Introduction to Process Scheduling [FCFS] Convoy effect

Process Scheduling →

* Basis of Multi-Programming O.S
* By Switching CPU among Processes, OS can make Computer more Productive
* Many Processes are kept in memory at a time, when a Process must wait or time quantum Expires, the O.S takes the CPU away from that Process & give the CPU to another Process & this Pattern Continues.

__/__/____

2) **CPU Scheduler :-**
* Whenever the CPU become ideal, OS must Select one Process from the ready queue to be Executed.
* Done by STS

3) **Non-Preemptive Scheduling :-**
* once CPU has been allocated to a Process, the Process keeps the CPU until it releases CPU either by terminating / by Switching to wait State.
Demerit * Starvation, as a Process with long burst time may Starve less burst time Process.
* Low CPU utilization.

4) **Preemptive Scheduling :-**
* CPU is taken away from a Process after time quantum Expires along with terminating (or) Switching to wait State.
@More overhead
* Less Starvation * High CPU utilization.

5) **Goals of CPU Scheduling :-**
* Max CPU utilization * Minimum Turnaround time (TAT)
* Min wait time * Min response time
* Max throughput of the System.

6) **Throughput** : No of Processes Completed per unit time.
7) **Arrival Time [AT]** : Time when Process is arrived at Ready Q
8) **Burst Time [BT]** : The time required by the Process for its Execution.
9) **Turnaround Time [TAT]** : Time taken from 1st time Process enters ready State till it terminates.
[CT - AT].

__/__/____

10) Wait Time (WT): Time Process Spends waiting for CPU.
[WT = TAT - BT].

11) Response Time: Time duration b/w Process getting into ready Q & Process getting CPU for the 1st time

12) Completion Time: (CT) → Time taken till Process gets terminated.

1 **) FCFS { First Come First Serve }:

* Whichever Process comes 1st in R.Q will get CPU first.
* In this, if one Process has longer BT, it will have major effect on average W.T of diff Processes, Called 'Convoy effect'.

"It is a Situation where many Processes, who needs to use a resource for a short time, are blocked by one Process holding that resource for a long time"

↳ This Causes poor resource Management.

Imp) 1)

Process	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)
P_1	0	20	20	20	0
P_2	1	2	22	21	19
P_3	2	2	24	22	20

→ Avg: 13

⊙ Gantt Chart

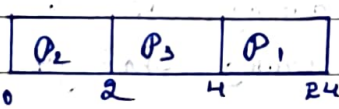
* More avg W.T because P_1 has huge BT.

2)

Process	AT	BT	CT	TAT	WT
P_2	0	2	2	2	0
P_3	1	2	4	3	1
P_1	2	20	24	22	2

→ Avg = 1

⊙ Gantt Chart



* least avg time since P_2 is at last.

__/__/__

2) **Shortest Job First [SJF] :-**

(i) **Non - Preemptive :**

* Process with least B.T will be dispatched to CPU 1st.
* Must do estimation of BT for each Process in R.Q beforehead, Correct estimation of BT is an impossible task (Ideally).
* Process Starvation may happen if 1st P has large BT

3)

P	AT	BT	CT	TAT	WT
P_1 ← It will be 1st	0	8	8	8	0
P_2	1	4	12	11	7
P_3	2	9	26	24	15
P_4	3	5	17	14	9

→ Avg = 7.75

* Gantt :-

P_1	P_2	P_4	P_3

$t=0$ 8 (4) 12 (5) 17 (9) 26

{all A, P, P+ arrived}

Criteria for Selection : AT + BT.

(ii) **Preemptive SJF :-**

4)

P_1	P_2	P_4	P_1	P_3	

$t=0$ 1 (4) 5 (3) 10 (7) 17 (9) 26

(it is getting fully executed since other 3 are greater than this)

AT + BT

P	AT	BT	CT	TAT	WT
→ P_1	0	~~8~~ 1 (4)	17	17	9
P_2	1	4	5	4	0
P_3	2	9	26	24	15
P_4	3	5	10	7	2

→ Avg = 6.5

* No Convey effect → Less Starvation.
* Since even though P_1 is with more B.T ; It is Preempted.
* Gives average W.T less for a given Set of Process as Scheduling Short job before a long one for W.T.

