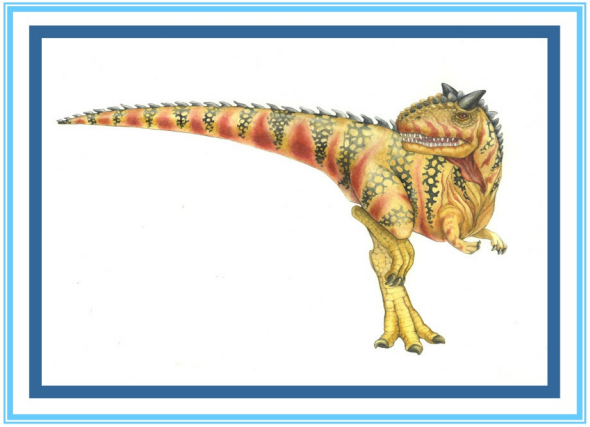
Chapter 6: CPU Scheduling



Chapter 6: CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Examples

Objectives

- To introduce CPU scheduling, which is the basis for multiprogrammed operating systems
- To describe various CPU-scheduling algorithms
- To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system

Basic Concepts

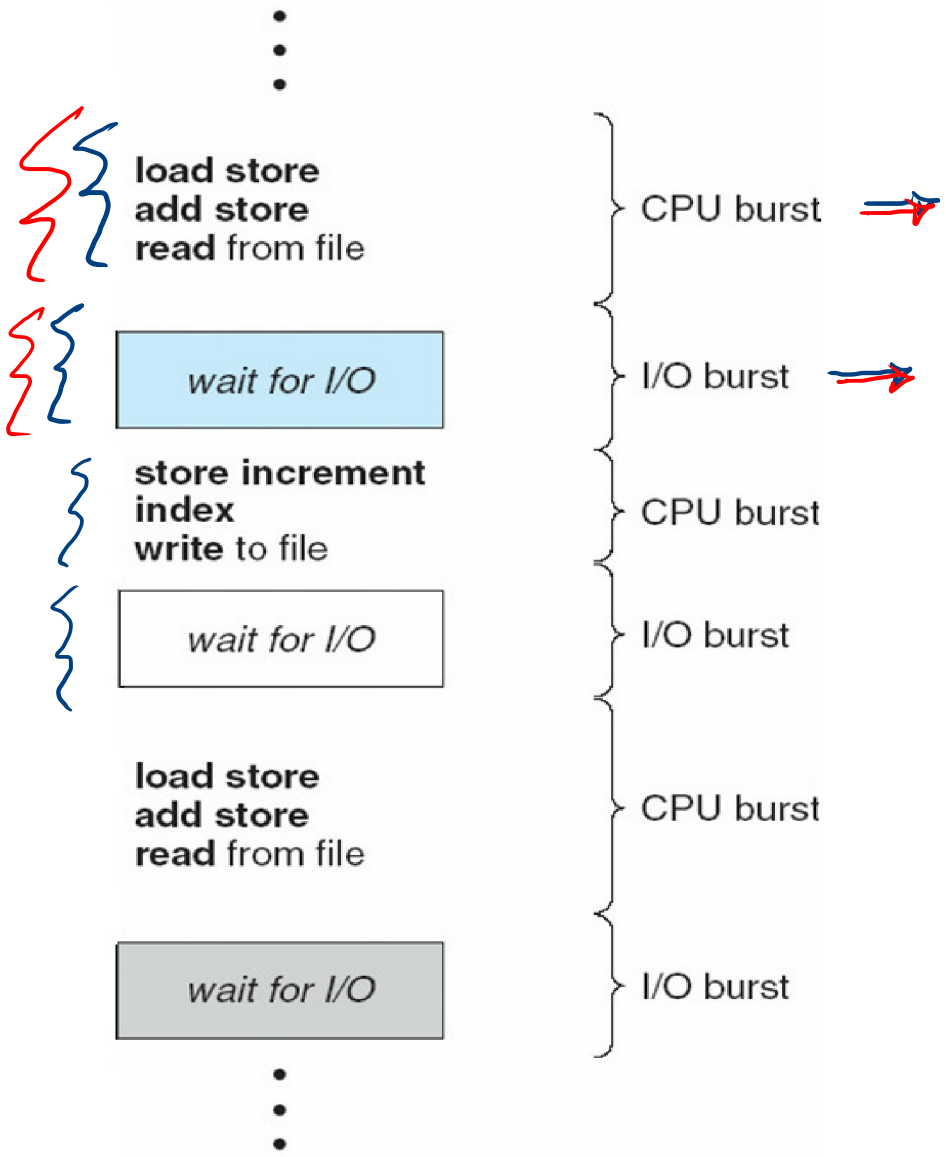
- Maximum CPU utilization obtained with multiprogramming (MPg)
- Continuous Cycle :
  - one process has to wait (I/O)
  - Operating system takes the CPU away
  - Give CPU to another process
    - This pattern continues
- CPU–I/O Burst Cycle – Process execution consists of a *cycle* of CPU execution and I/O wait

CPU and I/O Burst Cycle

- Almost all processes alternate between two states in a continuing ***cycle***, as shown in Figure below ***:***
  - A CPU burst of performing calculations, and
  - An I/O burst, waiting for data transfer in or out of the system.

- Processes alternate back and forth between this two states.

Alternating Sequence of CPU and I/O Bursts

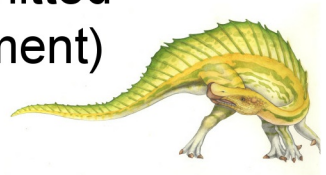


CPU Scheduler

- Selects from among the processes in ready queue, and allocates the CPU to one of them
  - FIFO queue
  - Priority queue
  - Tree
  - Unordered linked-list
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state (I/O request)
  2. Switches from running to ready state (e.g. when interrupt occurs)
  3. Switches from waiting to ready (e.g. at completion of I/O)
  4. Terminates
- Scheduling under 1 and 4 is **nonpreemptive**
- All other scheduling is **preemptive**
  - Consider access to shared data
  - Consider preemption while in kernel mode
  - Consider interrupts occurring during crucial OS activities

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time**
  – amount of time to execute a particular process $(T_c - T_s)$
  -- the interval from the time of submission of a process to the time of the completion.
  -- sum of the periods spent waiting to get into memory, wainting in the ready queue, executing on the CPU, doing I/O
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)



Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

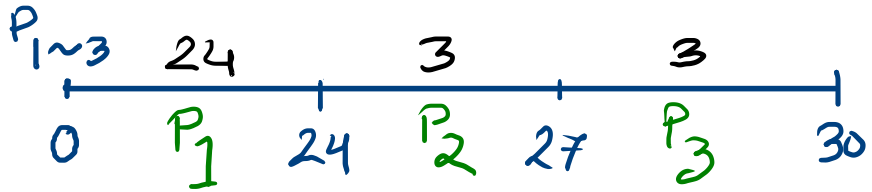
- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:

- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Section - 1

P_1	24
P_2	3
P_3	3

$P_1 \rightarrow P_2 \rightarrow P_3$



(*) Turnaround time

P_1 : 24 − 0 = 24
P_2 : 27 − 0 = 27
P_3 : 30 − 0 = 30

} Avg = 21

(*) Waiting time

P_1 : 0 − 0 = 0
P_2 : 24 − 0 = 24
P_3 : 27 − 0 = 27

} Avg = (17)

(*) Response time

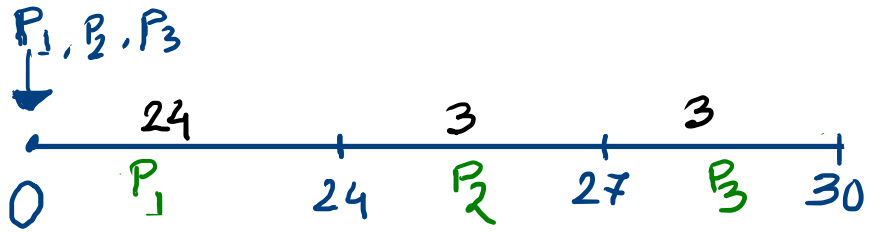
P_1 : 0 − 0 = 0
P_2 : 24 − 0 = 24
P_3 : 27 − 0 = 27

} Avg = (17)

$(C_t - A_t)$ + ☐

Section - 02

	B_T
P_1	24
P_2	3
P_3	3



(*) $P_1 \rightarrow P_2 \rightarrow P_3$

(*) Turnaround times

P_1 : 24 − 0 = 24
P_2 : 27 − 0 = 27
P_3 : 30 − 0 = 30

Avg → T_A = (27)

→ (*) Waiting times.

P_1 : 0
P_2 : 24
P_3 : 27

Avg → W_t = (17)

(*) Response times

P_1 : 0
P_2 : 24
P_3 : 27

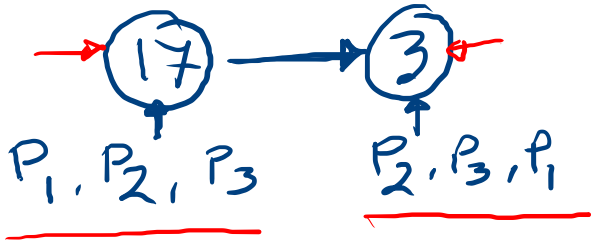
Avg → R_t = (17)

(*) Non-preemptive
W_t, R_t converge

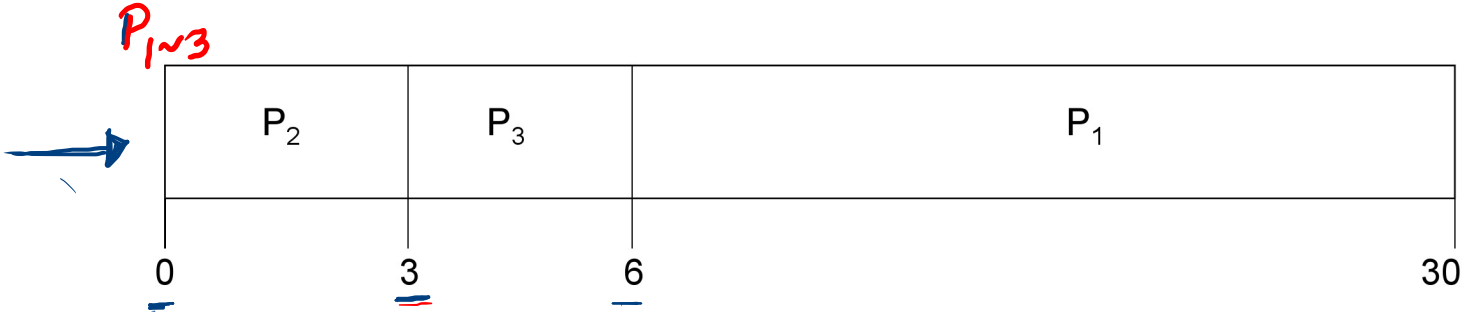
FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order:

P_2 , P_3 , P_1



- The Gantt chart for the schedule is:



Waiting time for $P_1 = 6$; $P_2 = 0$, $P_3 = 3$

Average waiting time: $(6 + 0 + 3)/3 = 3$

Much better than previous case

Convoy effect - short process behind long process

Consider one CPU-bound and many I/O-bound processes

FCFS Scheduling (Cont.)

Convoy Effect :

many I/O bound process and one CPU bound process

CPU bound process	I/O bound process	Effect
I/O device	I/O queue	CPU site idle
CPU processing	Ready queue	I/O site idle

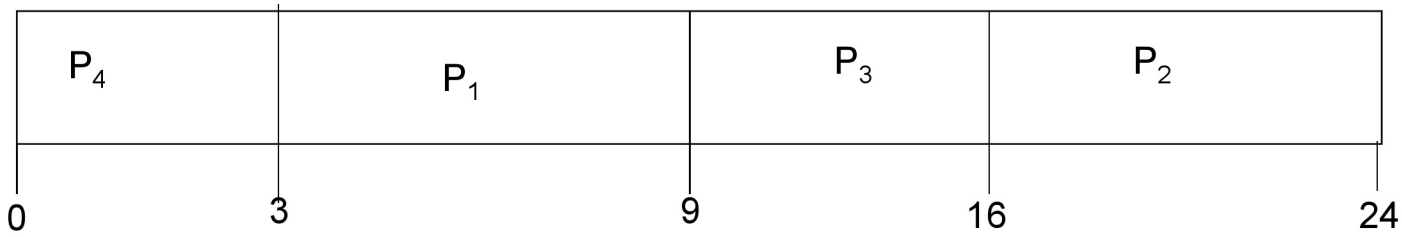
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- Two schemes:
  - **Non-preemptive** – once CPU is given to the process it cannot be preempted until it completes its CPU burst
  - **preemptive** – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF)
- SJF is optimal – gives minimum average waiting time for a given set of processes

Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

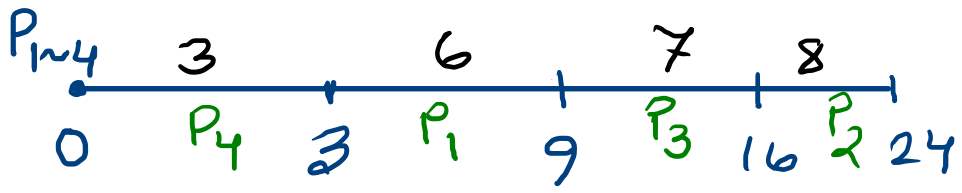
- SJF scheduling chart



Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

Section - 1

P_1	6
P_2	8
P_3	7
P_4	3



$P_4 \rightarrow P_1 \rightarrow P_3 \rightarrow P_2$ ✓ ✓ ✓ ✓

SJF = Sort + FCFS

Waiting times.

