



# **Operating Systems**

## **CPU Scheduling-Part2**

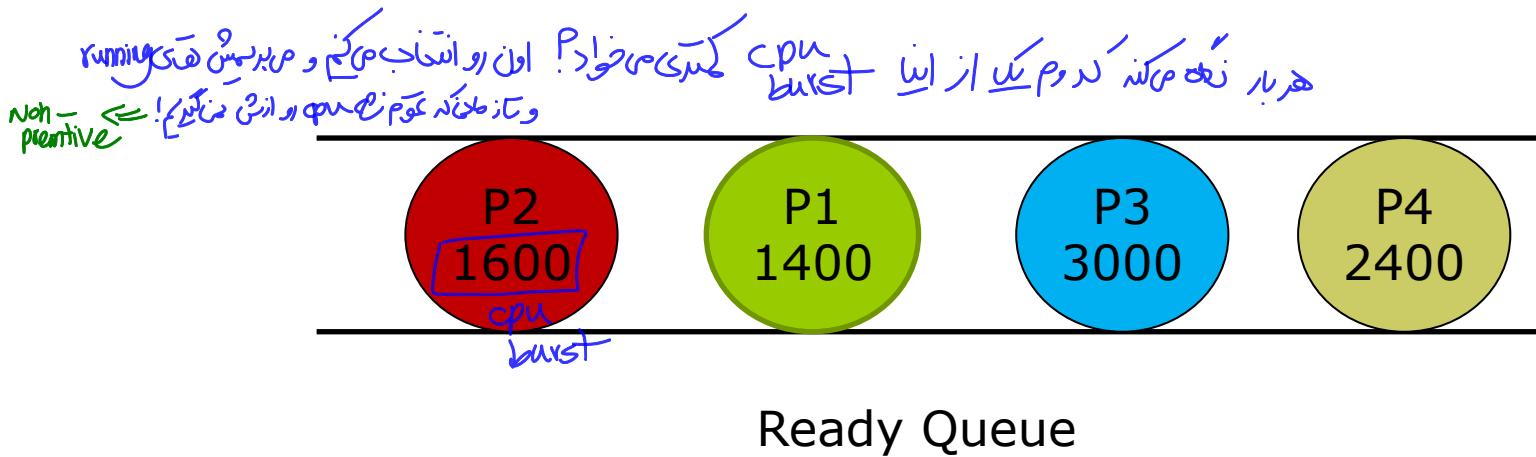
Seyyed Ahmad Javadi

[sajavadi@aut.ac.ir](mailto:sajavadi@aut.ac.ir)

Fall 2021

# 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.



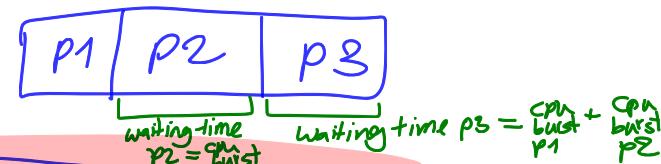
ارد: اگر معلم بردارهای  $cpu\_burst$  کوتاه باش، حکمه خوبت به اینکه  
starvation طولانی داشته باز! ==> باید صدای  $cpu\_burst$

# Shortest-Job-First (SJF) Scheduling (cont.)

## SJF is optimal

- Gives minimum average waiting time for a given set of processes.

Civil Faculty queue SJF for CPU



## Preemptive version called shortest-remaining-time-first

برهان حلف:



حالاً لو نحن مررنا بـ PM فـ PN ينادي  
ألا أجزئي ، سأنتهي في وقت لاحق  
↓  
waiting time

