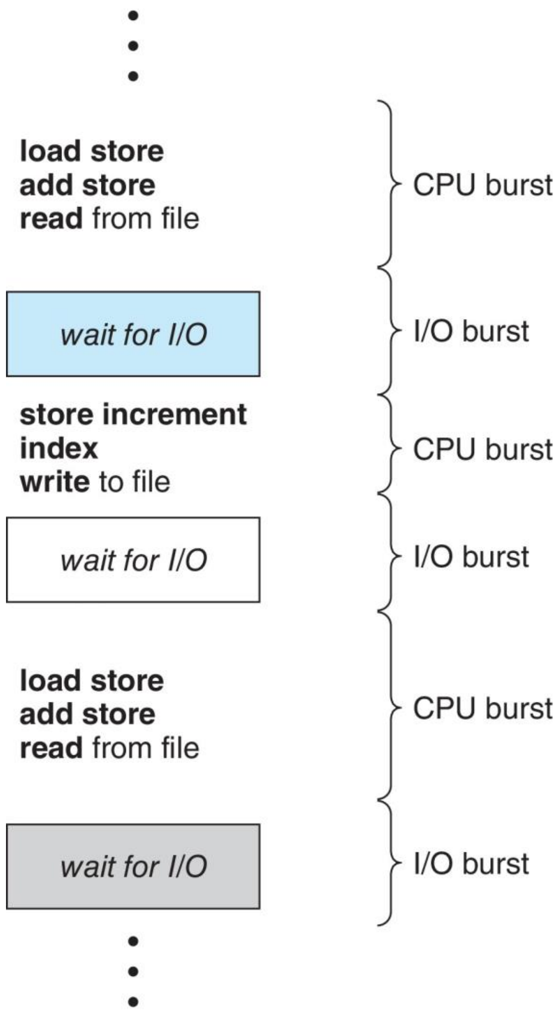
CPU Scheduling

Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle – Process execution consists of a **cycle** of CPU execution and I/O wait
- **CPU burst** followed by **I/O burst**
- CPU burst distribution is of main concern



CPU Scheduler

- The **CPU scheduler** selects from among the processes in ready queue, and allocates a CPU core to one of them
  - Queue may be ordered in various ways
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state
  2. Switches from running to ready state
  3. Switches from waiting to ready
  4. Terminates
- For situations 1 and 4, there is no choice in terms of scheduling. A new process (if one exists in the ready queue) must be selected for execution.
- For situations 2 and 3, however, there is a choice.

Preemptive and Nonpreemptive Scheduling

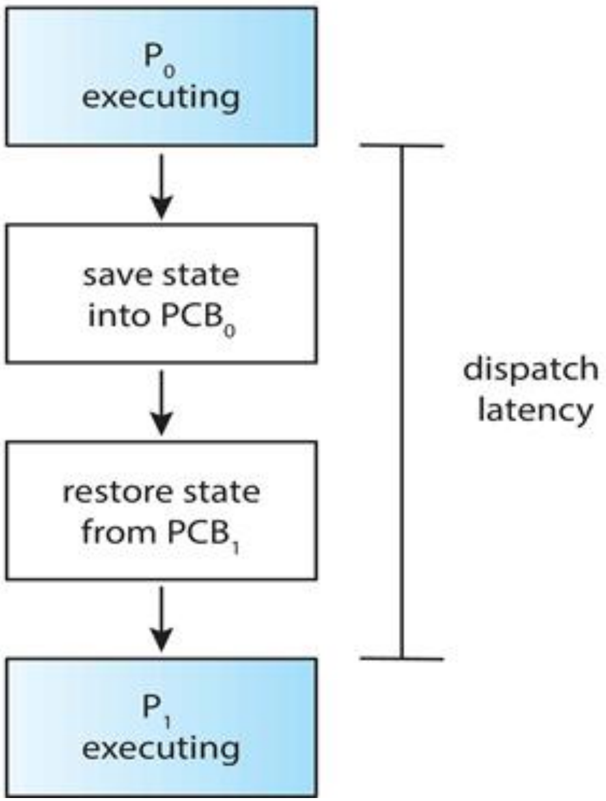
- When scheduling takes place only under circumstances 1 and 4, the scheduling scheme is **nonpreemptive**.
- Otherwise, it is **preemptive**.
- Under Nonpreemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases it either by terminating or by switching to the waiting state.
- Virtually all modern operating systems including Windows, MacOS, Linux, and UNIX use preemptive scheduling algorithms.