P_1: $3 - 0 = 3$

P_2: $16 - 0 = 16$

P_3: $9 - 0 = 9$

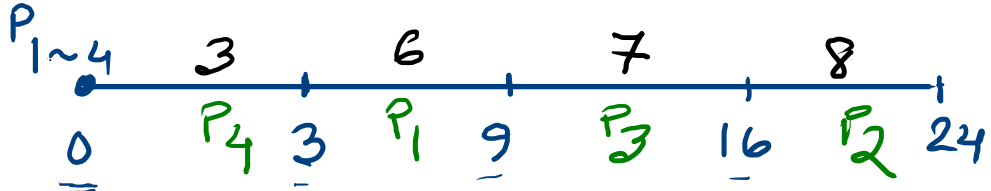
P_4: $0 - 0 = 0$

Avg: (7)

P_1	6
P_2	8
P_3	7
P_4	3

SJF = Sort followed by FCFS

Section-02



→ $P_4 - P_1 - P_3 - P_2$

(*) Waiting Times ⟶ $(C_t - A_t) + \square$ $\quad (C_{Lt} - A_t)$

$P_1 \rightarrow 3 - 0 = 3$

$P_2 \rightarrow 16 - 0 = 16$

$P_3 \rightarrow 9 - 0 = 9$

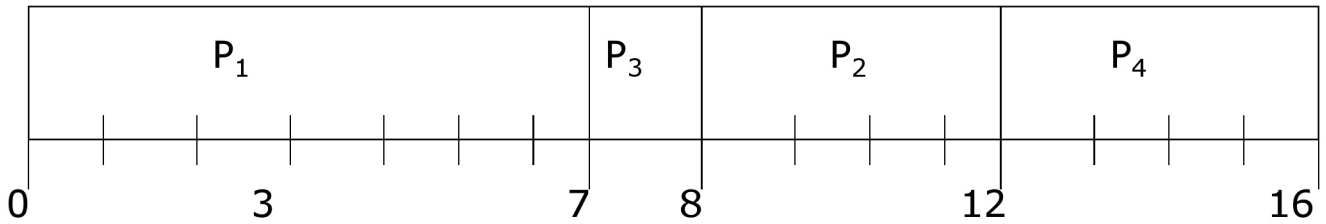
$P_4 \rightarrow 0 - 0 = 0$

Avg. = (7) →

Example of Non-Preemptive SJF

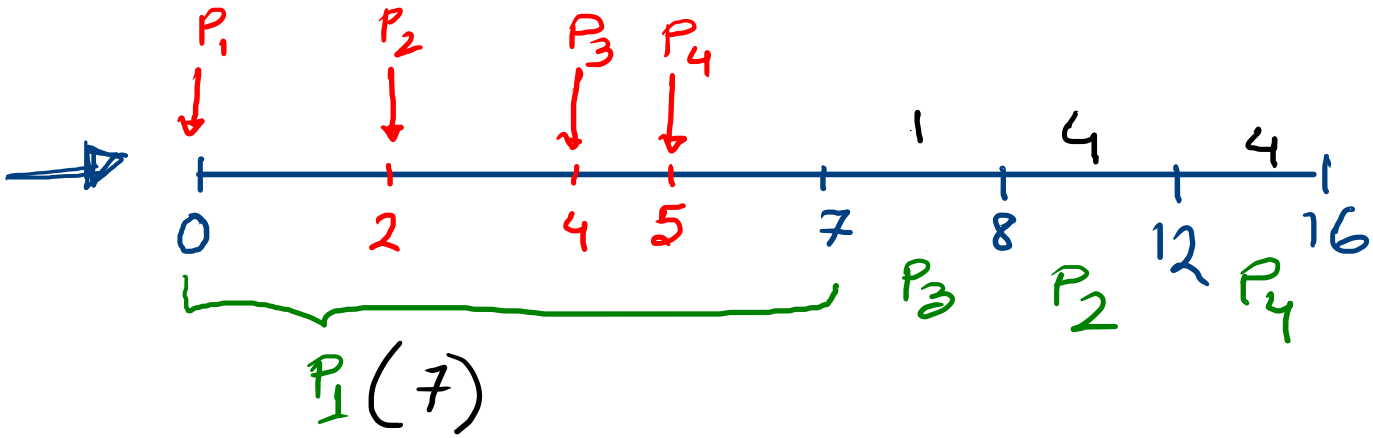
Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)



Average waiting time = (0 + 6 + 3 + 7)/4 = 4

	A	B
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4



$P_3 \rightarrow P_2 \rightarrow P_4$

Waiting times.

$P_1: 0 - 0 = 0$

$P_2: 8 - 2 = 6$

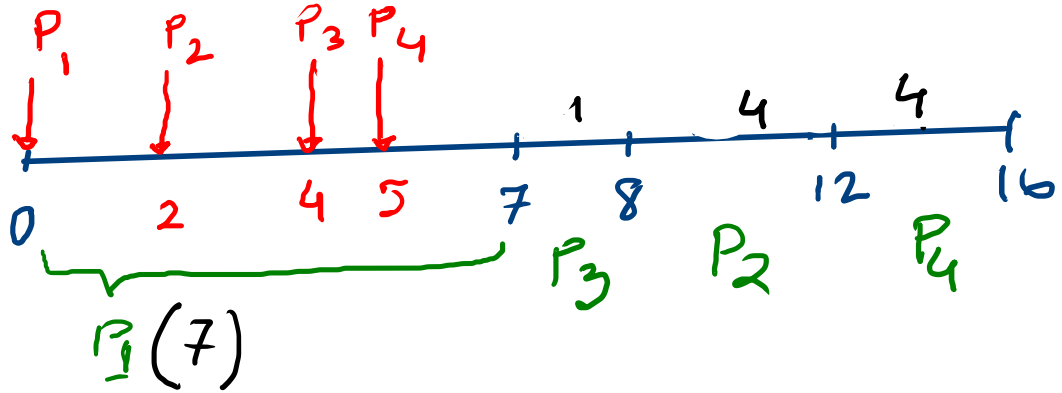
$P_3: 7 - 4 = 3$

$P_4: 12 - 5 = 7$

Avg: $16/4 = 4$

	A	B
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

$P_3 \rightarrow P_2 \rightarrow P_4$



Waiting times.

