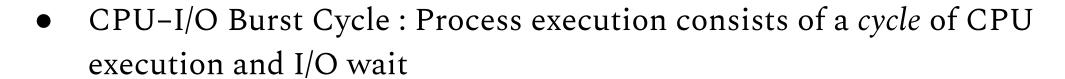
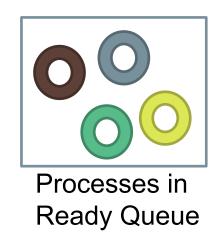
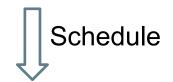
OPERATING SYSTEMS CPU Scheduling

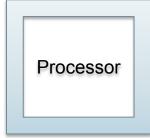
Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- Continuous Cycle :
 - one process has to wait (I/O)
 - Operating system takes the CPU away
 - Give CPU to another process
 - This pattern continues









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
- -- 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, waiting 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

OPERATING SYSTEMS

CPU Scheduling Algorithms - First Come First Serve (FCFS)

First-Come, First-Served (FCFS) Scheduling

Process	Arrival Time	Burst Time
P ₁	0	8
P ₂	1	4
P ₃	2	9
P ₄	3	5

The Gantt Chart for the schedule is:

	P ₁	P ₂		P ₃	P ₄	
0		8	12	2	1 26	

Ready Queue

- $P_1(8)$
- 1 P₂(4)
- $2 \qquad | P_2(4) | P_3(9)$
- 3 $P_2(4) P_3(9) P_4(5)$
- 8 P₃(9) P₄(5)
- 12 P₄(5)
- 21 Empty

26 Empty

OPERATING SYSTEMS

CPU Scheduling Algorithms - Shortest Job First (SJF)

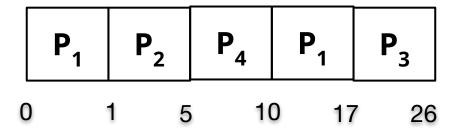
Shortest-Job-First (SJF) Scheduling

- 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 given to the process it cannot be preempted until completes its CPU burst
 - o **preemptive** if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF)
- <u>SJF is optimal</u> gives minimum average waiting time for a given set of processes

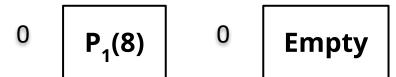
Example of Shortest-remaining-time-first

Process	Arrival Time	Burst Time
P ₁	0	8
P ₂	1	4
P ₃	2	9
P ₄	3	5

Gantt Chart:



Ready Queue



OPERATING SYSTEMS

CPU Scheduling Algorithms - Priority Scheduling

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
 - Nonpreemptive
- 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 being paid for computer use, department sponsoring works etc.

Example of Priority Scheduling (Preemptive)

Process	Arrival Time	Burst Time	Priority
P ₁	0	8	3
P ₂	1	4	1
P ₃	2	9	4
P ₄	3	5	5

Priority scheduling Gantt Chart

- Problem = **Starvation** low priority processes may never execute
- Solution \equiv **Aging** as time progresses increase the priority of the process

Ready Queue

- $0 \quad P_1(8) \quad 0 \quad Empty$
- 1 P₂(4) 1 P₁(7)
- 2 P₁(7) P₃(9)
- 3 $P_1(7)$ $P_3(9)$ $P_4(5)$
- 5 P₃(9) P₄(5)
- 12 P₄(5)
- 21 Empty
- 26 Empty

OPERATING SYSTEMS

CPU Scheduling Algorithms - Round Robin (RR)

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)q time units.
- Timer interrupts every quantum to schedule next process
- Performance
 - \circ q large \Rightarrow FIFO
 - o $q \text{ small} \Rightarrow q \text{ must be large with respect to context switch, otherwise overhead is too high}$