Preemptive Scheduling and Race Conditions

- Preemptive scheduling can result in race conditions when data are shared among several processes.
- Consider the case of two processes that share data. While one process is updating the data, it is preempted so that the second process can run. The second process then tries to read the data, which are in an inconsistent state.

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the CPU scheduler; this involves:
  - Switching context
  - Switching to user mode
  - Jumping to the proper location in the user program to restart that program
- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running



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 (Completion Time of process – Arrival Time of process)
- **Waiting time** – amount of time a process has been waiting in the ready queue (Turn Around Time of process – Burst Time of process)
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced.

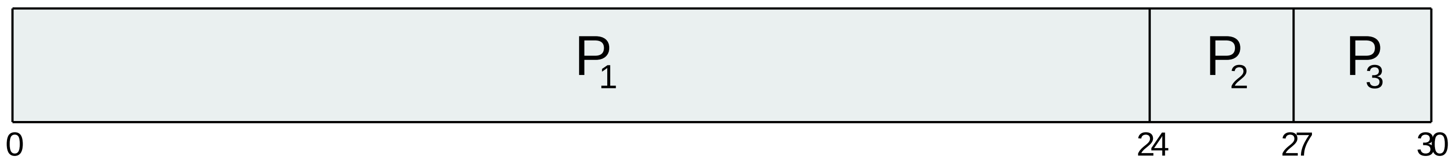
Scheduling Algorithm Optimization Criteria

- 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: $(0 + 24 + 27)/3 = 17$

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order:

$$P_2 \text{ , } P_3 \text{ , } P_1$$

- 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

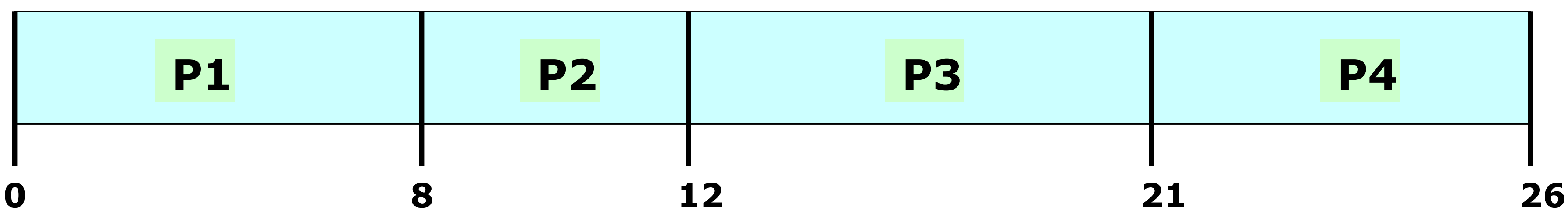
Example of FCFS Scheduling

- Find average turn around time and average waiting time of the following scenario using FCFS scheduling algorithm.

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

Example of FCFS Scheduling (Cont.)

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



Example of FCFS Scheduling (Cont.)

Process	Arrival Time (A.T.)	Burst Time (B.T.)	Completion Time (C.T.)	Turn Around Time (T.A.T) TAT=CT-AT	Waiting Time (W.T.) WT=TAT-BT
1	0	8	8	8-0=8	8-8=0
2	1	4	12	12-1=11	11-4=7
3	2	9	21	21-2-19	19-9=10
4	3	5	26	26-3=23	23-5=18

- Average turn around time = (8+11+19+23)/4 = 61/4 = 15.25
- Average waiting time = (0+7+10+18)/4 = 35/4 = 8.75

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
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user

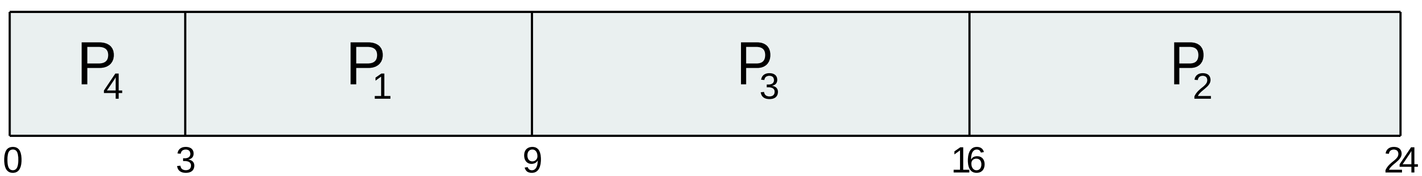
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
- Preemptive version called **shortest-remaining-time-first**
- How do we determine the length of the next CPU burst?
  - Could ask the user
  - Estimate

Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart (Gantt chart)



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

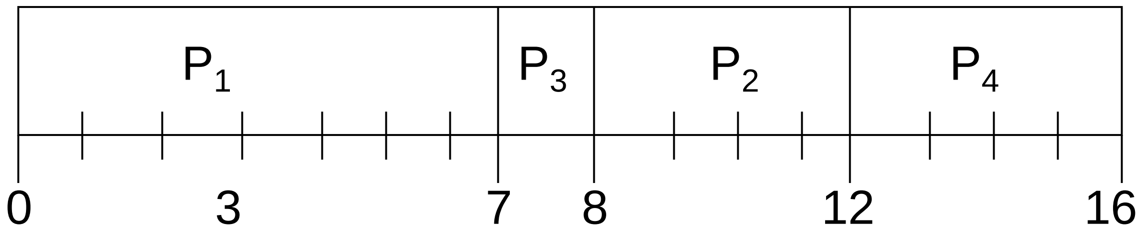
Example of Non-Preemptive SJF

- Find average turn around time and average waiting time of the following scenario using non-preemptive SJF scheduling algorithm.

Process	Arrival Time	Burst Time
1	0	7
2	2	4
3	4	1
4	5	4

Example of Non-Preemptive SJF (Cont.)

Process	Arrival Time	Burst Time
1	0	7
2	2	4
3	4	1
4	5	4



Example of Non-Preemptive SJF (Cont.)

Process	Arrival Time (A.T.)	Burst Time (B.T.)	Completion Time (C.T.)	Turn Around Time (T.A.T) TAT=CT-AT	Waiting Time (W.T.) WT=TAT-BT
1	0	7	7	7-0=7	7-7=0
2	2	4	12	12-2=10	10-4=6
3	4	1	8	8-4=4	4-1=3
4	5	4	16	16-5=11	11-4=7

- Average turn around time = (7+10+4+11)/4 = 32/4 = 8
- Average waiting time = (0+6+3+7)/4 = 16/4 = 4

Determining Length of Next CPU Burst

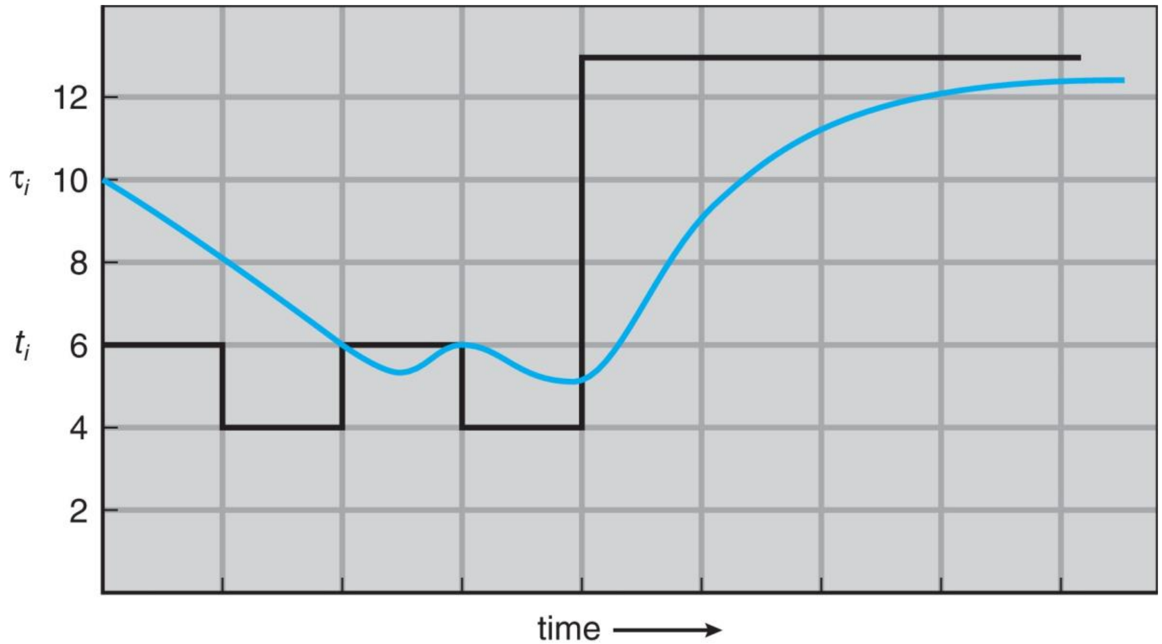
- Can only estimate the length – should be similar to the previous one
  - Then pick process with shortest predicted next CPU burst
- Can be done by using the length of previous CPU bursts, using exponential averaging