__/__/__

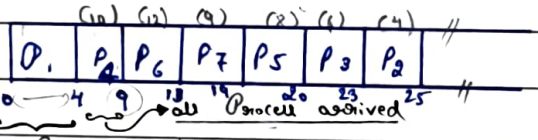
3 **) **Priority Scheduling**: Assign Priority to each Job.

↳ i) **Non-Preemptive.**

* Priority is assigned to a Process when it is Created
* SJF is a Special Case of general Priority Scheduling with Priority inversely Proportional to B.T,

5)

P	Priority	AT	BT	CT	TAT	WT	(1 → high Priority)
P_1	2	0	4	4	4	0	
P_2	4	1	2	25	24	22	
P_3	6	2	3	23	22	19	
P_4	10	3	5	9	6	1	
P_5	8	4	1	20	16	15	
P_6	12	5	4	13	8	4	
P_7	9	6	6	19	13	7	→ 9.714



(for Next Process, Since t=4; All Process with A.T 0-4 Come into action) → Select high Priority among them.

↳ ii) **Preemptive.**

* Current RUN State job will be Preempted if next job has higher Priority. (Pre + Non Pre)

Drawback: * Huge Convoy effect.

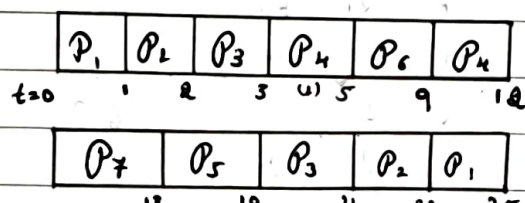
↳ * Indefinite Waiting (or) Extreme Starvation.

↳ Sol^n: - Ageing → We gradually Inc the Priority of Lowest Jobs

↳ Eg: Every 15 min → lowest Priority Jobs → Priority + 1 Incd.

6)

	Priority	A.T	B.T	C.T
1	2	0	~~4~~ 3	25
2	4	1	~~2~~ 1	22
3	6	2	~~3~~ 2	21
4	10	3	~~5~~ 3	12
5	8	4	1	19
6	12	5	4	9
7	9	6	6	18



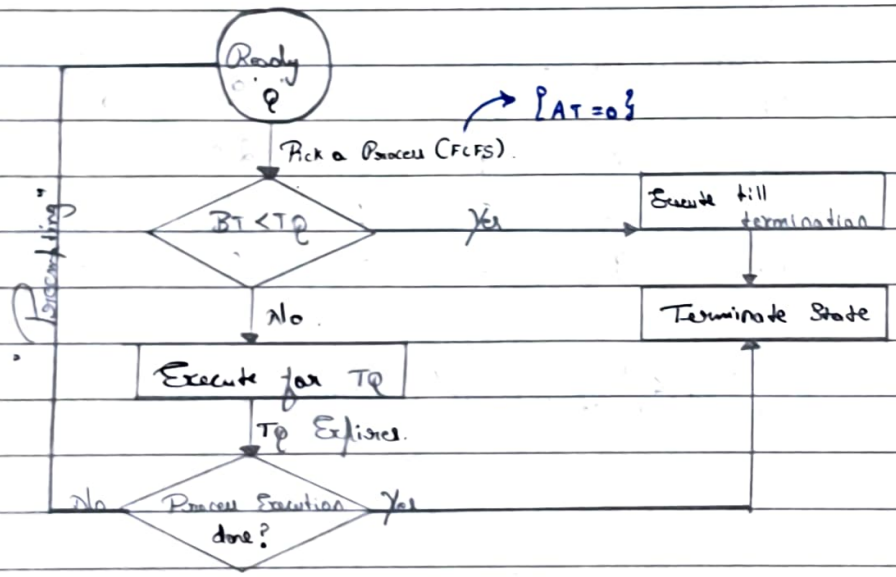
Avg = 11.4

* (Convoy) → IBM 7094 at MIT → 1967 → 1973 Convoy Jobs of lower Priority didn't get CPU yet).

__/__/____

4 ⋆⋆) Round Robin Scheduling (RR).

