Module3_CPU_Scheduling

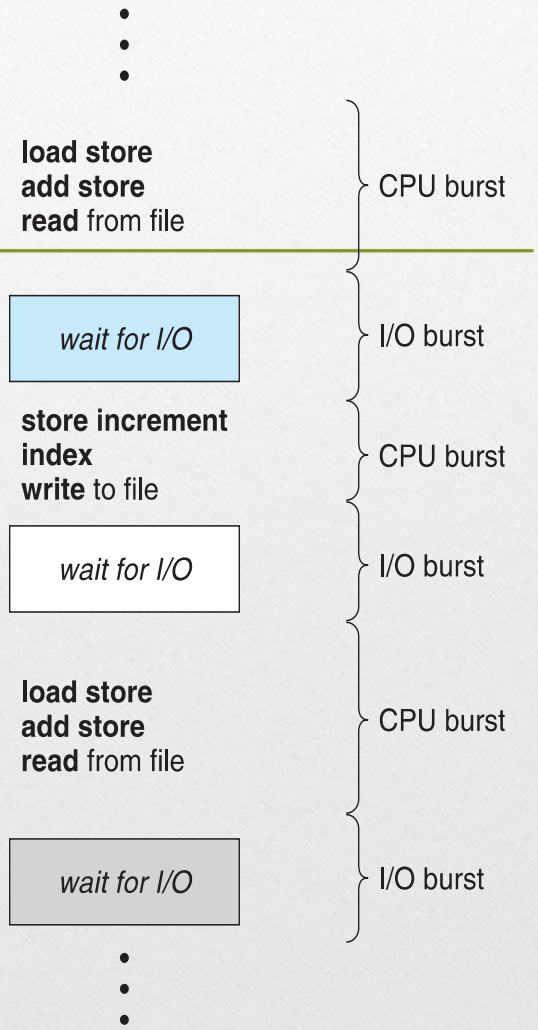
Reference: “OPERATING SYSTEM CONCEPTS”, ABRAHAM SILBERSCHATZ, PETER BAER GALVIN, GREG GAGNE , Wiley publications.

Objectives

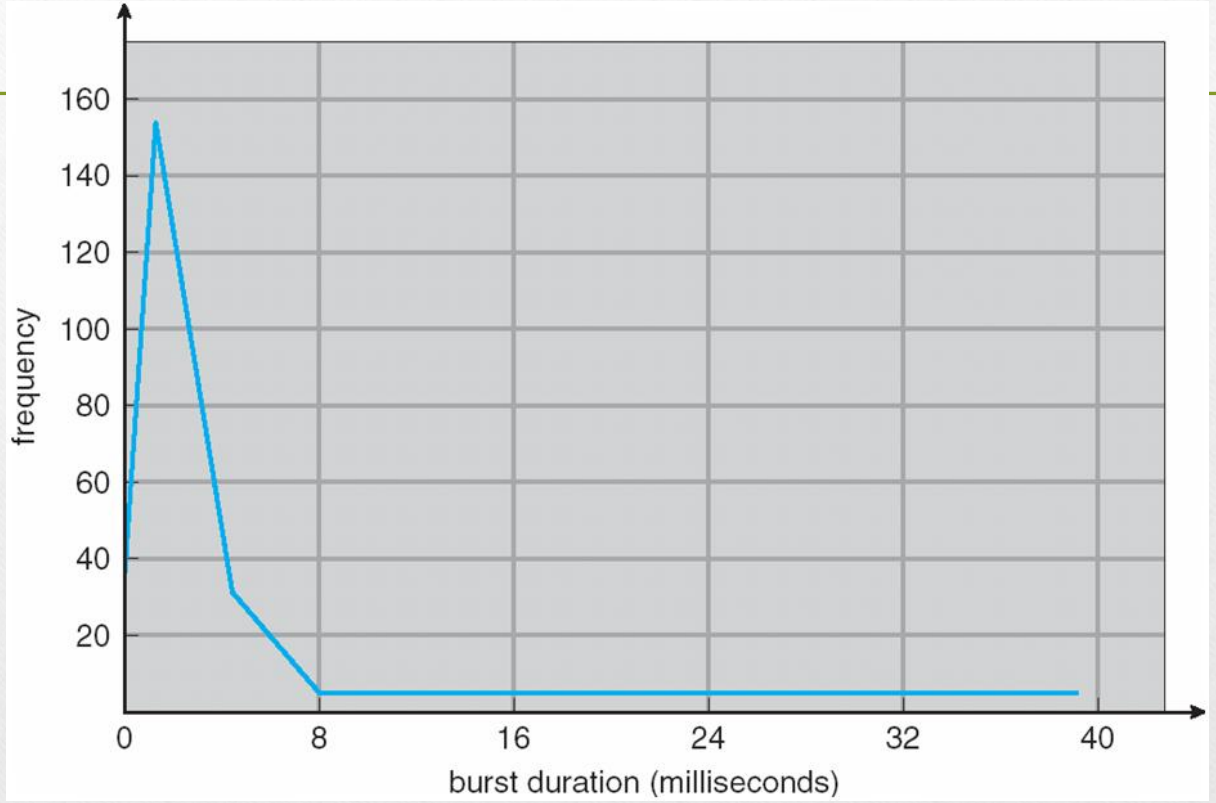
- To introduce CPU scheduling, which is the basis for multiprogrammed operating systems.
- To describe various CPU-scheduling algorithms.
- To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system.