  1. $t_n =$ actual length of n^{th} CPU burst
  2. $\tau_{n+1} =$ predicted value for the next CPU burst
  3. $\alpha, 0 \le \alpha \le 1$
  4. Define:

  $$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\tau_n.$$

- Commonly, α set

Prediction of the Length of the Next CPU Burst



CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Examples of Exponential Averaging

- α =0
  - $\tau_{n+1} = \tau_n$
  - Recent history does not count
- α =1
  - $\tau_{n+1} = \alpha\, t_n$
  - Only the actual last CPU burst counts
- If we expand the formula, we get:

$$\begin{aligned}\tau_{n+1} = \alpha\, t_n &+ (1 - \alpha)\alpha\, t_{n-1} + \dots \\ &+ (1 - \alpha)^j \alpha\, t_{n-j} + \dots \\ &+ (1 - \alpha)^{n+1} \tau_0\end{aligned}$$

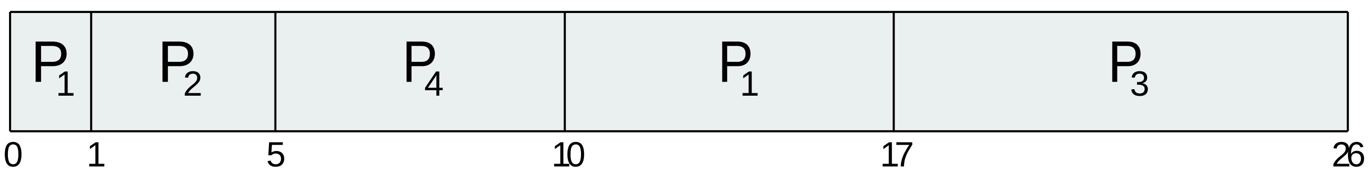
- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor

Example of Shortest-remaining-time-first (SRTF)

- 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

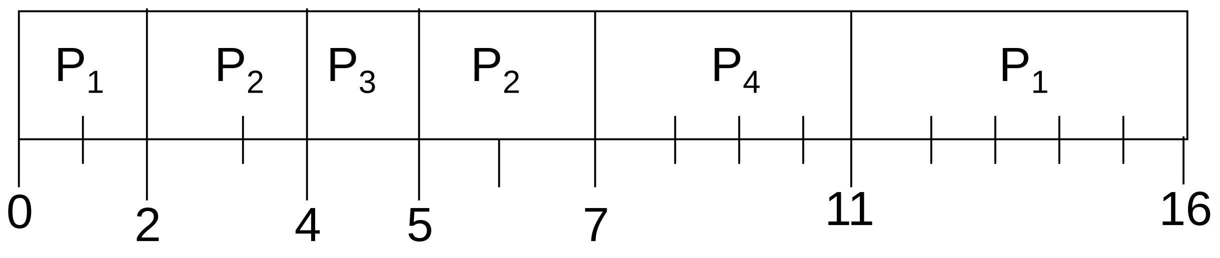
Example of SRTF

- Find average turn around time and average waiting time of the following scenario using SRTF scheduling algorithm.

Process	Arrival Time	Burst Time
1	0	7
2	2	4
3	4	1
4	5	4

Example of SRTF (Cont.)

Process	Arrival Time	Burst Time
1	0	7
2	2	4
3	4	1
4	5	4



Example of SRTF (Cont.)

Process	Arrival Time (A.T.)	Burst Time (B.T.)	Completion Time (C.T.)	Turn Around Time (T.A.T) TAT=CT-AT	Waiting Time (W.T.) WT=TAT-BT
1	0	7	16	16-0=16	16-7=9
2	2	4	7	7-2=5	5-4=1
3	4	1	5	5-4=1	1-1=0
4	5	4	11	11-5=6	6-4=2

- Average turn around time = (16+5+1+6)/4 = 28/4 = 7
- Average waiting time = (9+1+0+2)/4 = 12/4 = 3