## The difficulty is knowing the length of the next CPU request

- Could ask the user

(وَالْأَكْبَرُ مَنْ يَعْلَمُ بِزَانِي! 4000 جندي! 5000 جندي! 10000 جندي! 15000 جندي! 20000 جندي! 25000 جندي! 30000 جندي! 35000 جندي! 40000 جندي! 45000 جندي! 50000 جندي! 55000 جندي! 60000 جندي! 65000 جندي! 70000 جندي! 75000 جندي! 80000 جندي! 85000 جندي! 90000 جندي! 95000 جندي! 100000 جندي! 105000 جندي! 110000 جندي! 115000 جندي! 120000 جندي! 125000 جندي! 130000 جندي! 135000 جندي! 140000 جندي! 145000 جندي! 150000 جندي! 155000 جندي! 160000 جندي! 165000 جندي! 170000 جندي! 175000 جندي! 180000 جندي! 185000 جندي! 190000 جندي! 195000 جندي! 200000 جندي! 205000 جندي! 210000 جندي! 215000 جندي! 220000 جندي! 225000 جندي! 230000 جندي! 235000 جندي! 240000 جندي! 245000 جندي! 250000 جندي! 255000 جندي! 260000 جندي! 265000 جندي! 270000 جندي! 275000 جندي! 280000 جندي! 285000 جندي! 290000 جندي! 295000 جندي! 300000 جندي! 305000 جندي! 310000 جندي! 315000 جندي! 320000 جندي! 325000 جندي! 330000 جندي! 335000 جندي! 340000 جندي! 345000 جندي! 350000 جندي! 355000 جندي! 360000 جندي! 365000 جندي! 370000 جندي! 375000 جندي! 380000 جندي! 385000 جندي! 390000 جندي! 395000 جندي! 400000 جندي!

- Estimate (we do not cover this in the class and the exams)

# Example of SJF

Non-preemptive

## Process

$P_1$

$P_2$

$P_3$

$P_4$

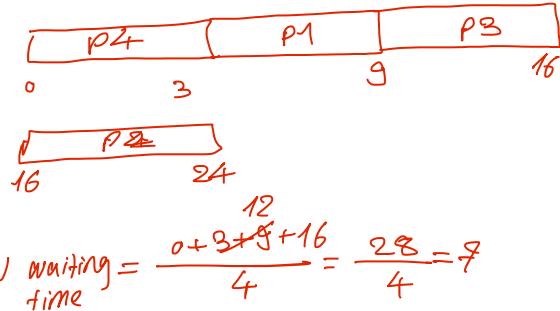
## Burst Time

6

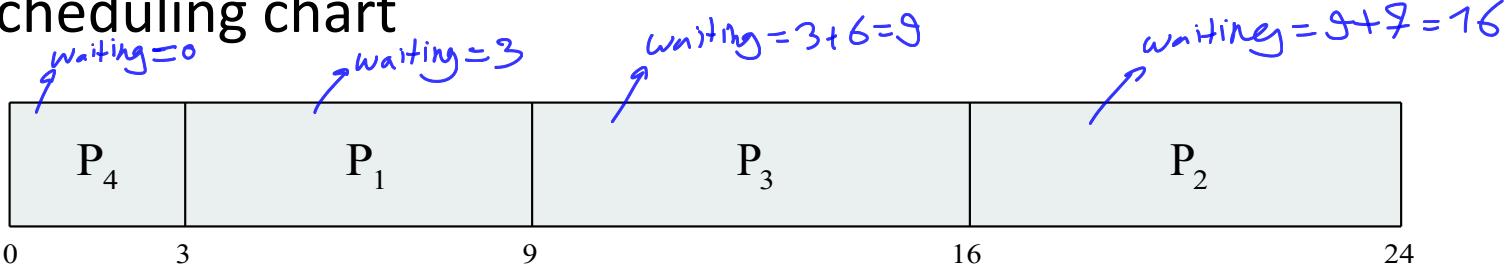
8

7

3



### SJF scheduling chart



$$\text{Average waiting time} = (3 + 16 + 9 + 0) / 4 = 7$$



# Round Robin (RR)

رُونڈ رُوبِن

- 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.  
*(preemptive!)*
- If there are  $n$  processes in the ready queue and the time quantum is  $q$ :
  - Each process gets  $1/n$  of the CPU time in chunks of at most  $q$  time units at once.
  - No process waits more than  $(n-1)q$  time units.