P_1 : $0 - 0 = 0$

P_2 : $8 - 2 = 6$

P_3 : $7 - 4 = 3$

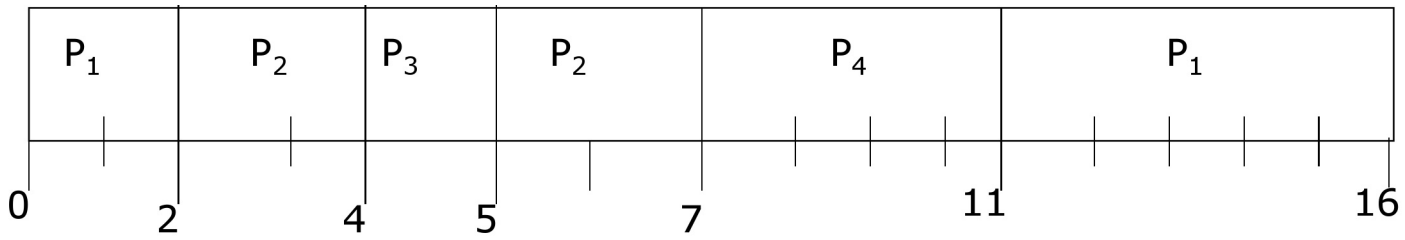
P_4 : $12 - 5 = 7$

Avg = 4

Example of Preemptive SJF

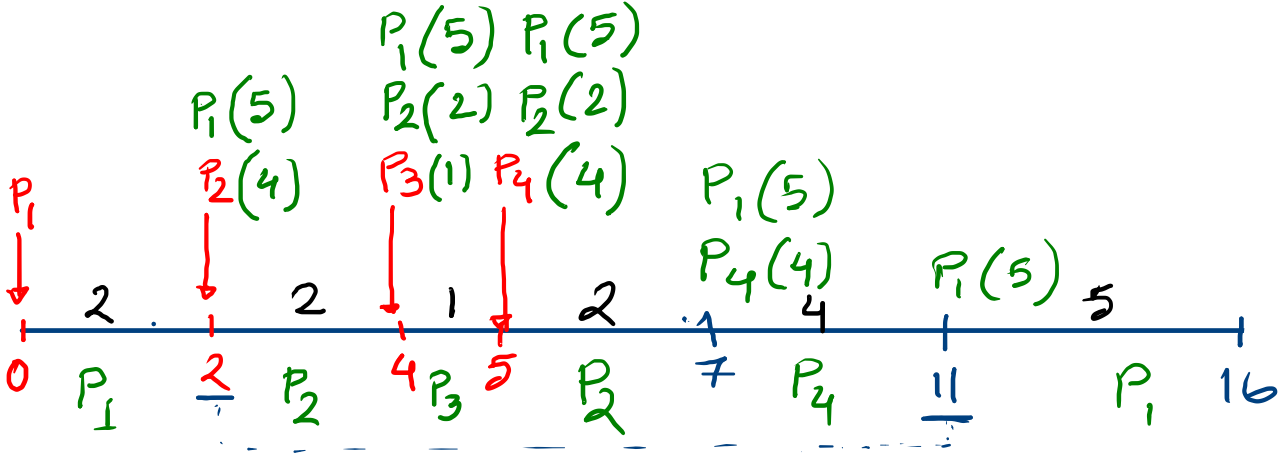
Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

	A	B
P_1	0	~~7~~ ~~3~~ 0
P_2	2	~~4~~ ~~2~~ 0
P_3	4	~~1~~ 0
P_4	5	~~4~~ 0



Waiting times.

P_1: $(0-0)+(11-2)=9$

P_2: $(2-2)+(5-4)=1$

P_3: $(4-4)=0$

P_4: $(7-5)=2$

Avg: $12/4 = 3$

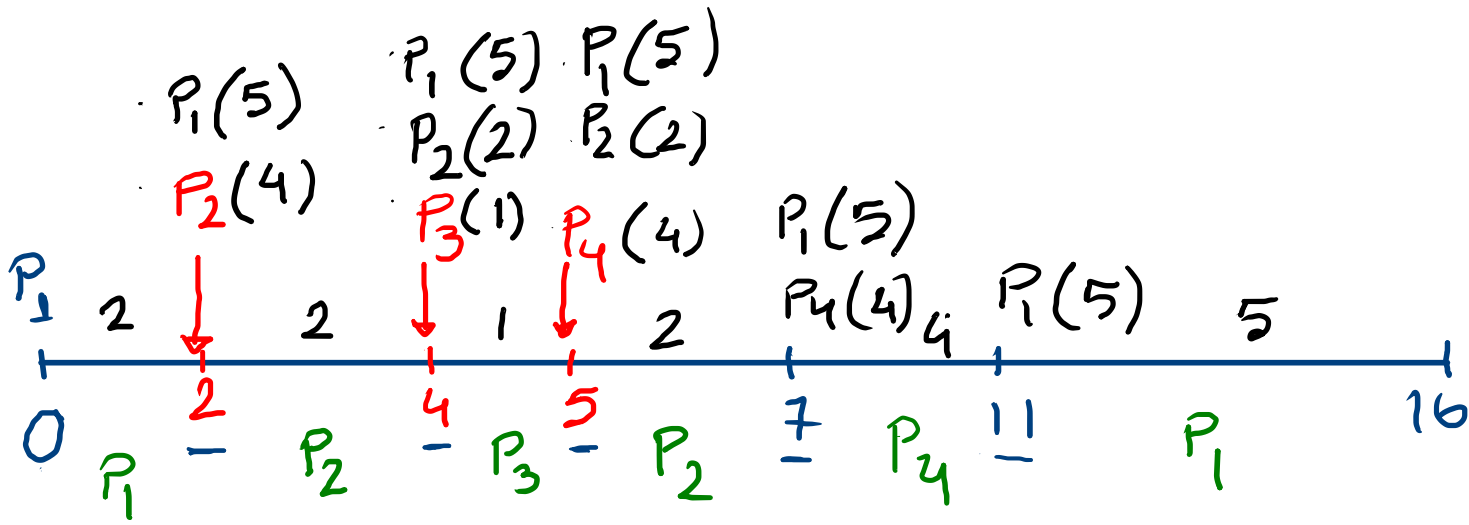
$$w_t = (C_t - A_t) + \sum(CPU_A - CPU_{LR})$$

$$R_t = (C_t - A_t)$$

$$\text{Avg } w_t = \frac{1}{n}\sum\left[(C_{t_i} - A_{t_i}) + \sum(CPU_{A_i} - CPU_{LR_i})\right]$$

$$\text{Avg } R_t = \frac{1}{n}\sum(C_{t_i} - A_{t_i})$$

	A	B
✓ P_1	0	~~7~~ ~~5~~ 0
✓ P_2	2	~~4~~ ~~2~~ 0
✓ P_3	4	~~1~~ 0
✓ P_4	5	~~4~~ 0



Waiting time:

$P_1 = (0-0) + (11-2) = 9$

$P_2 = (2-2) + (5-4) = 1$

$P_3 = (4-4) = 0$

$P_4 = (7-5) = 2$

Avg: $12/4 = 3$

* $W_t = (C_t - A_t) + \sum (CPU_A - CPU_{LR})$
* $R_t = (C_t - A_t)$

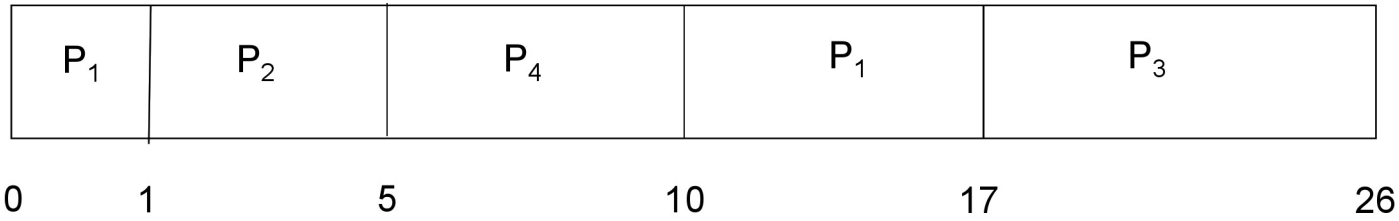
* Avg $W_t = \frac{1}{n} \sum [(C_{t_i} - A_{t_i}) + \sum (CPU_{A_i} - CPU_{LR_i})]$
* Avg $R_t = \frac{1}{n} \sum (C_{t_i} - A_{t_i})$