Basic Concepts

- Maximum CPU utilization is 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 Burst Time: The amount of time a process executes before it goes to wait state

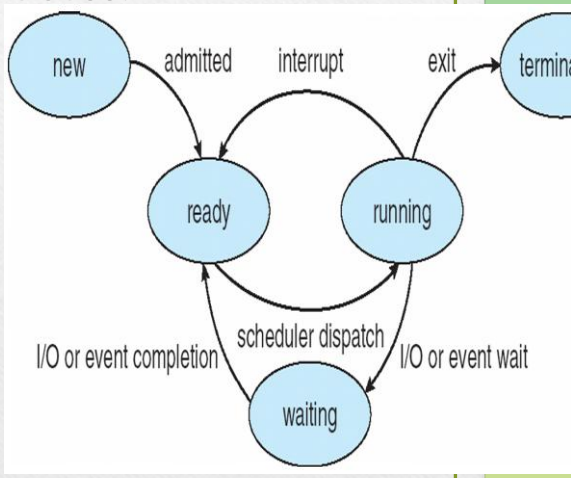


Histogram of CPU-burst Times



CPU Scheduler

- **Short-term scheduler** selects from among the processes in ready queue, and allocates the CPU 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
- 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



Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term 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 taken by the 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
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time taken from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Scheduling Algorithm Optimization Criteria

- Maximize CPU utilization
- Maximize throughput
- Minimize turnaround time
- Minimize waiting time
- Minimize response time

First- Come, First-Served (FCFS) Scheduling

Example 1: Consider the following processes with their burst time. Use FCFS to schedule the processes and compute the average waiting time.

Process ID	Burst Time
P_1	24
P_2	3
P_3	3

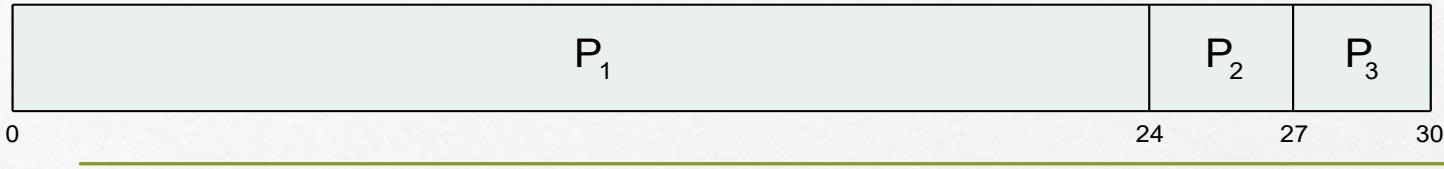
- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:

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

First- Come, First-Served (FCFS) Scheduling

Example 1: Continued

The Gantt Chart for the schedule is:



Process ID	Burst Time	Completion Time	TurnAround Time	Wait Time
P1	24	24	24	0
P2	3	27	27	24
P3	3	30	30	27

- Turnaround Time = Completion Time - Arrival Time
- Waiting Time = Turn Around Time - Burst Time

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

FCFS Scheduling (Cont.)

Example 1: Continued

Suppose that the **processes arrive in the order: P_2 , P_3 , P_1**

Process ID	Burst Time
P_1	24
P_2	3
P_3	3

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
- Performance of FCFS is dynamic
  - **Convoy effect** - short process behind long process
    - Consider one CPU-bound and many I/O-bound processes

Process ID	Burst Time	Completion Time	TurnAround Time	Wait Time
P1	24	30	30	6
P2	3	3	3	0
P3	3	6	6	3

FCFS Scheduling (Cont.)

Convoy Effect

- FCFS algorithm is non-preemptive, that is, once CPU time has been allocated to a process, other processes can get CPU time only after the current process has finished. This property of FCFS scheduling leads to the situation called Convoy Effect.
- Suppose there is one CPU intensive (large burst time) process in the ready queue, and several other processes with relatively less burst times but are Input/Output (I/O) bound (Need I/O operations frequently).
  - The I/O bound processes are first allocated CPU time. As they are less CPU intensive, they quickly get executed and goto I/O queues.
  - Now, the CPU intensive process is allocated CPU time. As its burst time is high, it takes time to complete.
  - While the CPU intensive process is being executed, the I/O bound processes complete their I/O operations and are moved back to ready queue.
  - However, the I/O bound processes are made to wait as the CPU intensive process still hasn't finished. This leads to I/O devices being idle.
  - When the CPU intensive process gets over, it is sent to the I/O queue so that it can access an I/O device.
  - Meanwhile, the I/O bound processes get their required CPU time and move back to I/O queue.
  - However, they are made to wait because the CPU intensive process is still accessing an I/O device. As a result, the CPU is sitting idle now.
  - Hence in Convoy Effect, one slow process slows down the performance of the entire set of processes, and leads to wastage of CPU time and other devices.

First- Come, First-Served (FCFS) Scheduling

- Example 2

Process	Arrival Time	Burst Time
P_1	0	7
P_2	0	4
P_3	0	1
P_4	0	4

- Gantt chart

Schedule: P_1 P_2 P_3 P_4

- Average waiting time: (0 + 7 + 11 + 12)/4 = 7.5

First- Come, First-Served (FCFS) Scheduling

Example 2: Continued

The Gantt Chart for the schedule is:

Process	Arrival Time	Burst Time
P_1	0	7
P_2	0	4
P_3	0	1
P_4	0	4

- Gantt chart

Schedule: P_1 P_2 P_3 P_4

- Average waiting time: (0 + 7 + 11 + 12)/4 = 7.5

Process ID	Burst Time	Completion Time	TurnAround Time	Wait Time
P1	7	7	7	0
P2	4	11	11	7
P3	1	12	12	11
P4	4	16	16	12