جُمَعَةٌ وَقْتٌ مُحْسَنٌ (Starvation). دعْنَى ادْعَنَى نَارَهُ خَبَيْرَهُ اجْرَاهُ

کاریکاتوری



# Round Robin (RR) (cont.)

- Timer *interrupts* every quantum to schedule next process
- Performance

- $q$  large  $\Rightarrow$  FIFO (like CFS)  $\Rightarrow$   
جفن و صنایع بزرگ و عدای کار  
هر داده همچو قسمی دارند، و جعلی  
صنایع نیز بزرگ دارند
- $q$  small  $\Rightarrow q$  must be large with respect to context switch,

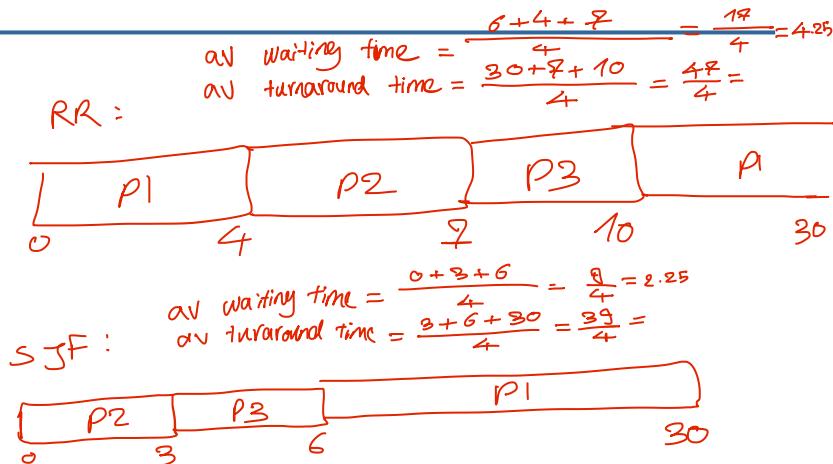
otherwise overhead is too high

لایه اینترنالیتی هر proc' را درین در کامپیو  
اینام بدهید و لایم کومنیٹی همچنین

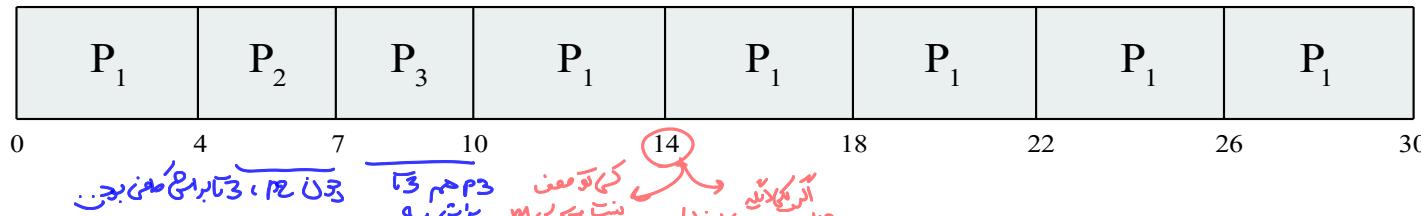


# Example of RR with Time Quantum = 4

<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

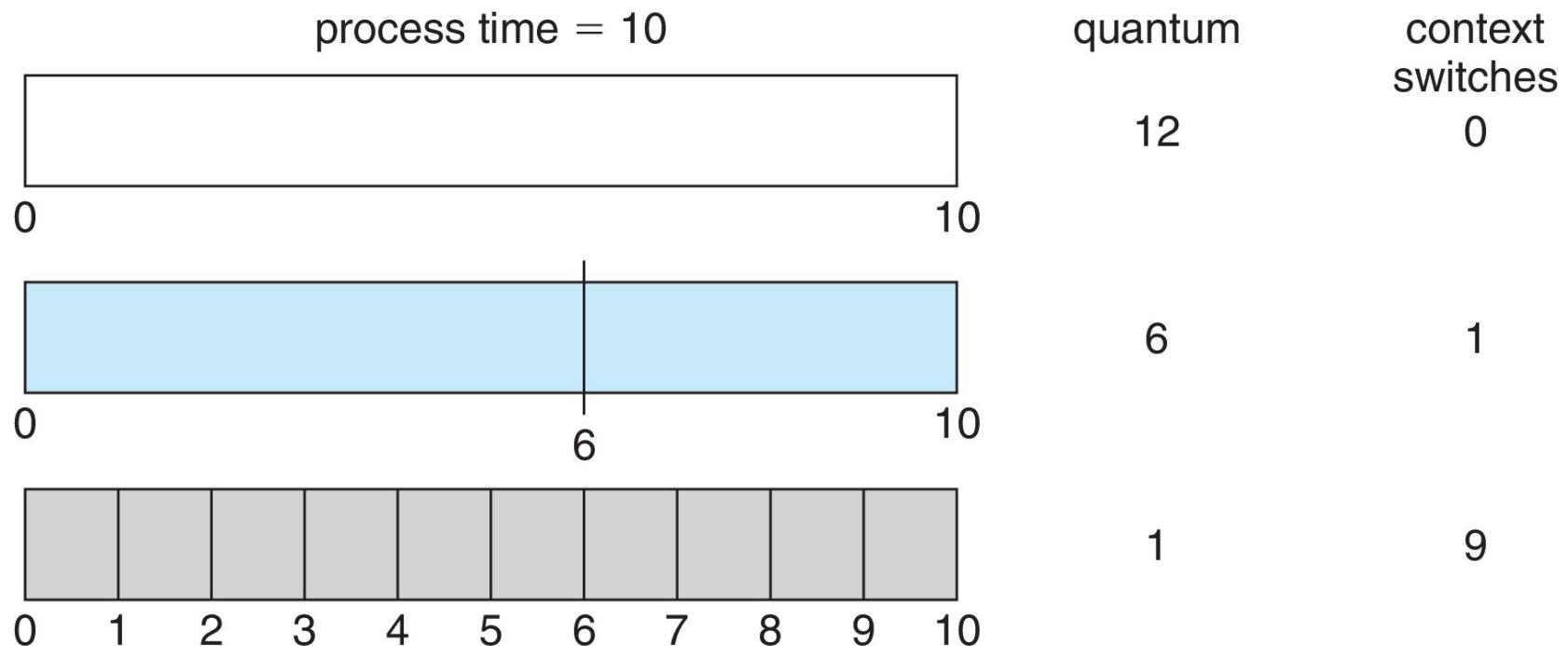


- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better **response**
- q** should be large compared to context switch time
  - q usually 10 milliseconds to 100 milliseconds,
  - Context switch < 10 microseconds

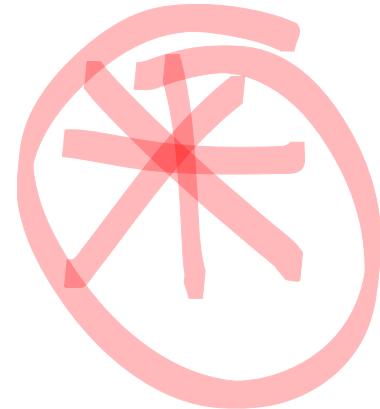
# Time Quantum and Context Switch Time



# Turnaround Time Varies With The Time Quantum



process	time
