rightarrow 6$ |
| $P_3 \rightarrow 6$ | $P_3 \rightarrow 4$ | $P_4 \rightarrow 5$ | $P_3 \rightarrow 2$ | $P_4 \rightarrow 5$ |
| | $P_4 \rightarrow 5$ | $P_1 \rightarrow 2$ | $P_4 \rightarrow 5$ | |

| Process | Completion Time (ms) | Turn Around time (ms) | Waiting time (ms) |
|---|---|---|---|
| $P_1$ | 7 | 7 | 0 |
| $P_2$ | 17 | 17 | 10 |
| $P_3$ | 9 | 7 | 1 |
| $P_4$ | 14 | 9 | 14 |
| Avg → | | 10 ms | 3.75 ms |

16

| P id | BT |
|---|---|
| 0 | 2 |
| 1 | 4 |
| 2 | 8 |

Gantt chart :




$P_0 \to 2$ $P_0 \to 2$ $P_0 \to 2$ $P_0 \to 2$ $P_0 \to 2$ $P_0 \to 1$
$P_1 \to 4$ $P_1 \to 3$ $P_1 \to 3$ $P_1 \to 2$ $P_1 \to 2$ $P_1 \to 1$
$P_2 \to 4$ $P_2 \to 4$ $P_2 \to 3$ $P_2 \to 3$ $P_2 \to 2$ $P_2 \to 1$

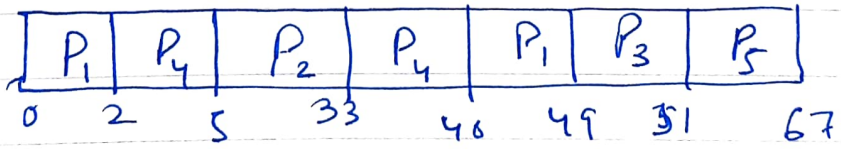
| Process | Completion Time | Turn Around Time |
|---|---|---|

Average turn around time = $\frac{39}{3} = 13$ units

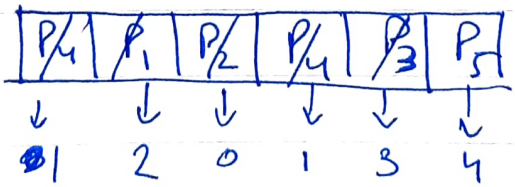
17.

| Process | Arrival time (ms) | Burst time (ms) | Priority |
|---|---|---|---|
| $P_1$ | 0 | 11 | 2 |
| $P_2$ | 5 | 28 | 0 |
| $P_3$ | 12 | 2 | 3 |
| $P_4$ | 2 | 10 | 1 |
| $P_5$ | 9 | 16 | 4 |

Gantt Chart :




Ready queue →




$P_1 \to 9$

$P_1 \to 7$

| Process | Completion time (ms) | Turn Around time (ms) | Waiting time (ms) |
|---|---|---|---|
| $P_1$ | 49 | 49 | 38 |
| $P_2$ | 33 | 28 | 0 |
| $P_3$ | 51 | 39 | 37 |
| $P_4$ | 40 | 38 | 28 |
| $P_5$ | 67 | 58 | 42 |

Avg waiting time = $\frac{38 + 0 + 37 + 28 + 42}{5} = \frac{145}{5} = 29$ ms