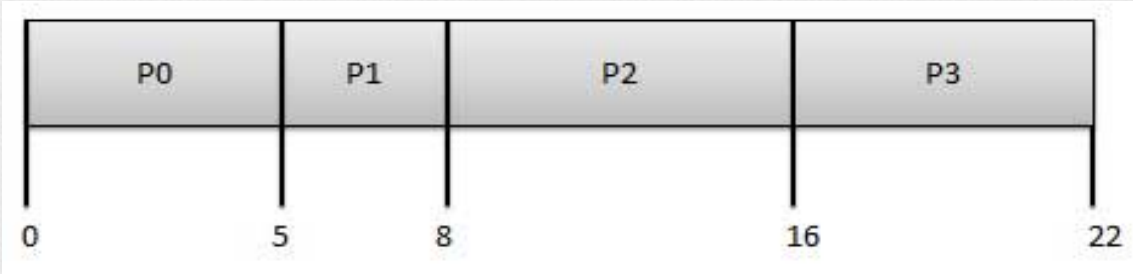
- Average waiting time: (0 +7+11+12)/4 = 7.5

First- Come, First-Served (FCFS) Scheduling

Example 3: Consider the following processes with their **burst time and arrival time**. Use FCFS to schedule the processes and compute the average waiting time.

Process ID	BurstTime	ArrivalTime
P0	5	0
P1	3	1
P2	8	2
P3	6	3

The Gantt Chart for the schedule is:



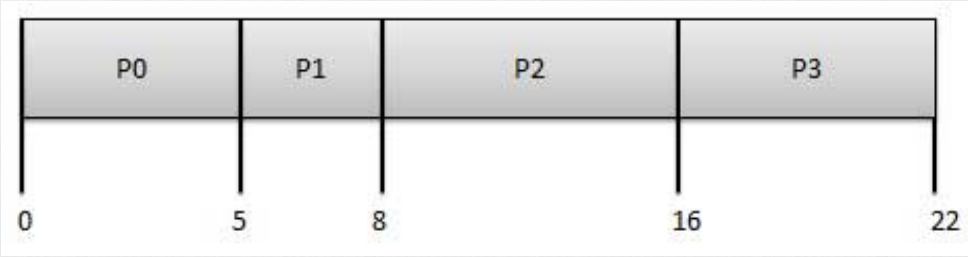
- Average Waiting Time: (0+4+6+13) / 4 = 5.75

First- Come, First-Served (FCFS) Scheduling

Example 3 (cont'd):

Process ID	Burst Time	Arrival Time	Completion Time	TurnAround Time	Wait Time
P0	5	0	5	5	0
P1	3	1	8	7	4
P2	8	2	16	14	6
P3	6	3	22	19	13

The Gantt Chart for the schedule is:



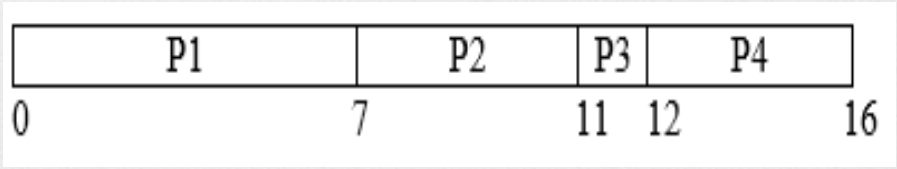
- Average Waiting Time: (0+4+6+13) / 4 = 5.75

First- Come, First-Served (FCFS) Scheduling

Example 4: Consider the following processes with their burst time and arrival time. Use FCFS to schedule the processes and compute the average waiting time.

Process ID	Burst Time	Arrival Time
P1	7	0
P2	4	2
P3	1	4
P4	4	5

The Gantt Chart for the schedule is:



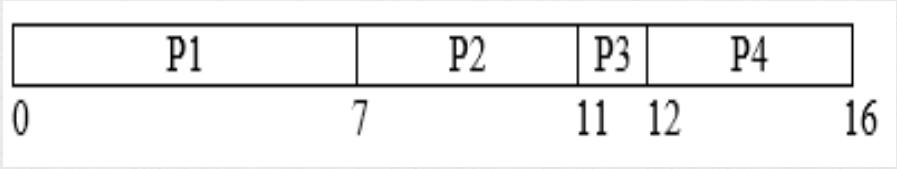
- Average waiting time = (0 + 5 + 7 + 7)/4 = 4

First- Come, First-Served (FCFS) Scheduling

Example 4: (continued)