$P_1$	6
$P_2$	3
$P_3$	1
$P_4$	7



خطای کوئنچری :-

↑  
best practice

A rule of thumb is that

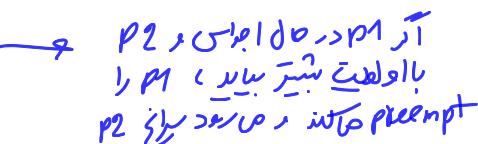
80% of CPU bursts should

be shorter than  $q$

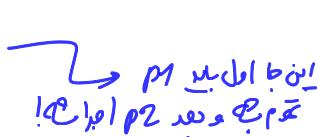
! خطای

# 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 → 

P2 > P1 (higher priority)  
P1 runs until its burst time is over  
P2 runs until its burst time is over, into preempt

- Nonpreemptive → 

P1 runs until its burst time is over  
P2 runs until its burst time is over, and P1 resumes

- SJF is priority scheduling where priority is the *inverse of predicted next CPU burst time*

*next CPU burst time* → اولین بروز از  $\text{cpu burst}$   $\Rightarrow$   $\text{cpu burst} = \frac{\text{integer}}{\text{for priority}}$

# Priority Scheduling

## ■ Problem ≡ Starvation

- Low priority processes may never execute

## ■ Solution ≡ Aging

- As time progresses increase the priority of the process

۱۲۷ لے ملایا الگ امدادیت کی کوئی پردازه صحتی کو نہیں  
نمیزدگی میں ملائی کر دیں اور این کو دکھنے کی  
(امدادیت آئی را زیاد کرنے).

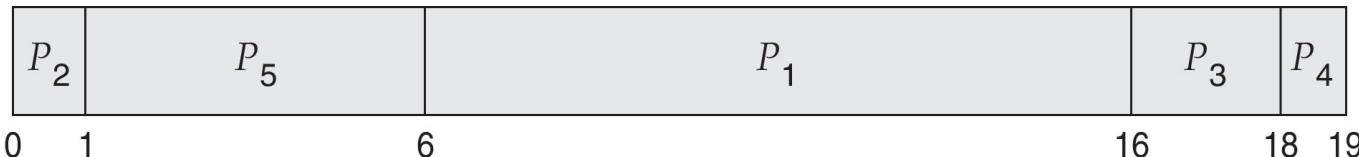


# Example of Priority Scheduling

<u>Process</u>	<u>Burst Time</u>	<u>Priority</u>
$P_1$	10	3
$P_2$	1	1
$P_3$	2	4
$P_4$	1	5
$P_5$	5	2

مدة  
ال Burst  
أول

- Priority scheduling Gantt Chart



- Average waiting time = 8.2

# Priority Scheduling w/ Round-Robin

## Process

## Burst Time

## Priority

$P_1$

~~4 2~~

3

$P_2$

~~5 2 1 0~~

2

$P_3$

~~8 3 4 0~~

2

$P_4$

7

1

$P_5$

~~3 1~~

3

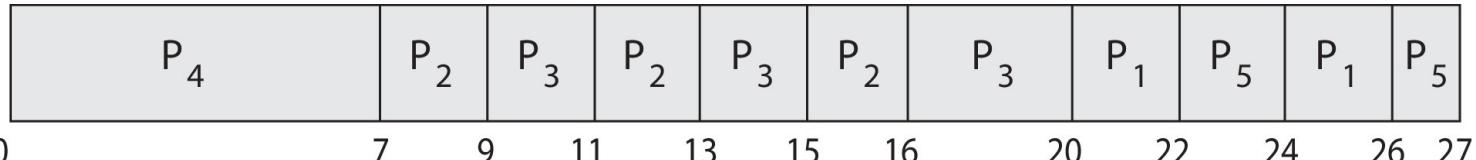
ب مح رجی ندارن!

اگر اینها از Round-Robin استفاده می‌کنند  
بعد از اینجا P2 نزدیک P3 اولین burst است  
برای انتخاب می‌کنند! (رجی ندارند!)

- Run the process with the highest priority. Processes with the same priority run round-robin

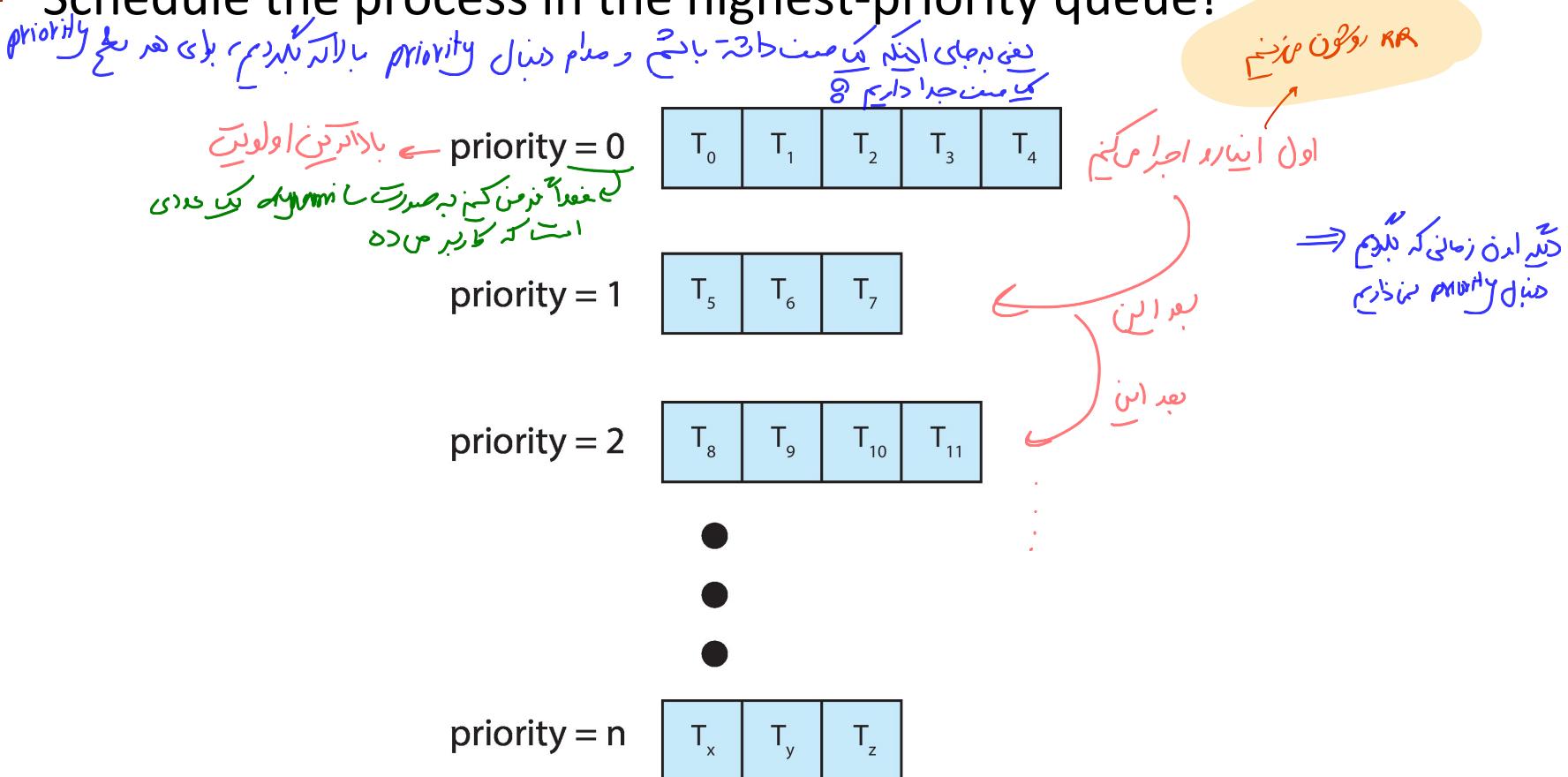


- Gantt Chart with time quantum = 2



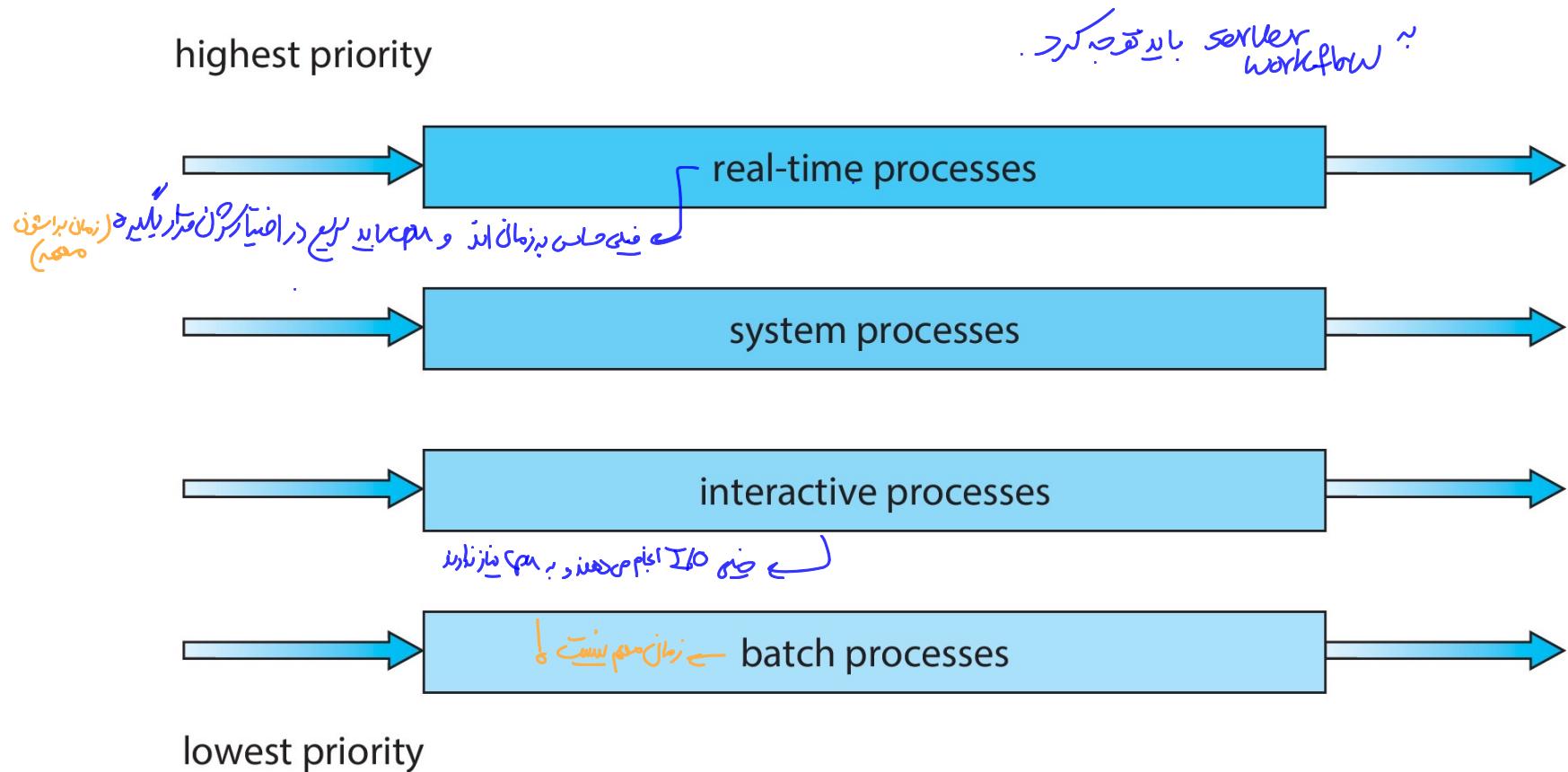
# Multilevel Queue

- With priority scheduling, have separate queues for each priority.
- Schedule the process in the highest-priority queue!