Example of Shortest-remaining-time-first

- Now we add the concepts of varying arrival times and preemption to the analysis

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

- *Preemptive* SJF Gantt Chart



- Average waiting time = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

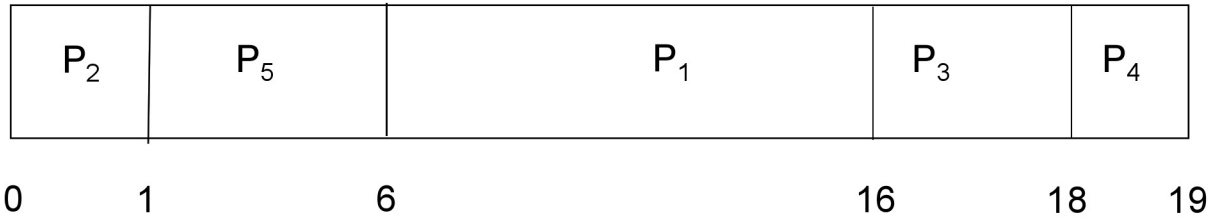
Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
    - Will preempt the CPU if the priority of the newly arrived process is higher than that of the currently running process.
  - Nonpreemptive
    - Will simply put the new process at the head of the ready queue.
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Priority can be defined either internally or externally.
  - Factors for internal priority assignment:
    - Time limit, memory requirements, the number or open files etc.
  - Factors for external priority assignment:
    - Importance of the process, the type and amount of funds of funds being paid for computer use, department sponsoring works etc.

Example of Priority Scheduling

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority scheduling Gantt Chart



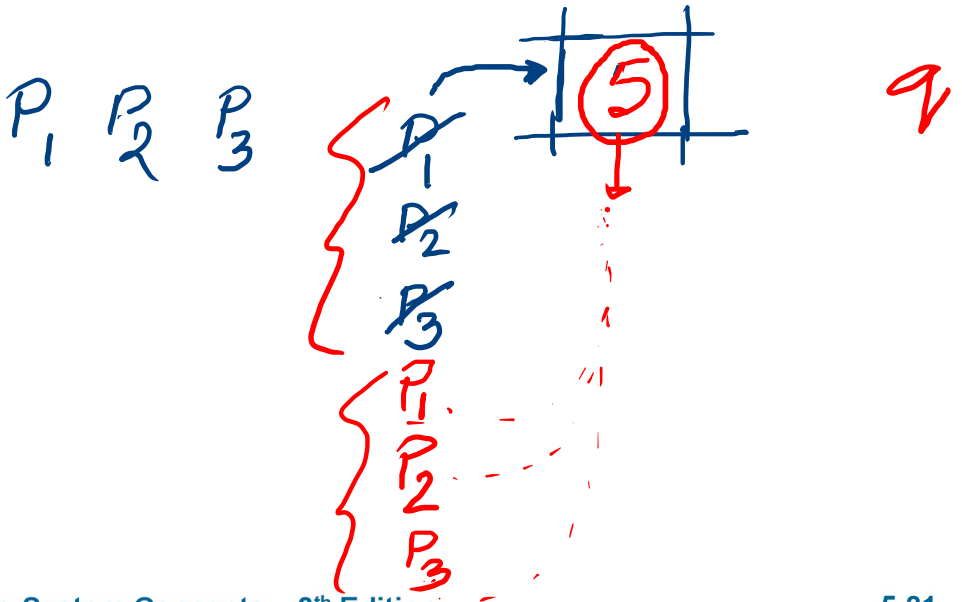
Average waiting time = 8.2 msec

Priority Scheduling

- Problem ≡ **Starvation** – low priority processes may never execute
- Solution ≡ **Aging** – as time progresses increase the priority of the process

Round Robin (RR)

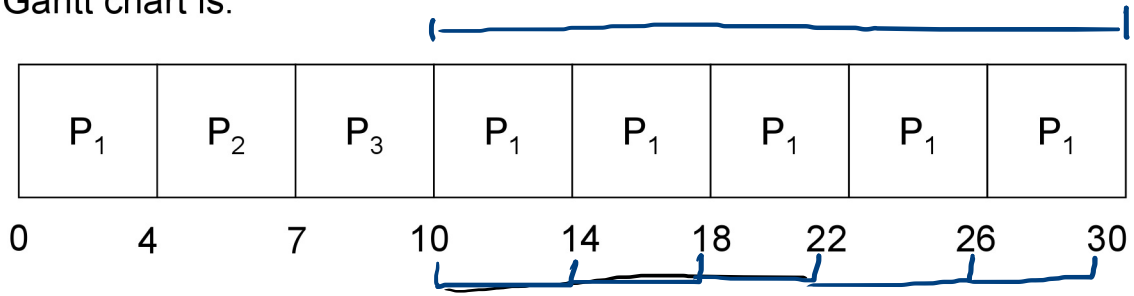
- Each process gets a small unit of CPU time (**time quantum** q), usually 10-100 milliseconds.
- After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then
  - Each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once.
  - No process waits more than [(*n*-1) x q] time units.
- Timer interrupts every quantum to schedule next process
- Performance
  - *q* large ⇒ FIFO
  - *q* small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high



Example of RR with Time Quantum = 4

Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



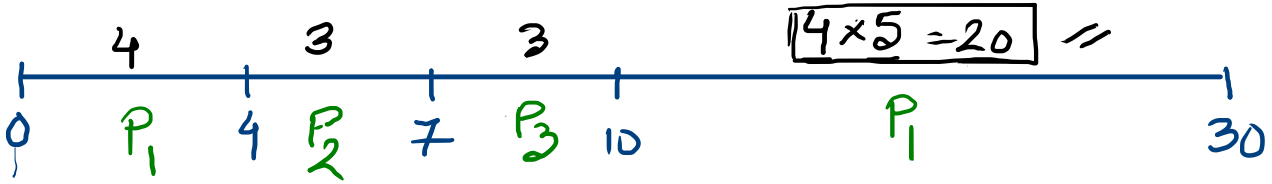
- Average waiting time is 17 / 3 = 5.66 milisecond
- Typically, higher average turnaround than SJF, but better *response*
- quantum should be large compared to context switch time
- Quantum usually 10ms to 100ms, context switch < 10 usec

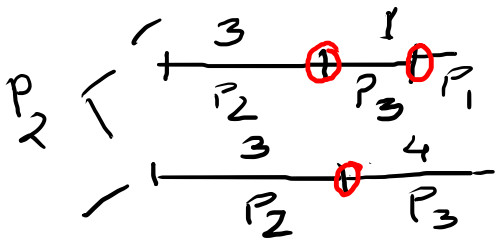
	B	
P_1	~~24~~	20
P_2	~~5~~	0
P_3	~~3~~	0