Process ID	Burst Time	Arrival Time	Completion Time	TurnAround Time	Wait Time
P1	7	0	7	7	0
P2	4	2	11	9	5
P3	1	4	12	8	7
P4	4	5	16	11	7

The Gantt Chart for the schedule is:



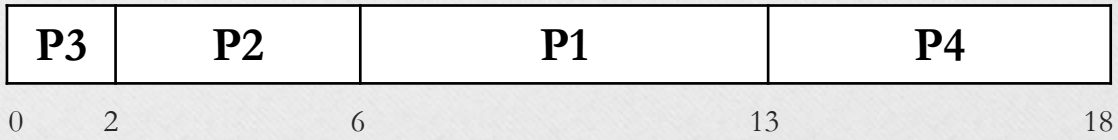
- Average waiting time = (0 + 5 + 7 + 7)/4 = 4.75

First- Come, First-Served (FCFS) Scheduling

Example 5: Consider the following processes with their burst time and arrival time. Use FCFS to schedule the processes and compute the average waiting time.

Process ID	Burst Time	Arrival Time
P1	7	4
P2	4	2
P3	2	0
P4	5	5

The Gantt Chart for the schedule is:



- Average waiting time = (2+0+0+8)/4 =2.5

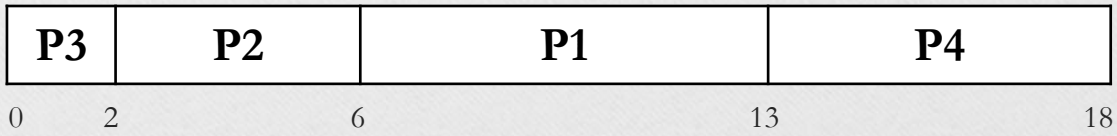
First- Come, First-Served (FCFS) Scheduling

Example 5: (continued)

Consider the following processes with their burst time and arrival time. Use FCFS to schedule the processes and compute the average waiting time.

Process ID	Burst Time	Arrival Time	TurnAround Time	Wait Time
P1	7	4	9	2
P2	4	2	4	0
P3	2	0	2	0
P4	5	5	13	8

The Gantt Chart for the schedule is:



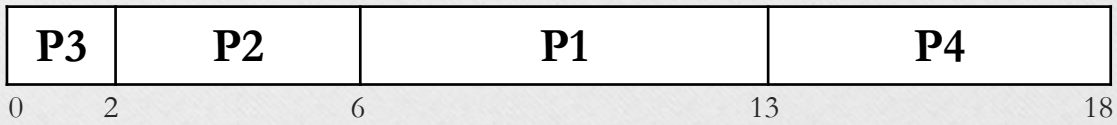
Average waiting time = (2+0+0+8)/4 =2.5

First- Come, First-Served (FCFS) Scheduling

Example 6: Consider the following processes with their burst time and arrival time. Use FCFS to schedule the processes and compute the average waiting time.

Process ID	Burst Time	Arrival Time
P1	7.2	4.6
P2	4.5	2.4
P3	2.6	0.6
P4	5.7	5.3

The Gantt Chart for the schedule is:



Average waiting time = (2+0+0+8)/4 =2.5

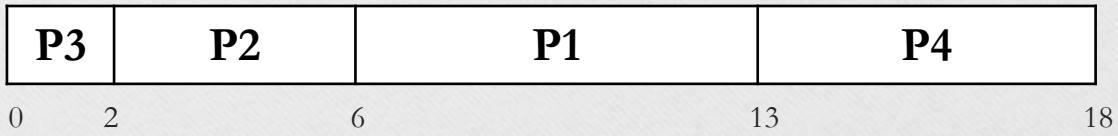
First- Come, First-Served (FCFS) Scheduling

Example 6: (continued)

Consider the following processes with their burst time and arrival time. Use FCFS to schedule the processes and compute the average waiting time.