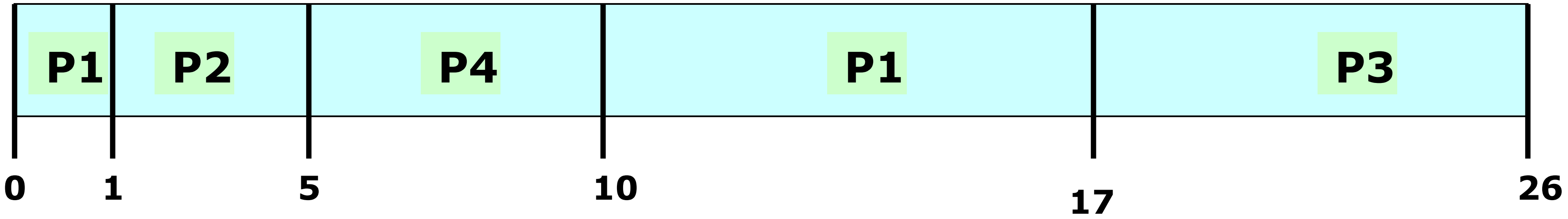
Example of SRTF (Cont.)

- Example: Find average turn around time and average waiting time of the following scenario using SRTF scheduling algorithm

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

Example of SRTF (Cont.)

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



Example of SRTF (Cont.)

Process	Arrival Time (A.T.)	Burst Time (B.T.)	Completion Time (C.T.)	Turn Around Time (T.A.T) TAT=CT-AT	Waiting Time (W.T.) WT=TAT-BT
1	0	8	17	17-0=17	17-8=9
2	1	4	5	5-1=4	4-4=0
3	2	9	26	26-2=24	24-9=15
4	3	5	10	10-3=7	7-5=2

- Average turn around time = (17+4+24+7)/4 = 52/4 = 13
- Average waiting time = (9+0+15+2)/4 = 26/4 = 6.5

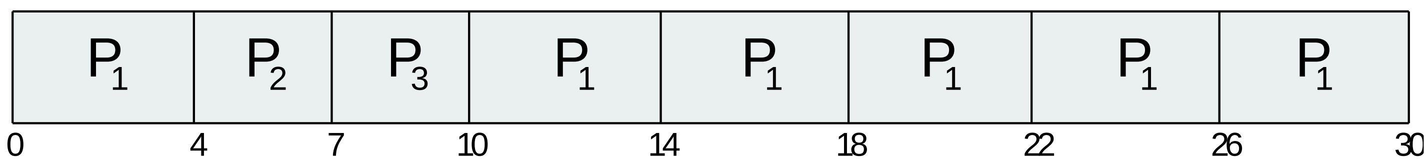
Round Robin (RR)

- 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
  - *q* large $\Rightarrow$ FIFO
  - *q* small $\Rightarrow$ *q* must be large with respect to context switch, otherwise overhead is too high

Example of RR with Time Quantum = 4

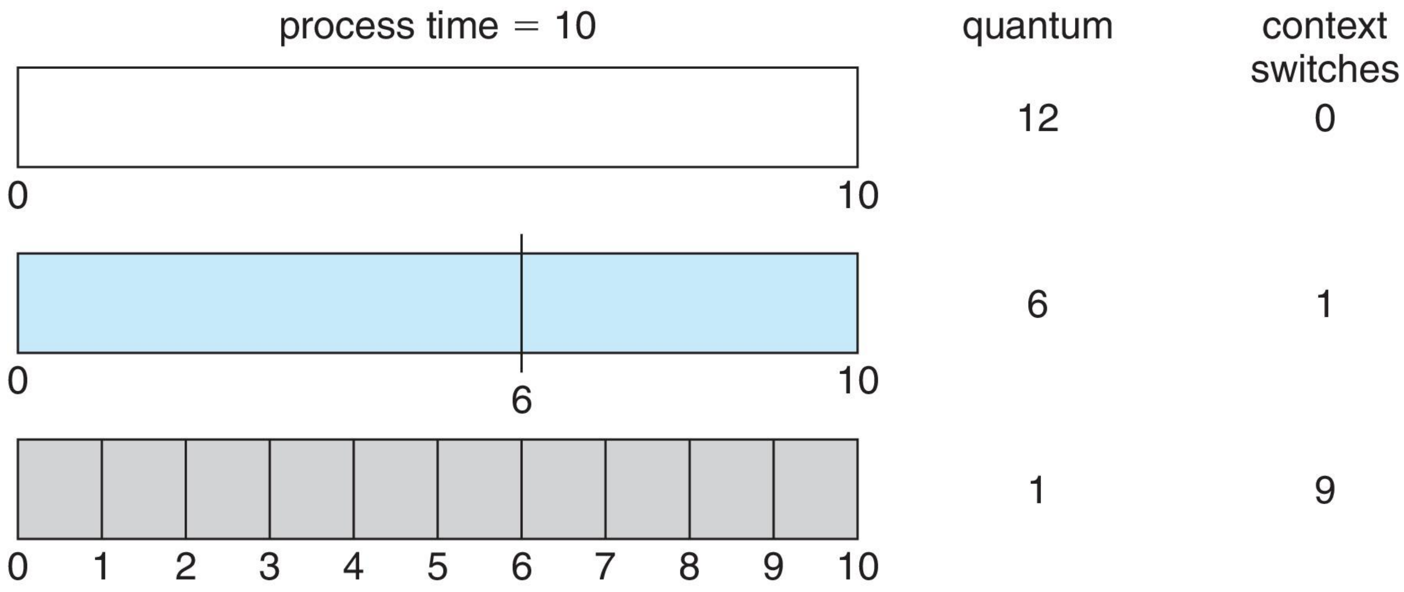
Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:

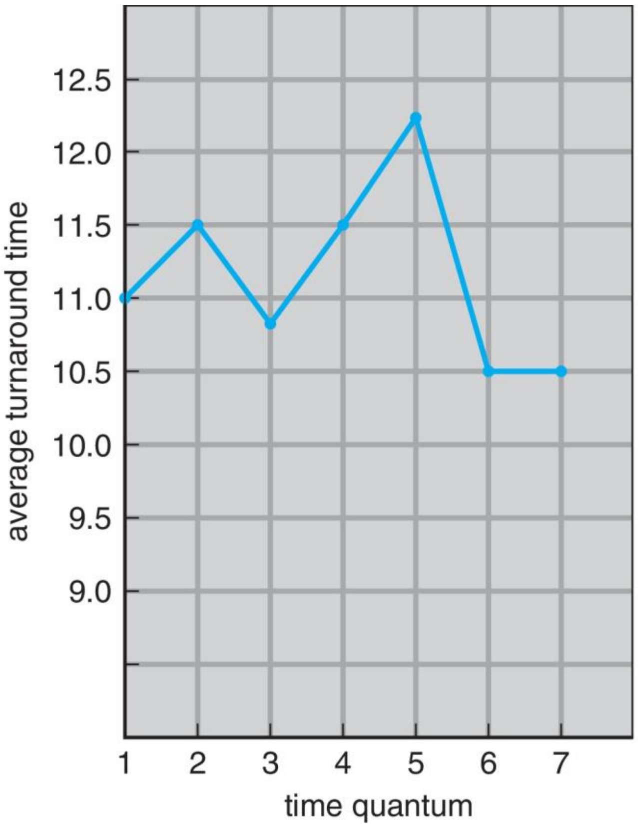


- Typically, higher average turnaround than SJF, but better ***response***
- q should be large compared to context switch time
  - q usually 10 milliseconds to 100 milliseconds,
  - Context switch < 10 microseconds

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

80% of CPU bursts
should be shorter than q

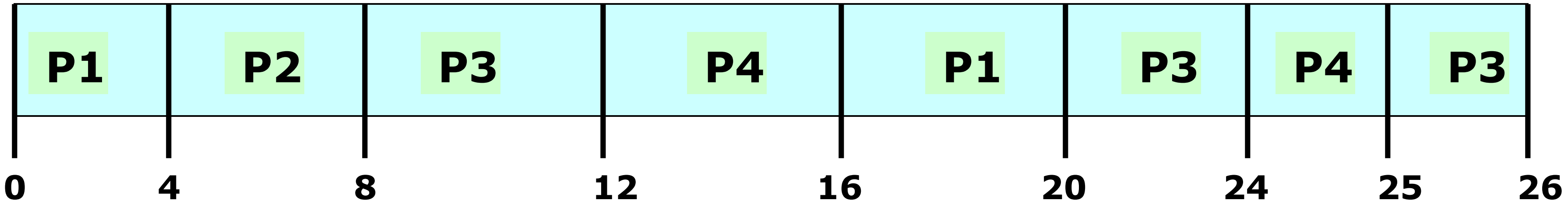
RR Scheduling Example

- Example: Find average turn around time and average waiting time of the following scenario using Round Robin (RR) scheduling algorithm. Assume quantum = 4 and no priority-based preemption is allowed.

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

RR Scheduling Example (Cont.)

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



RR Scheduling Example (Cont.)