# Multilevel Queue

- Prioritization based upon process type



# Multilevel Feedback Queue

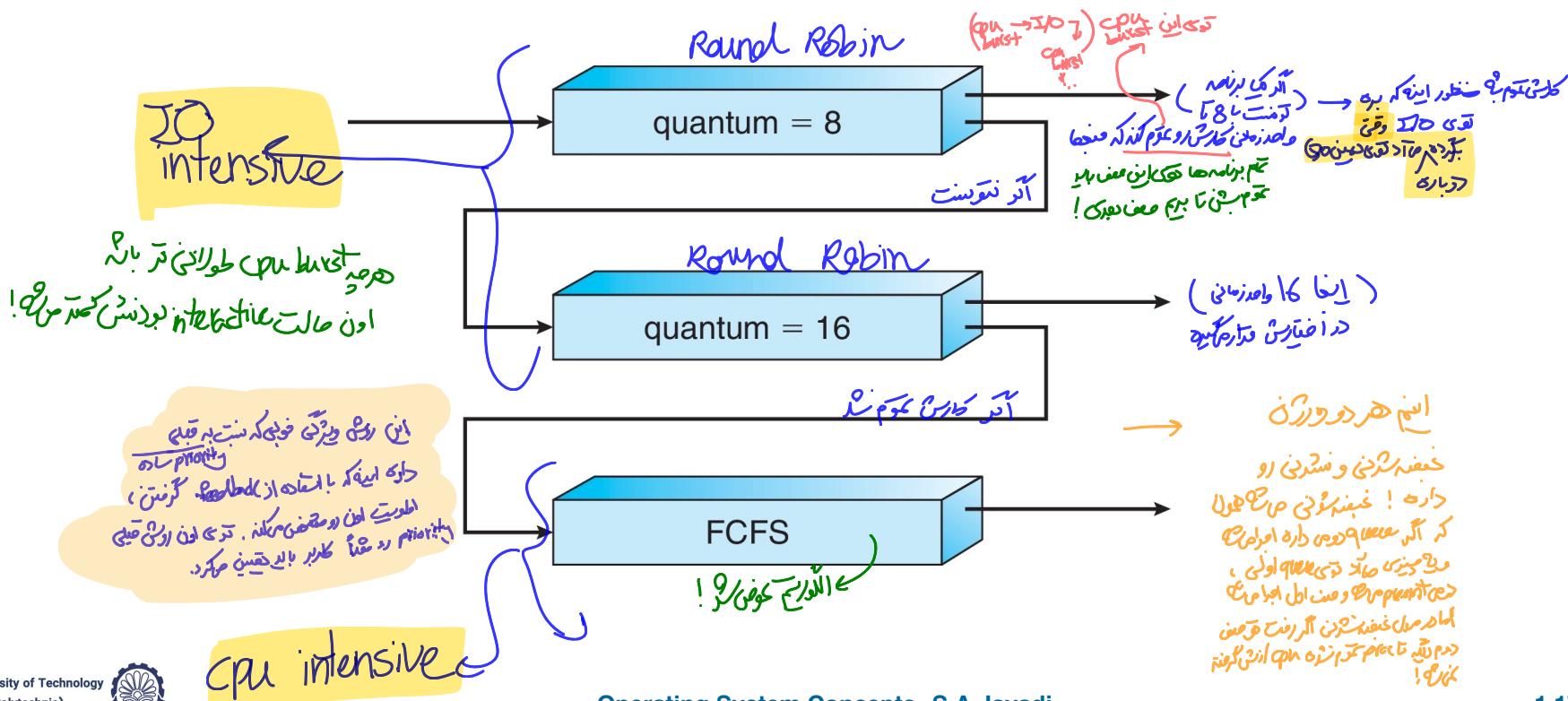
بررسی سازماندهی که دارای درجات مختلف حقیقی است

- A process can move between the various queues.  
حکم بالاترین اولویت (ساده اور حاصل کننده) با این ترتیب قبل از سایر ترتیب های general و
- Multilevel-feedback-queue defined by the following parameters:
  - Number of queues
  - Scheduling algorithms for each queue
  - Method used to determine when to upgrade a process  
امروزه ساده باش
  - Method used to determine when to demote a process  
امروزه ساده باشند
  - Method used to determine which queue a process will enter when that process needs service  
→
    - پنجه از منی
    - to system برای برخی procs
    - حالاتی برای برخی procs
    - دسترسی آسان برای برخی procs
    - دسترسی ساده برای برخی procs
- Aging can be implemented using multilevel feedback queue

# Example of Multilevel Feedback Queue

## Three queues:

- $Q_0$  – RR with time quantum 8 milliseconds
- $Q_1$  – RR time quantum 16 milliseconds
- $Q_2$  – FCFS



# Example of Multilevel Feedback Queue (cont.)

## Scheduling

- A new process enters queue  $Q_0$  which is served in RR
  - When it gains CPU, the process receives 8 milliseconds
  - If it does not finish in 8 milliseconds, the process is moved to queue  $Q_1$
- At  $Q_1$  job is again served in RR and receives 16 additional milliseconds
  - If it still does not complete, it is preempted and moved to queue  $Q_2$