23

$(4 \times 5) + 3 = 23$

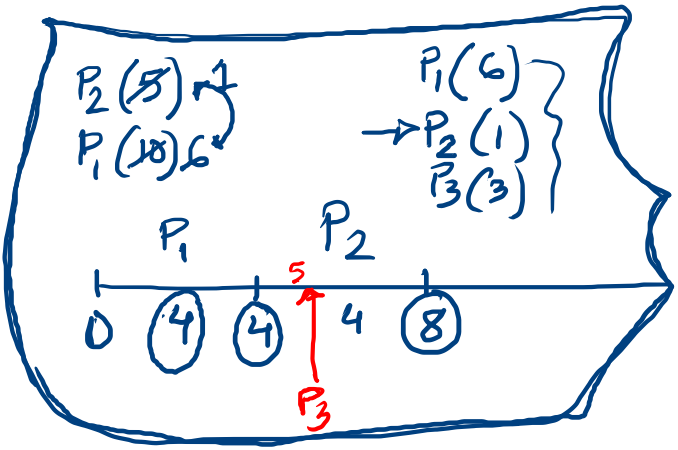
$14 \times 5 = 20$ //





$q = 4$

$P_1 \cdot P_2 \cdot P_3$



Waiting times.

P_1: $(0-0) + (10-4) = 6$

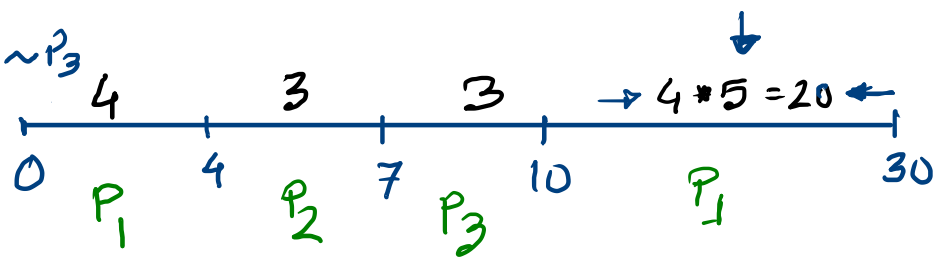
P_2: $(4-0) = 4$

P_3: $(7-0) = 7$

Avg: $\frac{17}{3} = 5.66$

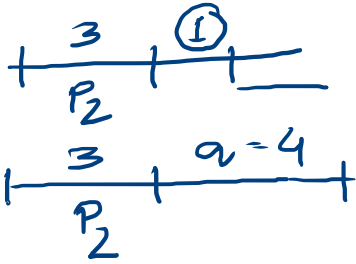
	B	
P_1	~~24~~	20
P_2	~~3~~	0
P_3	~~3~~	0

