Non – Preemptive Scheduling

Points To Remember…

- CPU is allocated to the process *till it terminates or switches to waiting state*
- *Execution process is not interrupted* even if higher priority ones arrive
- *No overhead* of switching the process from running to ready state
- Often termed as *“rigid”*

- ✓ Two processes with same priority are assigned on basis of “arrival time” [first – come – first – serve]
- ✓ Waiting time = Response time
- ✓ Last time instant of gantt chart equals the sum of all the burst time

Key Points

An Example :-

Lesser the number… Higher the priority

	Priority	Arrival Time (A.T)	Burst Time (B.T)	Completion Time (C.T)	Turn-around Time (T.T = C.T – A.T)	Waiting Time (W.T = T.T – B.T)	Response Time (R.T)
P1	3	0	8	**8**	8-0 = **8**	8 – 8 = **0**	**0**
P2	4	1	2	**17**	17 – 1 = **16**	16 – 2 = **14**	**14**
P3	4	3	4	**21**	21 – 3 = **18**	18 – 4 = **14**	**14**
P4	5	4	1	**22**	22 – 4 = **18**	18 – 1 = **17**	**17**
P5	2	5	6	**14**	14 – 5 = **9**	9 – 6 = **3**	**3**
P6	6	6	5	**27**	27 – 6 = **21**	21 – 5 = **16**	**16**
P7	1	10	1	**15**	15 – 10 = **5**	5 – 1 = **4**	**4**

Sum : 27

-: Gantt Chart :-

Left to Right : time at which CPU is allocated to the process

P1	P5	P7	P2	P3	P4	P6

0 8 14 15 17 21 22 27

Completion Time (C.T)

Advantages

- Easy to use
- High priority doesn't need to wait for long
- Good mechanism, relative importance of each process is precisely defined

Disadvantages

- If a **new high priority process keeps on coming** in the ready queue, then **the process which is in the waiting state may need to wait for long duration** of time [***preemptive scheduling***]
- **Current process is not interrupted**, and hence **a higher priority job may have to wait** [***non-preemptive scheduling***]

Starvation

(Indefinite Blocking)

We have solution :)

AGING

a technique of **gradually increasing the priority of processes that wait** in the system **for a long time**.