Example of RR with Time Quantum = 4

Process	Arrival Time	Burst Time
P ₁	0	8
P ₂	1	4
P ₃	2	9
P ₄	3	5

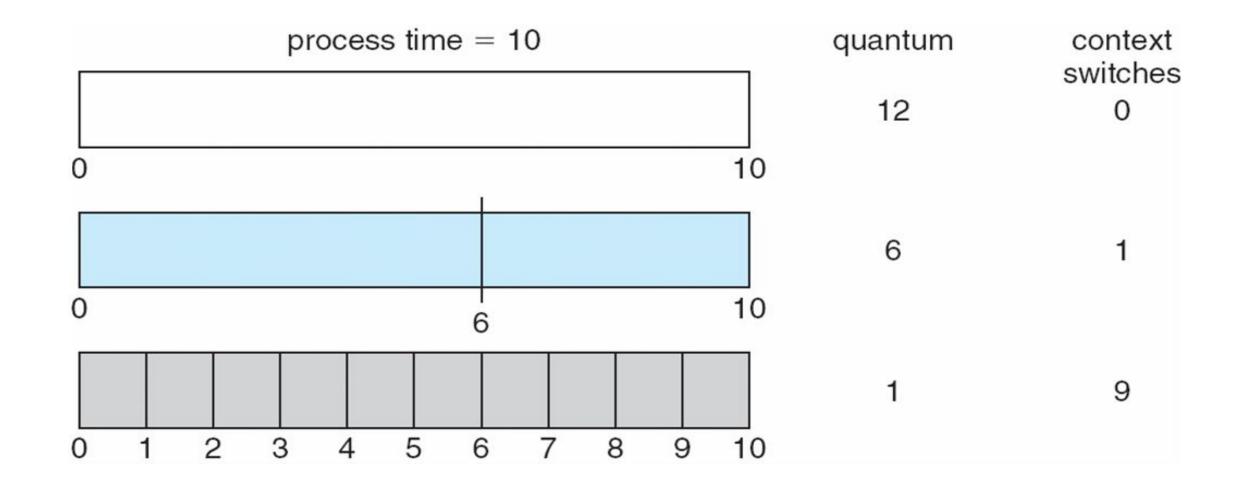
The Gantt chart is:

	P ₁	P ₂	P ₃	P ₄	P ₁	P ₃	P ₄	P ₃
0	4	8	12	2 1	6 2	0 2	4 2	5 26

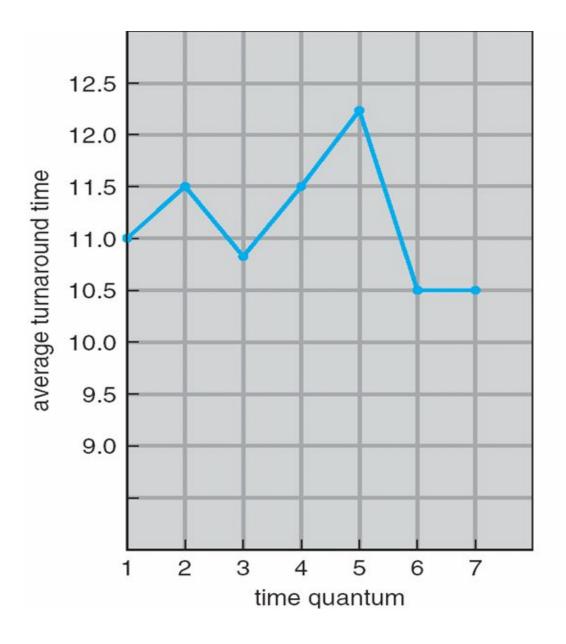
Ready Queue:

- $0 \qquad P_1(8)$
- 1 P₂(4)
- 2 $P_{2}(4)$ $P_{3}(9)$
- 3 $P_2(4)$ $P_3(9)$ $P_4(5)$
- 4 $P_3(9)$ $P_4(5)$ $P_1(4)$
- 8 P₄(4) P₁(4)
- 12 P₁(4) P₃(5)
- 16 $P_3(5)$ $P_4(1)$
- 20 P₄(1)
- 24 P₃(1)
- 25 Empty 26 Empty