Process	Arrival Time (A.T.)	Burst Time (B.T.)	Completion Time (C.T.)	Turn Around Time (T.A.T) TAT=CT-AT	Waiting Time (W.T.) WT=TAT-BT
1	0	8	20	20-0=20	20-8=12
2	1	4	8	8-1=7	7-4=3
3	2	9	26	26-2=24	24-9=15
4	3	5	25	25-3=22	22-5=17

- Average turn around time = (20+7+24+22)/4 = 73/4 = 18.25
- Average waiting time = (12+3+15+17)/4 = 47/4 = 11.75

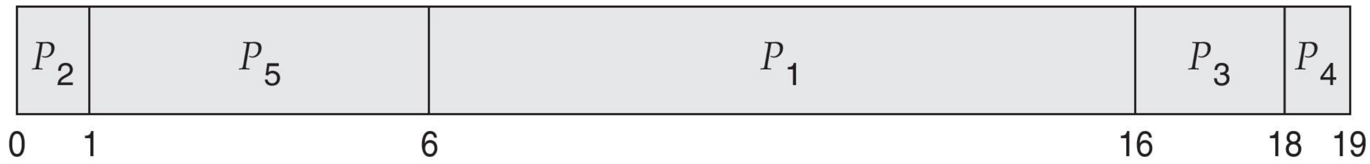
Priority Scheduling

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

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



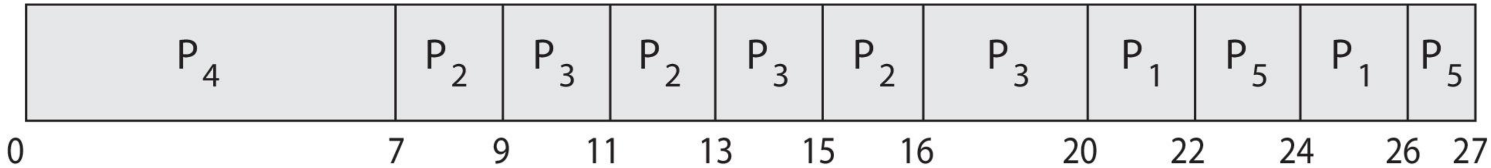
- Average waiting time = 8.2

Priority Scheduling w/ Round-Robin

Process	Burst Time	Priority
P_1	4	3
P_2	5	2
P_3	8	2
P_4	7	1
P_5	3	3

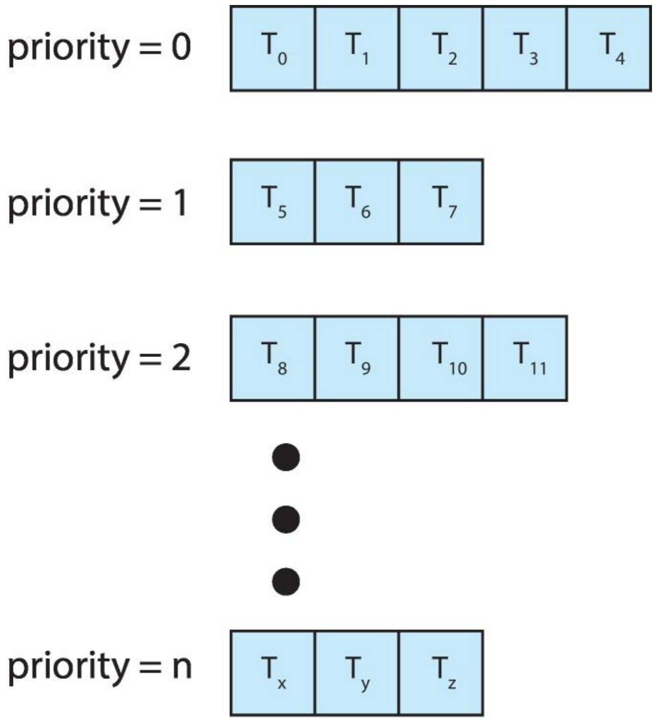
- Run the process with the highest priority. Processes with the same priority run round-robin

- Gantt Chart with time quantum = 2



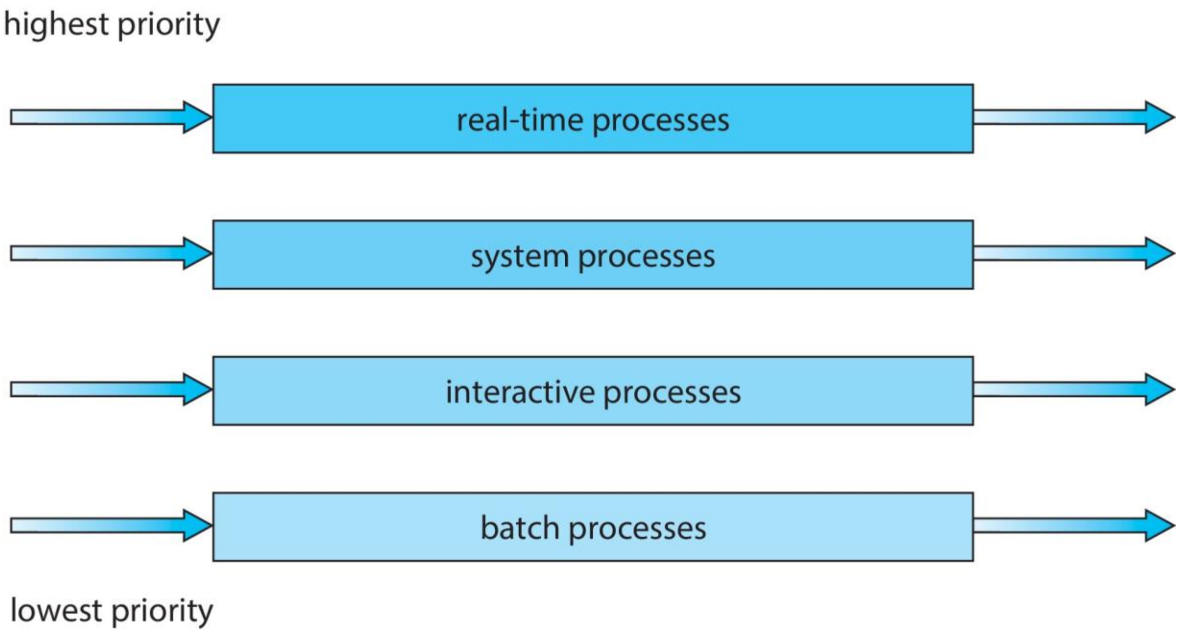
Multilevel Queue

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



Multilevel Queue

- Prioritization based upon process type

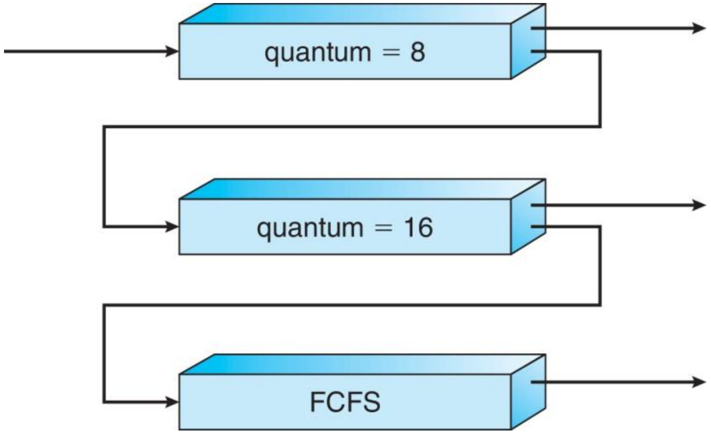


Multilevel Feedback Queue

- A process can move between the various queues.
- 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
- Aging can be implemented using multilevel feedback queue

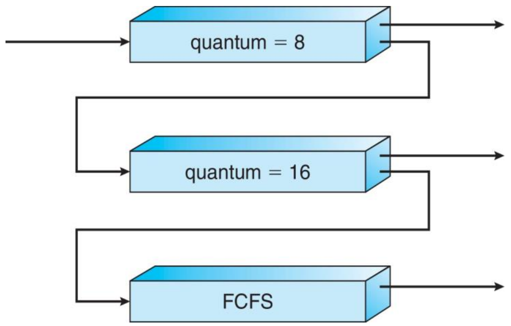
Example of Multilevel Feedback Queue

- Three queues:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS
- 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



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



Some more scheduling algorithms

- Longest Job First (LJF) scheduling - non-preemptive
- Longest Remaining Time First (LRTF) scheduling – preemptive
- Highest Response Ratio Next (HRRN) scheduling – non-preemptive
- Guaranteed scheduling
- Fair-share scheduling
- Lottery scheduling

References

- Operating Systems Concepts by Silberschatz, Galvin, and Gagne
- Modern Operating Systems by Tanenbaum.