Process ID	Burst Time	Arrival Time	TurnAround Time	Wait Time
P1	7	4	9	2
P2	4	2	4	0
P3	2	0	2	0
P4	5	5	13	8

The Gantt Chart for the schedule is:



Average waiting time = (2+0+0+8)/4 =2.5

Shortest-Job-First (SJF) Scheduling

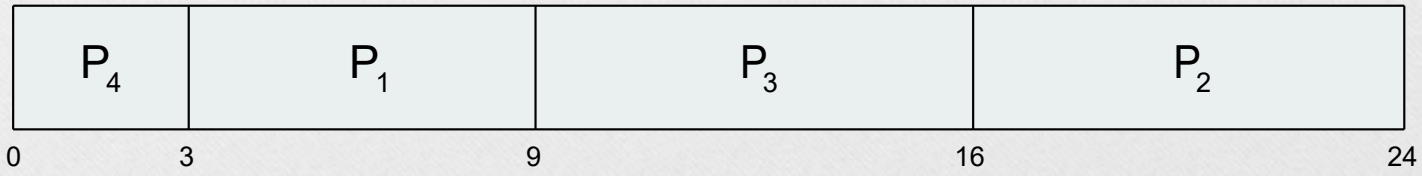
- Associate with each process the length of its next CPU burst
  - Use these lengths , schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is in knowing the length of the next CPU request
  - Could ask the user

SJF – Example 1

Example 1: Consider the following processes with their burst time. Use SJF to schedule the processes and compute the average waiting time.

Process ID	Burst Time
P1	6
P2	8
P3	7
P4	3

- SJF scheduling chart



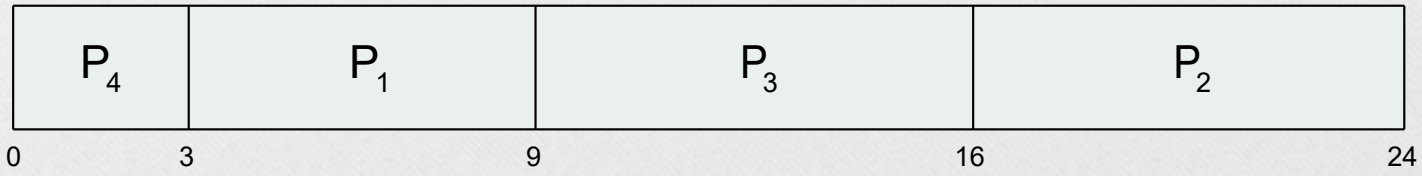
- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

SJF – Example 1 (cont'd)

- Scheduling Table

Process ID	Burst Time	Completion Time	TurnAround Time	Wait Time
P1	6	9	9	3
P2	8	24	24	16
P3	7	16	16	9
P4	3	3	3	0

- SJF scheduling chart



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

SJF Scheduling - Example

Example 2: Consider the following processes with their burst time and arrival time. Use SJF to schedule the processes and compute the average waiting time.

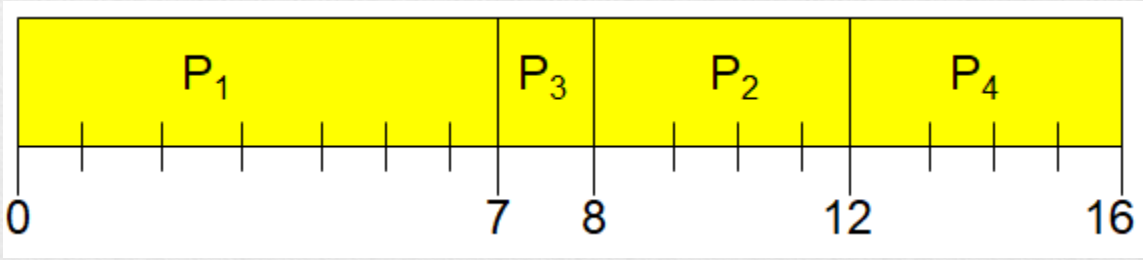
Process ID	Burst Time	Arrival Time
P1	7	0
P2	4	2
P3	1	4
P4	4	5

SJF Scheduling - Example

Example 2 (continued):Consider the following processes with their burst time and arrival time. Use SJF to schedule the processes and compute the average waiting time.

