* Scheduling (CPU scheduling).

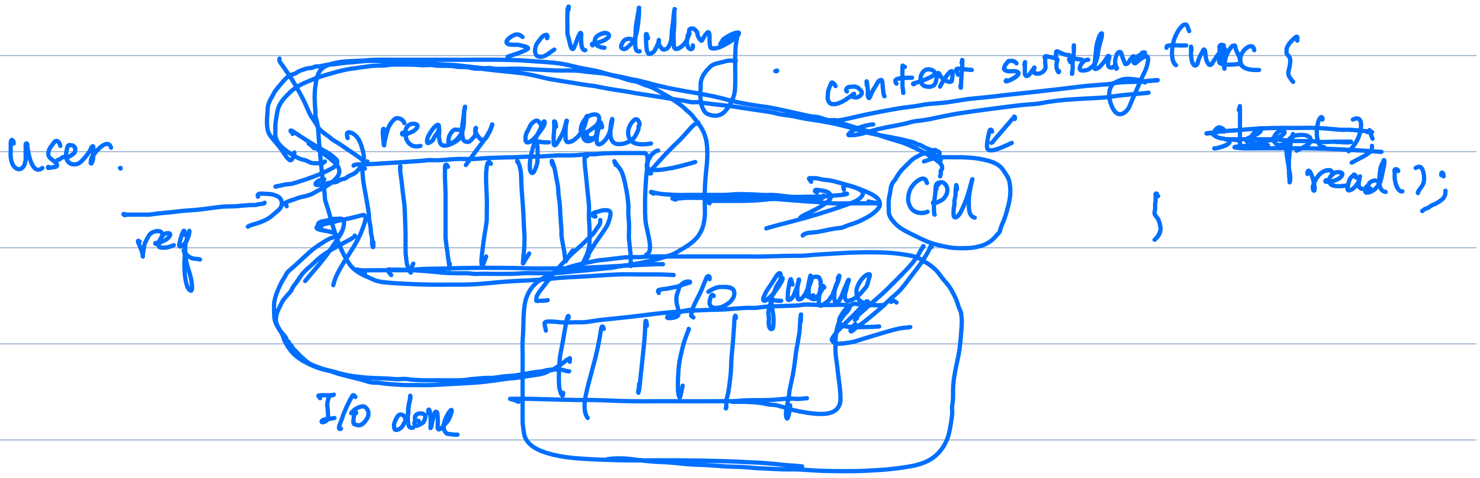
⇒ picking a new thread to run.

first create a thread. launch it. and then the thread gets executed.

→ struct Thread instance.

queue. : stores all thread instances.

pick one thread out of the queue & run it.



(Preemptive : timers. ← frequency.
quantum

Non-preemptive :

* FCFS (First Come. First Served)

Non-preemptive.

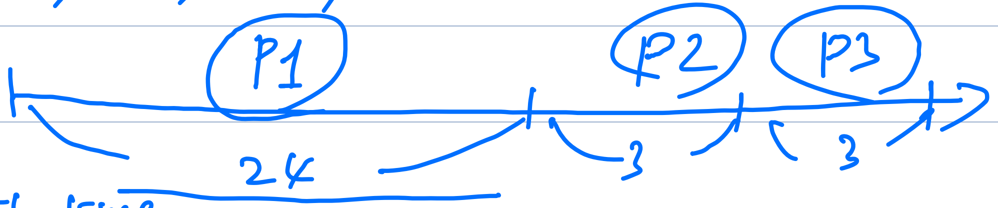
Metrics for comparison

* Metrics for comparison.

- Wait time in the queue.
- Throughput : # of threads that can complete the execution.
- Priority : Important threads first.
- scheduler latency
- CPU usage
- Turnaround time (creation to termination)
- Response time (creation to first response)

* FCFS example

P1. P2. P3
(24) (3) (3)



*Wait time

P1 : 0   P2 : 24.   P3 : 27.

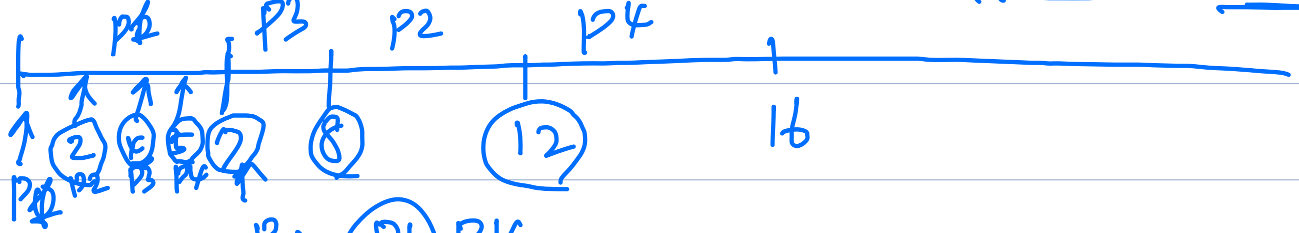
$$\frac{0 + 24 + 27}{3} = 17$$

* Shortest Job First

Big assumption

Execution time : P1 : 7.   P2 : 4.   P3 : 1 .   P4 : 4

Arrival time : P1 : 0.   P2 : 2.   P3 : 4.   P4 : 5



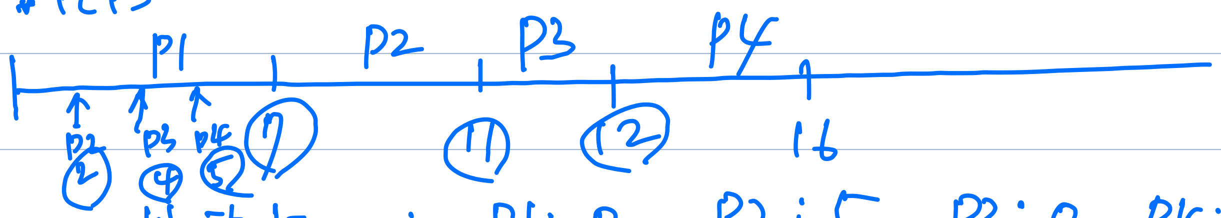
P2. P3. P4

* Wait time

P1 : 0 . P2 : 6 . P3 : 3 . P4 : 7.

$$\frac{0+6+3+7}{4} = 4.$$

* FCFS



Wait time : P1: 0 . P2: 5 . P3: 7. P4: 7.