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

OPERATING SYSTEMS

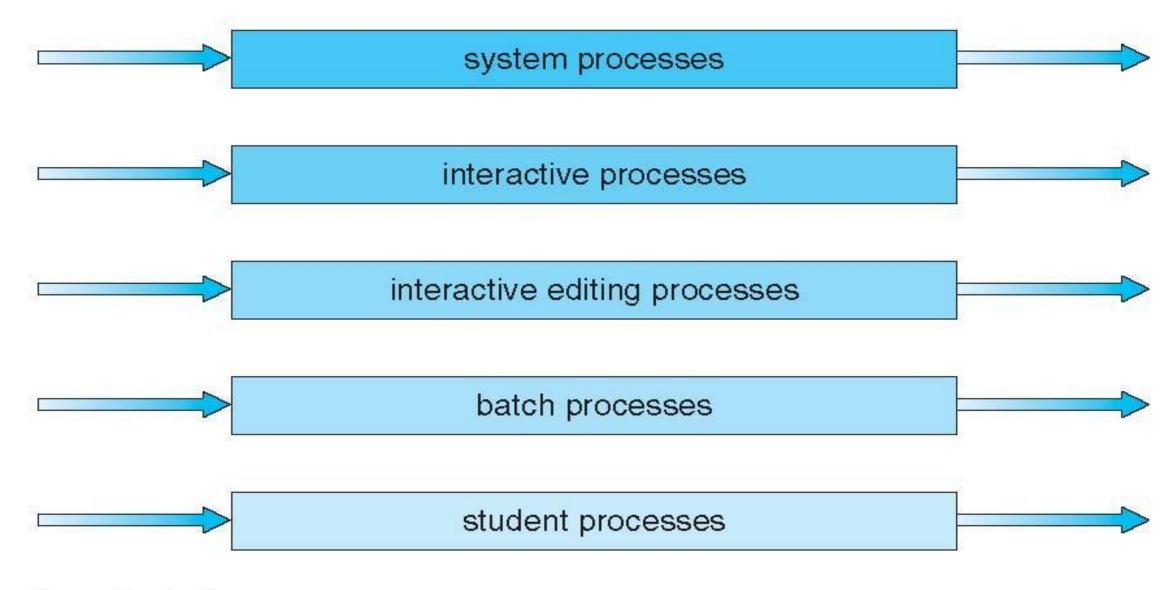
CPU Scheduling Multilevel Queue, Multilevel Feedback Queue

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

highest priority

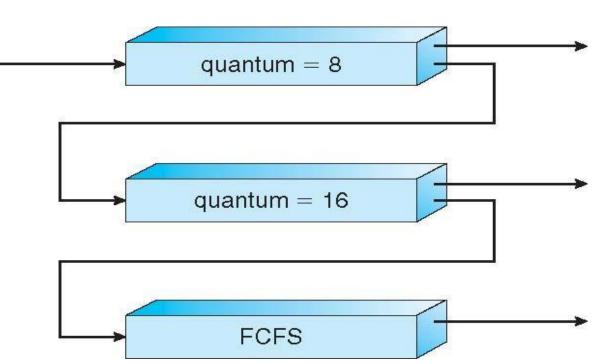


lowest priority

Multilevel Feedback Queue scheduling

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

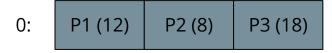
- Three queues: (can see the figure in next slide)
 - \circ Q_0 RR with time quantum 8 milliseconds
 - \circ Q_1 RR time quantum 16 milliseconds
 - \circ $Q_2 FCFS$
- Scheduling
 - A new job enters queue Q_0 which is served for RR
 - When it gains CPU, job receives 8 milliseconds
 - If it does not finish in 8 milliseconds, job is moved to queue Q₁
 - 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



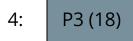
Problem 1:

Processes	Burst Time
P1	12
P2	8
P3	18

Queue 0 (Priority 0): Round-robin (quantum=2)







4: Empty

Queue 1 (Priority 1): Round-robin (quantum=4)



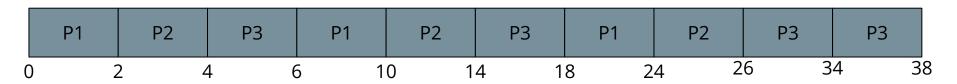


Queue 2 (Priority 2): Round-robin (quantum=8)



Gantt Chart:

CPU:



Problem 1:

Processes	Burst Time
P1	12
P2	8
P3	18

Gantt Chart:

CPU:

	P1	P2	P3	P1	P2	P3	P1	P2	P3	P3
0	2	2	4 6	5 1	0 1	4 1	8 2	4 2	6 3	4 38

Waiting Time:

P1=(0-0)+(6-2)+(18-10)=12

P2=(2-0)+(10-4)+(24-14)=18

P3=(4-0)+(14-6)+(26-18)+(34-34)=20

Avg=(12+18+20)/3=16.67

Response Time:

P1=(0-0)=0

P2=(2-0)=2

P3=(4-0)=4

Avg=(0+2+4)/3=2

Turnaround Time:

P1=(24-0)=24

P2=(26-0)=26

P3=(38-0)=38

Avg=(24+26+38)/3=29.33

P4

22

P2

P4

25

P1

27

P1

30

29

Р3

P4

38

P4

40

Р3

45

Р3

P2

15

13

P1

Problem 2:

P2

CPU:

Р3

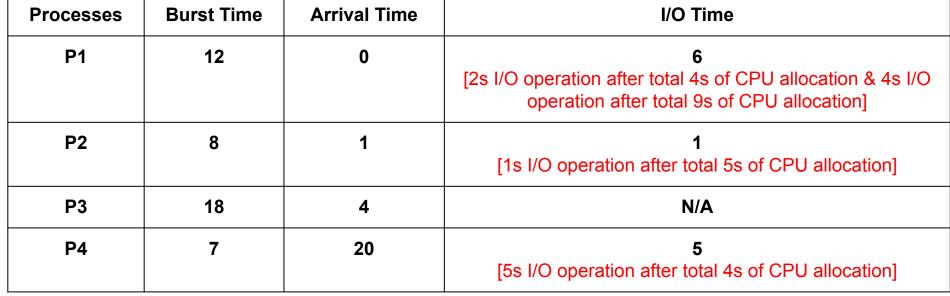
P1

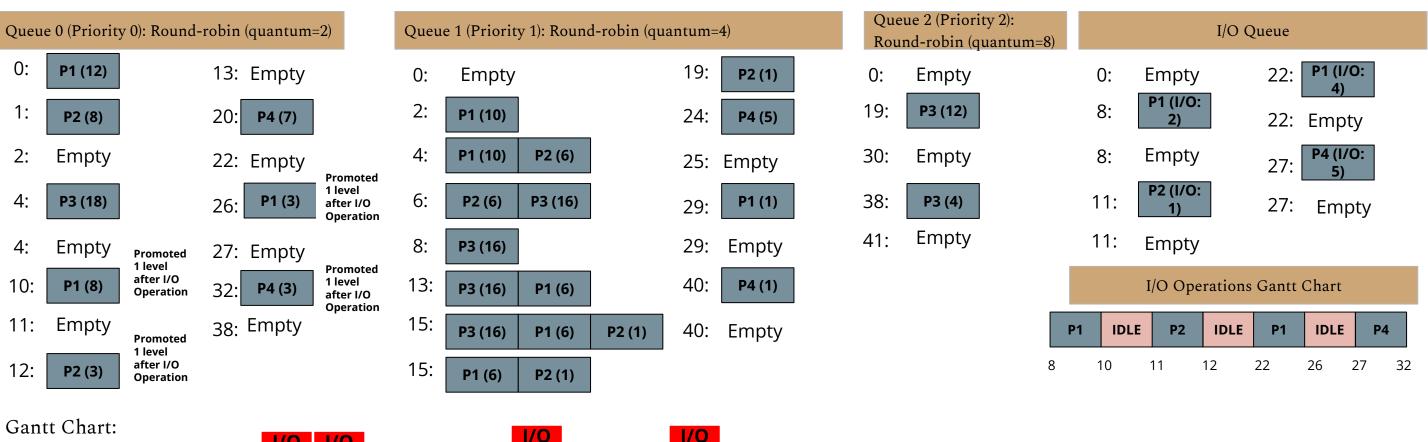
1/0

11

P1

P2



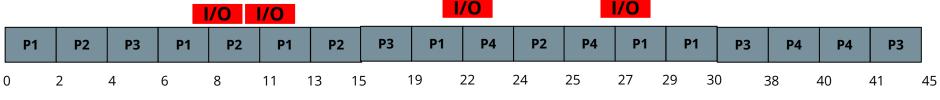


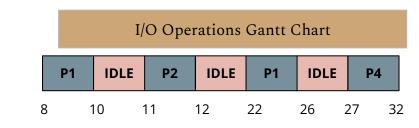
Problem 2:

Processes	Burst Time	Arrival Time	I/O Time
P1	12	0	6 [2s I/O operation after total 4s of CPU allocation & 4s I/O operation after total 9s of CPU allocation]
P2	8	1	1 [1s I/O operation after total 5s of CPU allocation]
Р3	18	4	N/A
P4	7	20	5 [5s I/O operation after total 4s of CPU allocation]



CPU:





Waiting Time:

Response Time:

$$P2=(2-1)=1$$

Turnaround Time: