"

TAHSIN

OS Note's

CSE 345

"

mid 2

CPU Scheduling (36)

CPU scheduling is the basis of multiprogrammed OS. By switching the CPU among process, the operating system can make the computer more productive.

note: ✓ single-processor system → only one process can run.
→ other program have to wait.

✓ Objective of multiprogramming → is to have some process running at all times, to maximize CPU (utilization) utilization.
→ A process executed until it must wait, typically for the completion of some I/O req.

✓ We want to the CPU to be utilized, not sit idle. So, accomplishing this we need CPU scheduling.

⑧ CPU & I/O Burst cycle:

Process execution consists of a cycle of CPU execution and I/O wait.

Processes alternate two state

- Process execution begins with a CPU burst followed by I/O followed by CPU Burst, so on...

✓ **CPU Burst:** when a process being executed in the CPU.

✓ **I/O Burst:** when CPU is waiting for I/O for further execution.

ex:

load store
add store } CPU Burst
read store

wait for I/O } I/O Burst

store
increment } CPU Burst
wait for I/O } I/O Burst

!!!

✍ Preemptive & Non-Preemptive scheduling:

Note: • CPU scheduler: assigning the CPU to different Processes for their execution which are in the (ready queue) to be executed.

• Dispatcher: is a module that gives control of the CPU to the process selected by the short-term scheduler.

✓ CPU-Scheduling decisions may take place under the following four circumstances:

(i) When process switches from running state to waiting state.

(ii) " " " " running state to ready.

(iii) " " " " waiting state to ready

(iv) When a process terminates.

NB: (i) & (iv) have no choice for scheduling. (I/O req. → have to assign other process) → new process will come.

(ii) & (iii) have a choice. → (interrupt occures)

When scheduling take place only under circumstances (I) & (IV), it can be called non-preemptive or cooperative. otherwise preemptive.

preemptive
- can be given other Process because of interrupt.

Scheduling Criteria:

types 5 → (i) CPU Utilization → keep CPU busy. (0-100%) ✓ in real (40-90%)

(ii) Throughput → a measure of work done by CPU. / a measure of work in number of process that are completed per time unit.

(iii) 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. (view) (time → all states)

(iv) Waiting time: is the sum of the periods spent waiting in the ready queue.

(v) Response time: (response latency (1st response; not full)) another measure is the time from the submission of a request until the first response in produced. This measure, called response time, is the time it takes to takes to start responding, not the it takes to output the response.

Scheduling Algo:

- FIFO queue.
- First-Come, First-served scheduling – (FCFS)

The process that request the CPU first is allocated the CPU first

- When a process enters the ready queue, its PCB is linked onto the tail of the queue.
- When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue.

ex:

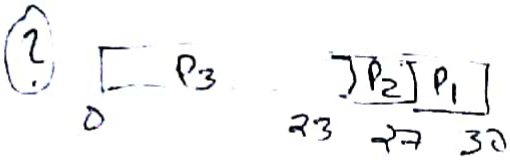
Process	Burst time
P1	24
P2	3
P3	3

⇒

Gantt chart.

⇒ Waiting time for, $P_1 = 0$ ms;
" " " , $P_2 = 24$ ms;
" " " , $P_3 = 27$ ms;

∴ Average waiting time, $\frac{(0+24+27)}{3}$
$= 17$ ms.

(?) 

⇒ avg time = ?

→

$\frac{P_1 + P_2 + P_3}{3}$ =

(W) FCFS scheduling algo is non-preemptive.

(W) Convoy Effect: if process with higher burst time arrived before the process with smaller burst time, then smaller process have to wait for a long time for longer processes to release the CPU.

(*) [Turn Around Time = Completion time − Arrival time] → entire time.

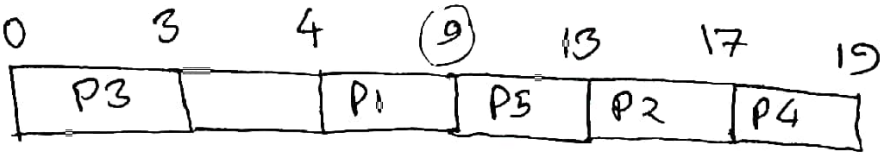
[(*) Waiting time = Turnaround time − Burst time]

(?) find Avg. waiting time and Avg. turnaround time?

ex:

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

Sol:



← chart.

[NOTE: When 2 Process comes at the same time, then the smaller process will come first. P1, P2]

→ for P1, Completion time = 9;

Turnaround time = 9 − 4 = 5

Waiting time = 5 − 5 = 0

[NOTE:
... Same goes P2, P3, P4, P5 ....]

Turnaround Avg = $\frac{5+11+3+13+8}{5}$ = 8 units

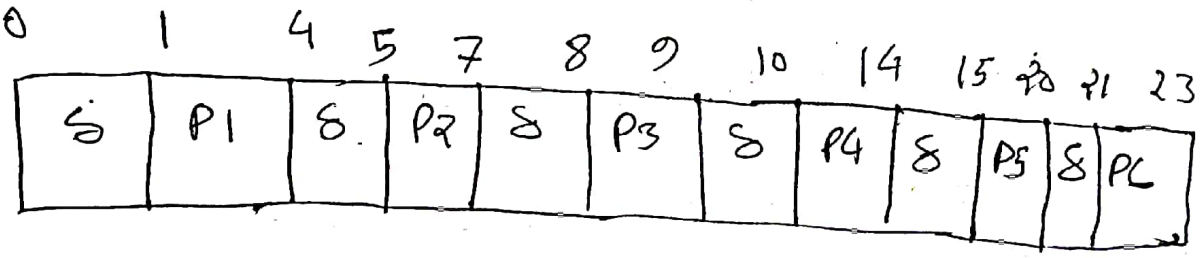
Waiting Avg = (0 + 7 + 0 + 11 + 4)/5 = 4.4 units.

(#)

Processes	Arrival Time	Burst Time
P1	0	3
P2	1	2
P3	2	1
P4	3	4
P5	4	5
P6	5	2

(?) there is 1 unit of overhead in scheduling ; find the effeiency of algorithm ?

⇒ Gantt chart:



Now, useless time or wasted time = 6 x S

= 6 x 1 = 6 units.

∴ useful time = 23 − 6 = 17 units.

∴ Efficiency = Useful time / Total time

(η) = 17/23 = 0.739 = 73.91%

SJF: Shortest-Job-First scheduling.

- This algo associates with each process the length of the process's next CPU burst.
- When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
- SJF can be either preemptive or non-preemptive.
- If 2 process are same, FCFS scheduling used to break the tie.

Ex: Burst Time

P1 → 6 ms

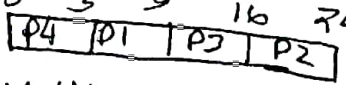
P2 → 8

P3 → 7

P4 → 3

(non-preemptive)

⇒ Gantt chart:



* Smallest Burst time.

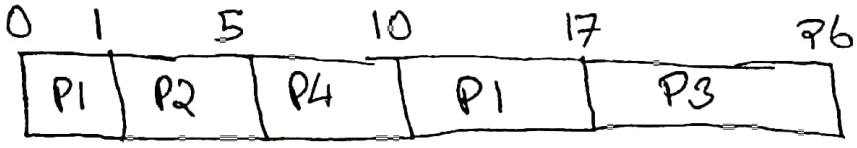
∴ Waiting time P1, P2, P3, P4 = 3, 16, 9, 0 ms.

∴ avg " = 7 ms.

Ex7: (Preemptive)

Procon ID	Arrival Time	Brust time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

⇒



[Waitin time = total waiting time − No. of procen execution − Arrival Time.]

∴ Waiting time for P1 = (10 − 1 − 0) = 9 ms

P2 = (1 − 0 − 1) = 0 ms

P3 = (17 − 0 − 2) = 15 ms

P4 = (5 − 0 − 3) = 2 ms

∴ Avg. = (9 + 0 + 15 + 2)/4 = 6.5 ms.

Priority Scheduling :-

- is associated with each process, and the CPU is allocated to the process with the highest priority.
- Equal-priority process are scheduled in FCFS.
- An SJF algo is simply a priority algo. where the priority is the inverse of the (Predicted) next CPU burst.

  The larger the CPU burst, the lower the priority and vice versa.
- Priority can be preemptive or non-preemptive.

  if the priority of the newly arrived process is higher than the priority of the currently running process,

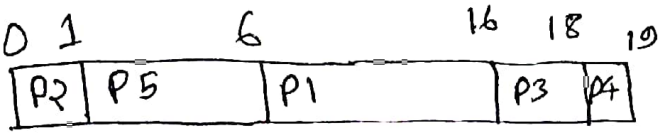
  will simply put the new process at the head of the ready queue.

ex:

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

[Assume, lower Priority in the hiest priority.]

Gantt chart :



here, waiting time for; $P1 = 6$, $P2 = 0$, $P3 = 16$, $P4 = 18$, $P5 = 1$ ms.

Avg. " $= (6 + 0 + 16 + 18 + 1) / 5$

$= 8.2$ ms.

✓

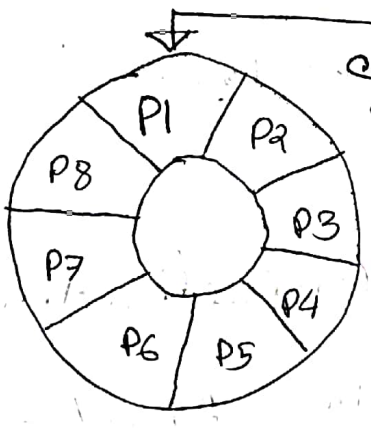
* Cons of Priority Alg. → ◦ indefinite blocking or starvation.

: low priority waiting indefinitely.

↳ solve → (aging)

Round-Robin Scheduling :

- especially for time sharing system.
- It's similar for FCFS scheduling, but Preemption is added to switch between Procen.
- The ready queue is treated as a circular queue.



Tail ○ ○ ○ ○ Head

FIFO

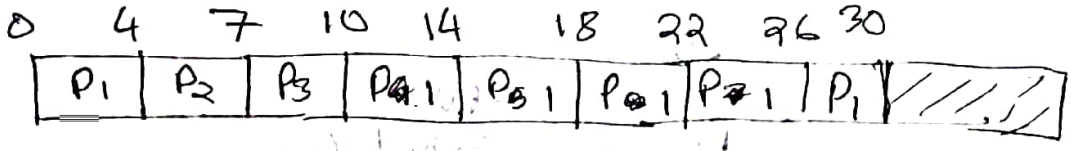
(4) Round Robin Scheduling-

Turnaround & Waiting Time :

Process	Burst Time
P1	24
P2	3
P3	3

[Time quantum = 4 ms]

⇒ gantt chart :



Rules : (1)

- Turn Aroun Time = Completion Time - Arrival Time
- Waiting Time = Turn Around time - Burst Time

OR

Rules : (11) Waiting time = Last start Time - Arrival time - (Preemption x Time quantum)

(For Rule 1)

Procen ID	completion Time	Turnaround Time	waiting Time
P_1	30	30-0=30	30-24=6
P_2	7	7-0=7	7-3=4
P_3	10	10-0=10	10-3=7

(For Rule 2) waiting time = Last ~~start~~ time - Arrival Time
- (Preemption x Time Quantum)

Procen ID	Waiting Time
P_1	26-0-(5x4) = 6
P_2	4-0x(0x4) = 4
P_3	7-0x(0x4) = 7.