* Most Popular * Like FCFS but Preemptive * Easy to implement.
* Designed for Time Sharing Systems
* Criteria: AT + time Quantum (TQ), doesn't depend on BT.
* No Process is going to Wait forever, Hence very low Starvation 'No Convey Effect'.
* If TQ is Small, more will be Context Switch (more overhead).



⋆) It is a CPU Scheduling algorithum where each Process is Cyclically assigned a fixed time Slot.

⋆) The Period of time for which a Process or job is allowed to run in a Pre-emptive method is Called "Time Quantum".

⋆)

P	AT	BT
1	0	4 2 end
2	1	5 3 1 end
3	2	2 0 end
4	3	1 0 end
5	4	6 4 2 end
6	5	3 1 end

TQ = 2 ⋆

P_1	P_2	P_3	P_1	P_4	P_5	P_2	P_6

t = 0 2 4 6 8 9 11 13 15

P_5	P_2	P_6	P_5

17 18 19 (total terminated 21)

Queue :

P_1	P_2	P_3	P_1	P_4	P_5	P_2
P_6	P_6	P_5	P_6	P_5		

Real-time Implementation

__/__/____

Multi-Level Queue Scheduling [MLQ]:

* Ready Q is divided into multiple queues depending upon Priority.
* A Process is Permanently assigned to one of the queues (inflexible) based on Some Property of Process, memory, Size, Process Priority (or) Process type.

Highest Priority

→ System Process (SP) Queue → * Scheduling algorithum = RR
  * Created by O.S * Eg → init, network related.

→ Interactive Process (IP) Q → * SA = RR * foreground Process.
  * Needs user I/P (I/O)
  → CLI & GUI Command linteractive lines.,

Least Priority

→ Batch Process (BP) Q → * SA = FCFS * Background Process
  * Runs Silently, No user i/p required.
  → ,) Compilation of Programs.,

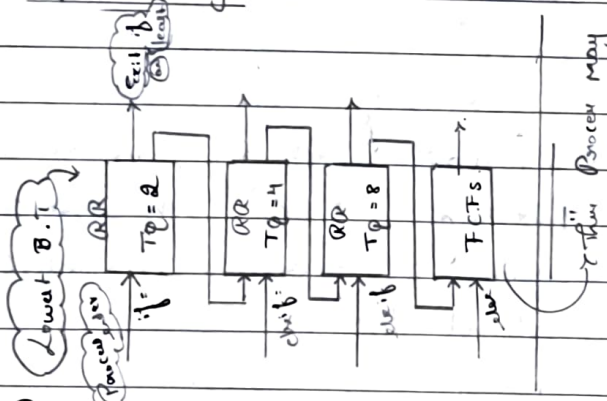
* Scheduling among different Sub-queues is implemented as "fixed Priority Preemptive Scheduling" Eg → F.P high Priority than B.P
* If F.P Comes & B.P is Currently Executing → then B.P will be Preempted.
* Problem: only after Completion of all the Processes from the top-level ready Q, the further level Ready Q will be Scheduled.
* This reesults in Starvation of lower Priority Process.
* Convey effect is Present.

__/__/____

✱) Multi-level feedback queue Scheduling (MLFQ) :-

* Multiple Sub-queue are present (Inter q Movement allowed),
* Allows the Process to move b/w queues. The idea is to Seperate Processes according to the Characteristics of their BT. If a Process uses too much CPU time, it will be moved to lower Priority Queue. This Scheme leaves I/O bound and interactive Processes in the higher-Priority q.
* In addition, a Process that waits much in lower-priority queue may be moved to a higher Priority q. This form of 'ageing' Prevents Starvation.
* Less Starvation than MLQ. * It is flexible
* Can be Configured to match a Specific System design requirement

✱) Sample Design :



Design
i) No of q
ii) Scheduling algorithm in each q
iii) Method to upgrade process to higher q
iv) Demote a Process to Lower Priority

✱) Comparision :

	FCFS	SJF	PSJF	Priority	PPriority	RR	MLQ	MLFQ
i) Design	Simple	Complex	Complex	Complex	Complex	Simple	Complex	Complex
ii) Preemption	No	No	Yes	No	Yes	Yes	Yes	Yes
iii) Convey Effect	Yes	Yes	No	Yes	Yes	No	Yes	Yes
iv) overload	No	No	Yes	No	Yes	Yes	Yes	Yes