$P_1 \sim P_3$



$P_2(3)$

$q = 4$



$q = 4$

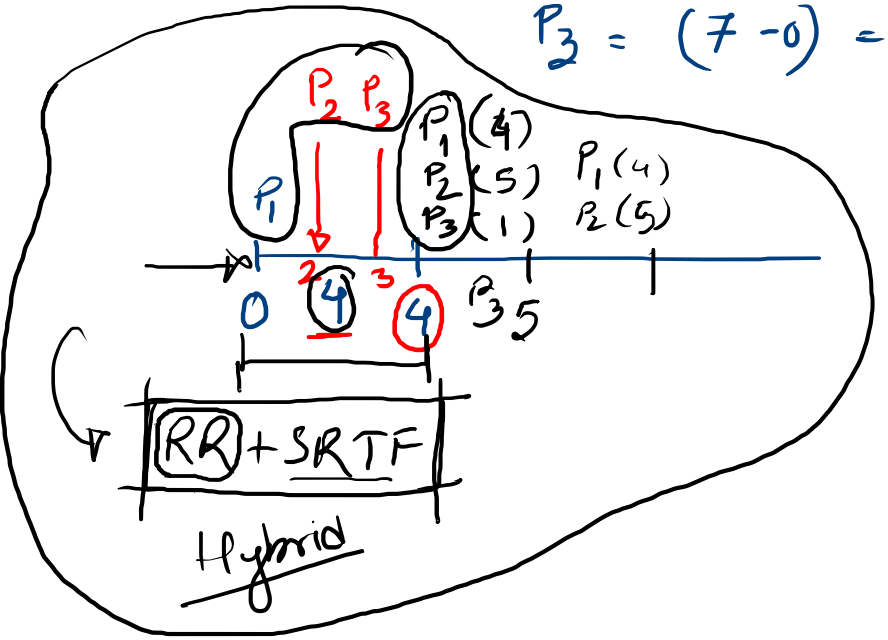
Waiting time

$P_1 = (0-0) + (10-4) = 6$

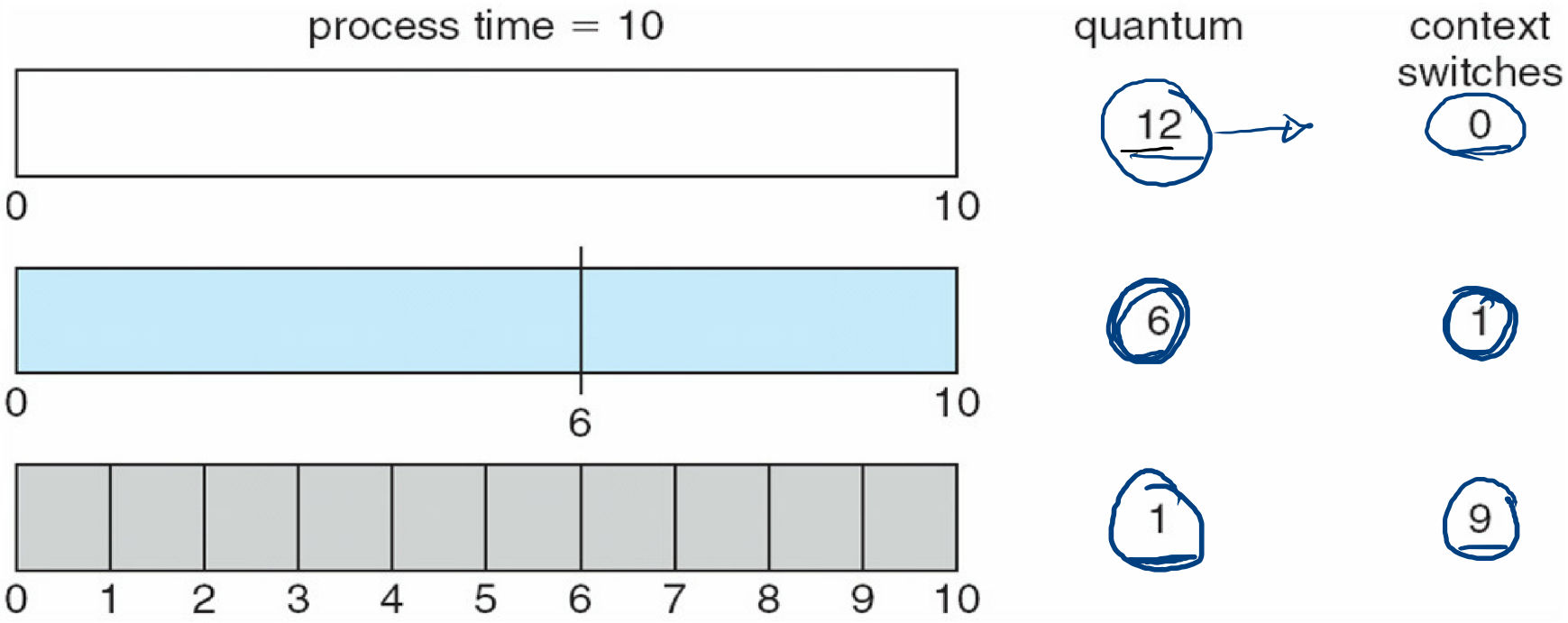
$P_2 = (4-0) = 4$

$P_3 = (7-0) = 7$

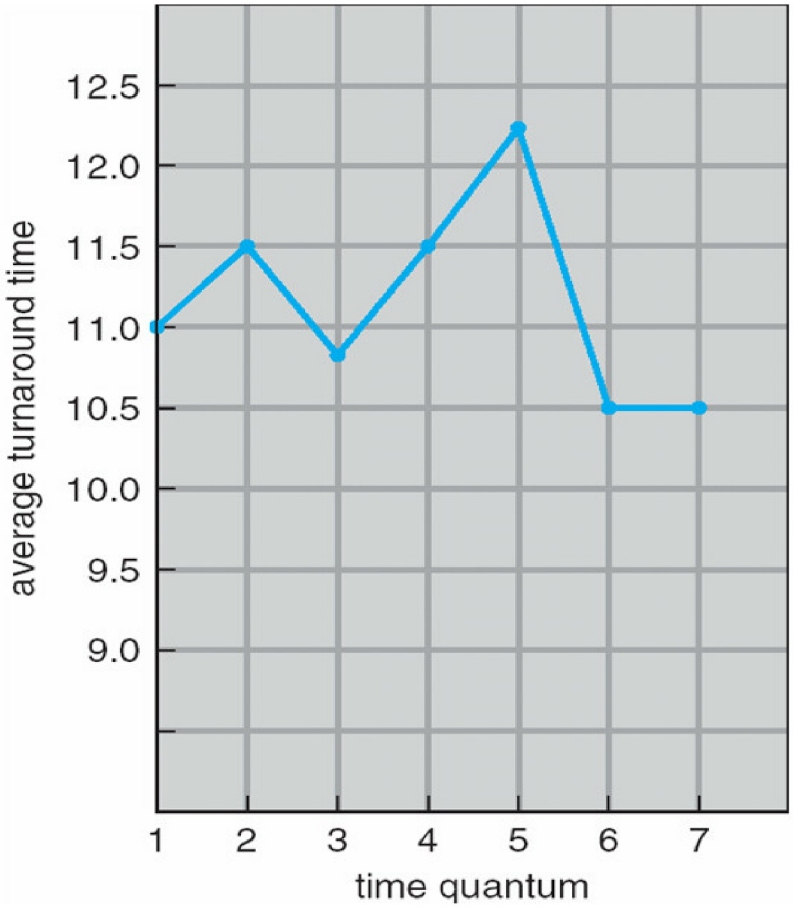
Avg: $\frac{17}{3} = 5.66$



Time Quantum and Context Switch Time



Turnaround Time Varies With The Time Quantum

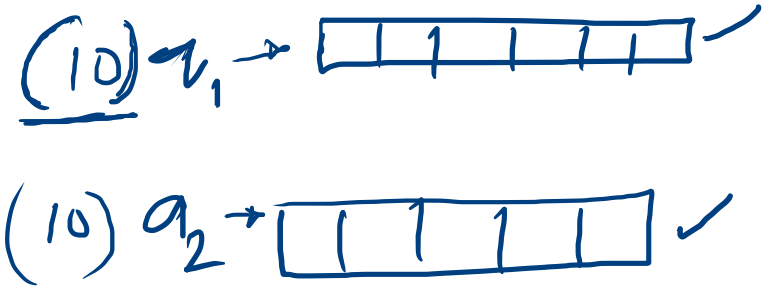


process	time
P_1	6
P_2	3
P_3	1
P_4	7

80% of CPU bursts should be shorter than quantum

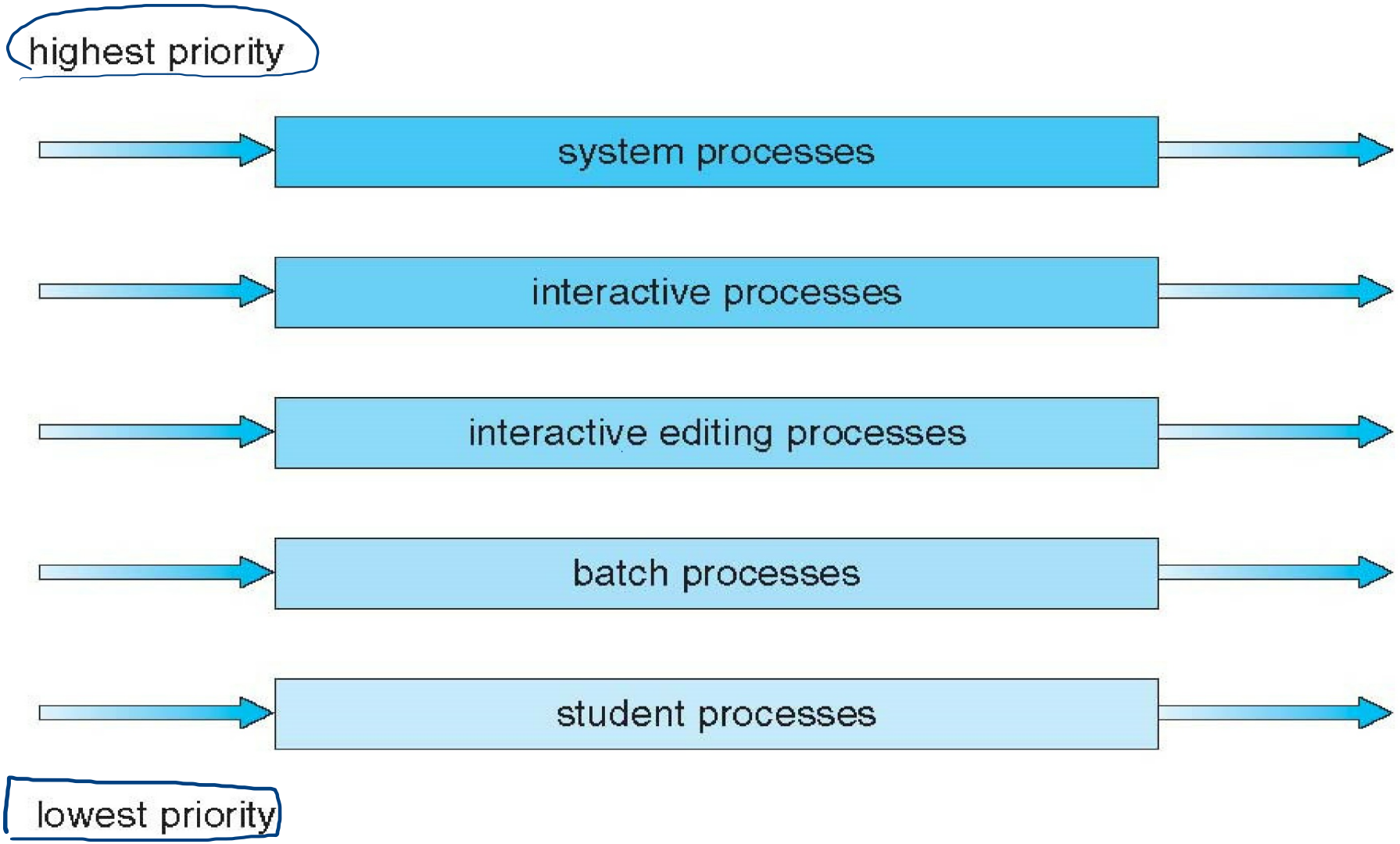
Multilevel Queue

- Another class of scheduling algorithm needs- in which processes are classified into different groups, e.g.:
  - foreground (interactive) processes
  - background (batch) processes
- They have different response time requirements-so different scheduling needs.
- Foreground processes may have priority over background processes.
- A multilevel queue-scheduling algorithm partitions the ready queue into several separate queues-we can see it in the figure of next slide:-
- Each queue has its own scheduling algorithm:
  - Foreground queue scheduled by – RR algorithm
  - Background queue scheduled by – FCFS algorithm



- Scheduling must be done between the queues:
  - Fixed priority preemptive scheduling; (i.e., serve all from foreground then from background). Possibility of starvation.
