CPU Scheduling

⇒ A major task of an operating system is to manage a collection of processes.

⇒ A system with a single CPU or a multiprocessor system with fewer CPU's than processes has to divide CPU time among different processes or threads that are competing to use it. This process is called CPU scheduling.

⇒ If there is only one processor then there will be only one process in a running state at any given time.

⇒ If there are more processes that need to be run, they will have to wait and there should be some mechanism to select which process should run next.

⇒ Scheduling is a fundamental operating system function. CPU is an important resource. It is very important to develop good scheduling algorithims.

⇒ Some scheduling objectives :-

- Maximize throughput = number of jobs completed per unit time.
- Minimize waiting time = total time a job spends waiting in the various queues for a resource to become available.
- Minimize turnaround time = waiting time + computation time + I/O time

• Minimize response time = time from entity of a command until first output starts to appear.

⇒ There are four conditions under which CPU scheduling may take place. They are:
① When a process switches from running state to waiting state.
② When a process switches from running state to ready state.
③ When a process switches from waiting state to the ready state.
④ When a process terminates.

⇒ Below is a list of some scheduling algorithims:
• First come First Serve.
• Shortest Job first.
• Optimal page replacement.

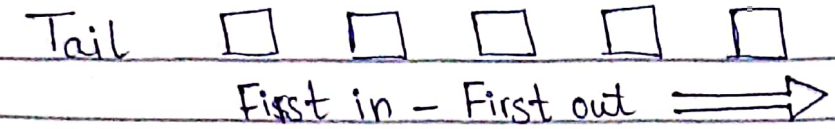
First come First serve :-

⇒ It is the simplest CPU scheduling algorithim.

⇒ It simply works on the principle that the process that requests the CPU first is allocated the CPU first.

⇒ The implementation of FCFS policy is easily managed with a FIFO queue.

⇒ A drawback of the FCFS algorithim is that the processes may have to wait for excessively long amount of time.



⇒ When a process enters a ready queue, its Process Control Block (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.

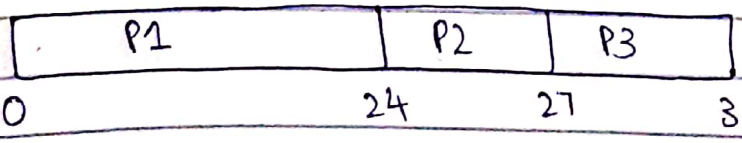
⇒ The running process is then removed from the queue.

Example :-

Consider the following set of processes that arrive at time 0:

Process	Burst Time
P1	24
P2	3
P3	3

If the processes arrive in the order P1, P2 and P3. Then, we get the following Gantt chart:



Waiting time = Finish time - burst time - arrival time

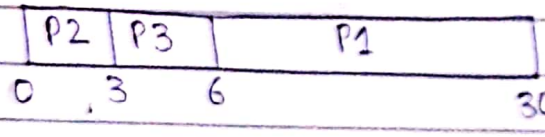
Waiting time for P1 = 0 ms
Waiting time for P2 = 24 ms
Waiting time for P3 = 27 ms

Average waiting time = (0+24+27)/3

= 17 ms

Example :-

If the Processes arrive in the order P2, P3 and P1.



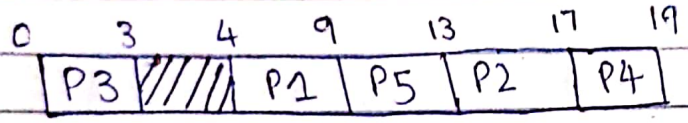
Waiting time for P1 = 6 ms
Waiting time for P2 = 0 ms
Waiting time for P3 = 3 ms

Average waiting time = (6 + 0 + 3)/3
= 3 ms

=> If processes with higher burst time arrived before the processes with smaller burst time, then, smaller processes have to wait for a long time for longer processes to release the CPU. This is called the convoy effect.

Example :-

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



▨ = idle time of CPU

Calculate average Turn Around time and waiting time ?

Turn Around time = Completion time − Arrival time
Waiting time = Turn around time − burst time

Process ID	Completion Time	Turn around time	waiting time
P1	9	5	0
P2	17	11	7
P3	3	3	0
P4	19	13	11
P5	13	8	4

average T. A. time = (5+11+3+13+8) / 5
= 8 units

average waiting time = (0+7+0+11+4) / 5
= 4.4 units

Job Scheduling Algorithim :-

⇒ It is an algorithim that picks the process which takes the least amount of time to complete.

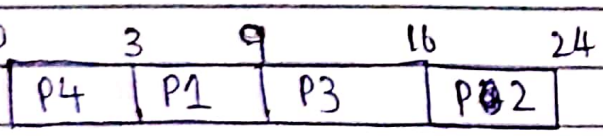
⇒ In FCFS, the average time was reduced by running the short job first. So, the SJF algorithim picks the shortest job in terms of burst size and places it on CPU.

⇒ When CPU is available, it is assigned to the process that has the smallest next CPU burst.

⇒ If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.

Example :-

Process ID	Burst time
P1	6
P2	8
P3	7
P4	3



Waiting time :-

P1 = 3 , P2 = 16 , P3 = 9 , P4 = 0

(3+16+9+0) / 4 = 7 (avg)