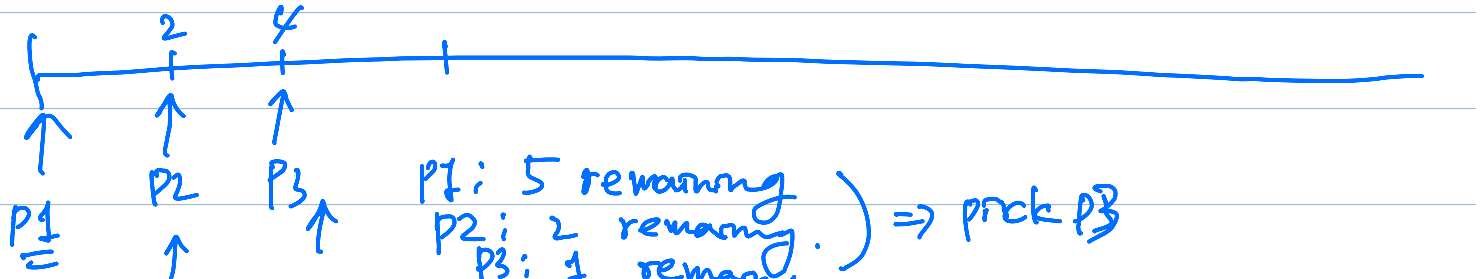
$$\frac{0+5+7+7}{4} = 19/4.$$

Starvation problem

* Shortest Remaining Time First

Exe: P1: 7. P2: 4. P3: 1. P4: 4

Arr: P1: 0. P2: 2. P3: 4. P4: 5



P1: 5 remaining
P2: 2 remaining
P3: 1 remaining) => pick P3

P1 : 5 remaining
P2: 4 remaining) => pick P2

starvation.

execution time assumption.

* Multiple queues. for priority.

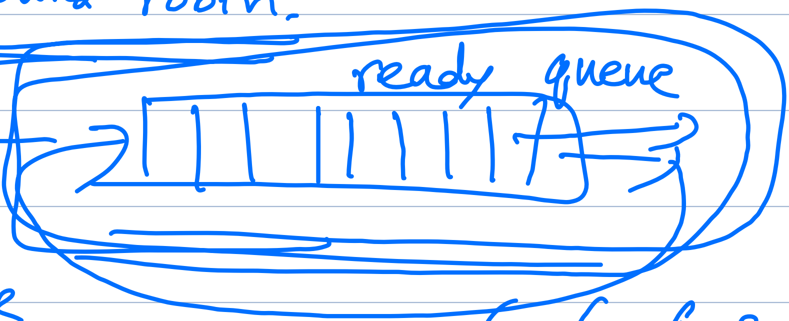
=> Multilevel queue scheduling.

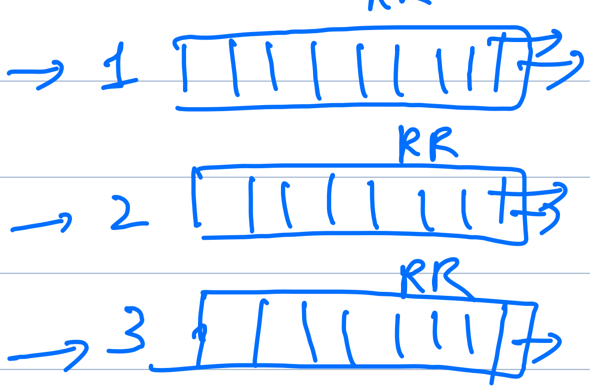
run as often as possible

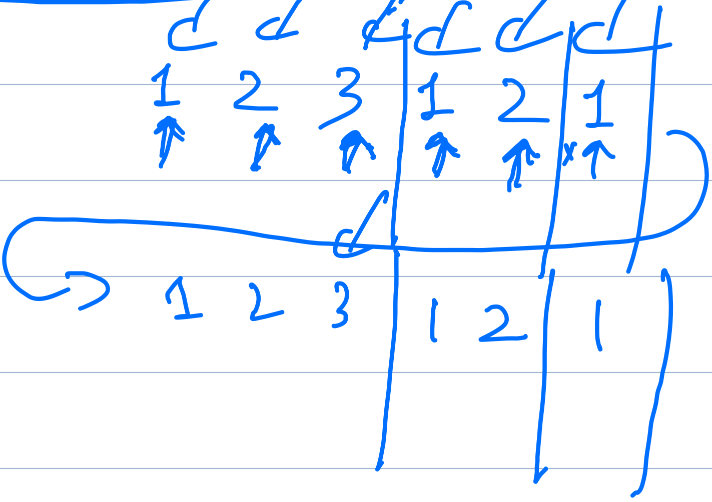
=> System processes   high priority queue foreground processes medium priority queue

background processes low priority queue

Weighted round robin.


ready queue




1 2 3 1 2 1
1 2 3 1 2 1