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., foreground queue can be given 80% of the CPU time for RR-scheduling among its processes, while 20% to background in FCFS manner.

Multilevel Queue Scheduling



Multilevel Feedback Queue scheduling

- In multilevel queue, processes do not move from one queue to the other----But
- Multilevel Feedback Queue scheduling, allows a process to move between queues.
- If a process uses too much CPU time, it will be moved to a lower priority queue.
- Similarly, a process that waits too long in a lower-priority queue may me moved to a higher-priority queue.

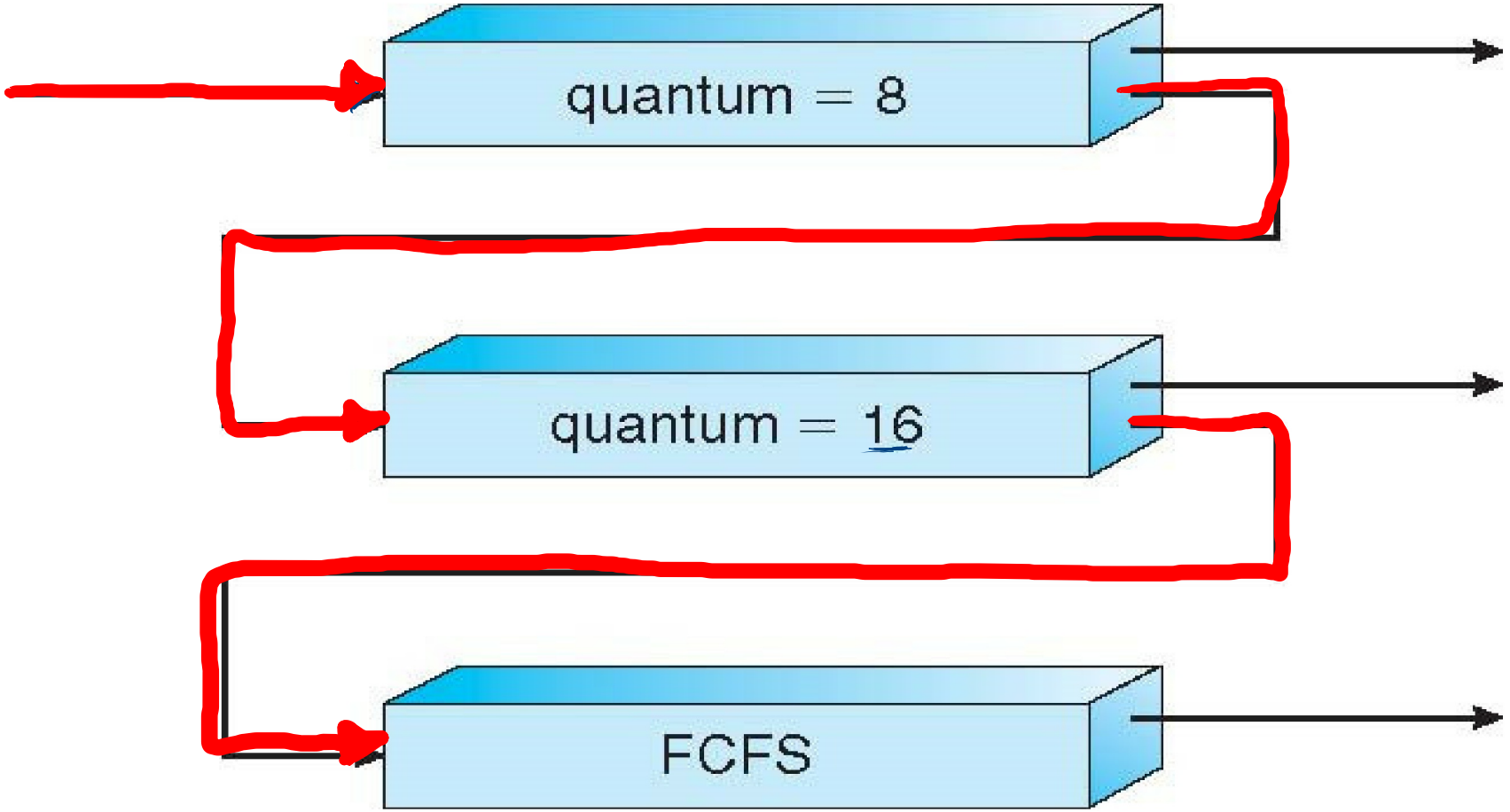
Multilevel Feedback Queue scheduling

- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service

Example of Multilevel Feedback Queue

- Three queues: (can see the figure in next slide)
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS
- Scheduling
  - A new job enters queue Q_0 which is served for RR
    - When it gains CPU, job receives 8 milliseconds
    - If it does not finish in 8 milliseconds, job is moved to queue Q_1
  - At Q_1 job is again served RR and receives 16 additional milliseconds
    - If it still does not complete, it is preempted and moved to queue Q_2

Multilevel Feedback Queues



End of Chapter 6