Process ID	Burst Time	Arrival Time
P1	7	0
P2	4	2
P3	1	4
P4	4	5

The Gantt Chart for the schedule is:



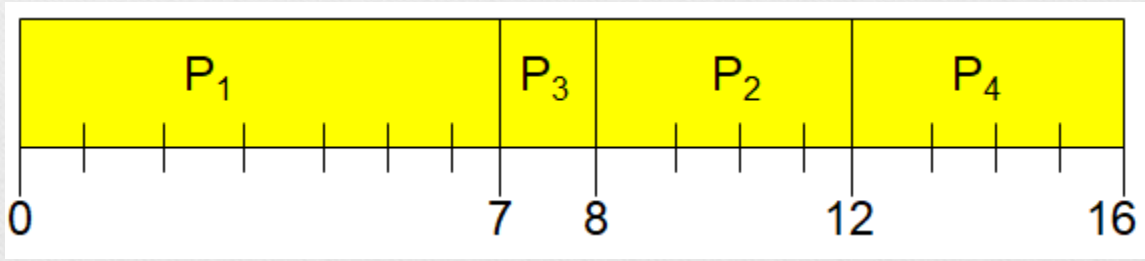
- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

SJF Scheduling - Example

Example 2: (continued)

Process ID	Burst Time	Arrival Time	Completion Time	TurnAround Time	Wait Time
P1	7	0	7	7	0
P2	4	2	12	10	6
P3	1	4	8	4	3
P4	4	5	16	11	7

The Gantt Chart for the schedule is:



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

Determining Length of Next CPU Burst

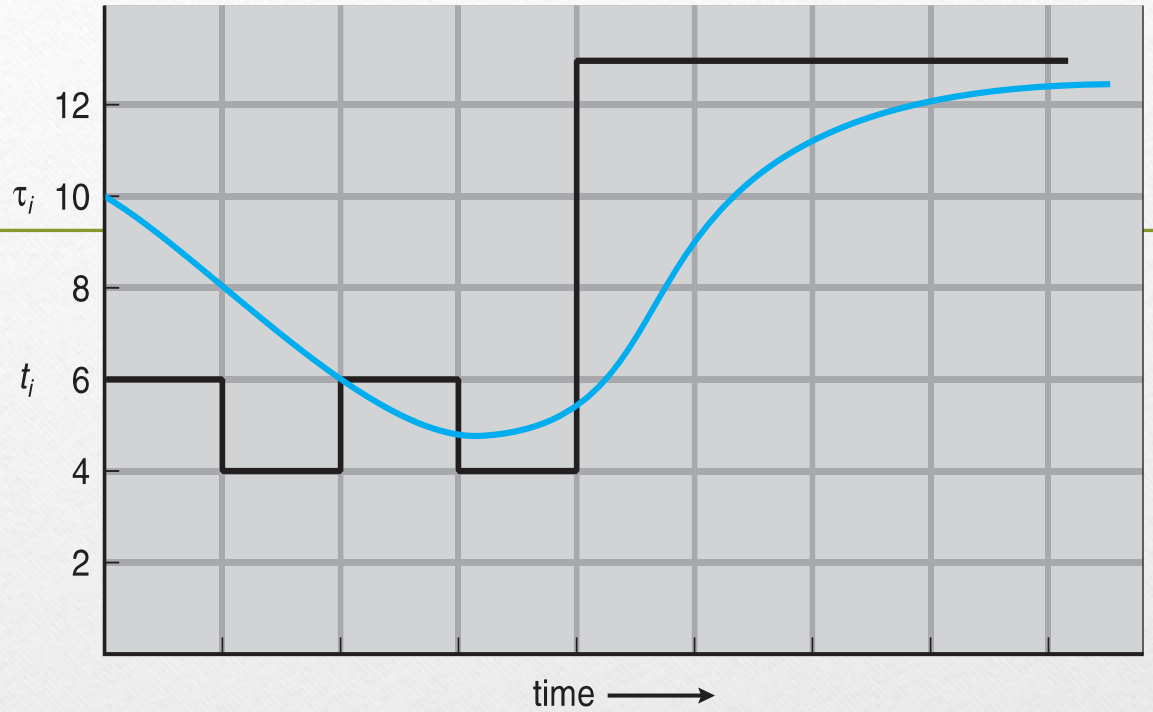
- Can only estimate the length – should be similar to the previous one
  - Then pick process with shortest predicted next CPU burst

- 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. τ_{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 to ½
- Preemptive version of SJF is called **Shortest-Remaining-Time-First (SRTF)**

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	...

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

Examples of Exponential Averaging

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

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

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\alpha\, t_{n-1} + \ldots$$
$$+ (1-\alpha)^j \alpha\, t_{n-j} + \ldots$$
$$+ (1-\alpha)^{n+1} \tau_0$$

- 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

- Preemptive version of SJF is called **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

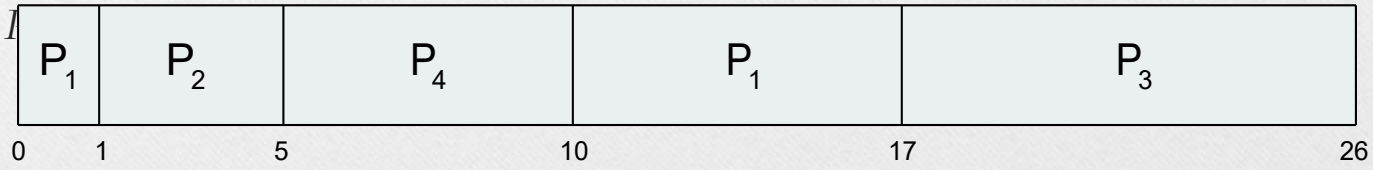
Example of Shortest-remaining-time-first

- Preemptive version of SJF is called **Shortest-Remaining-Time-First (SRTF)**

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

Process	Arrival Time	Burst Time	Completion Time	Wait Time	TAT
P_1	0	8	17	9	17
P_2	1	4	5	0	4
P_3	2	9	26	15	24
P_4	3	5	10	2	7

- *I*



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

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

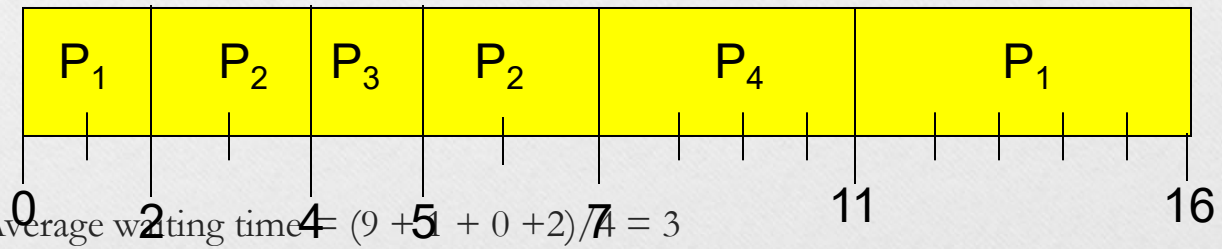
- SJF (preemptive)

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

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (preemptive)



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

Example of Preemptive SJF

- Example

Process ID	Burst Time	Arrival Time
P1	7	0
P2	4	2
P3	1	4
P4	4	5

Process ID	Burst Time	Arrival Time	Turn Around Time	Wait Time
P1	7	0	16	9
P2	4	2	5	1
P3	1	4	1	0
P4	4	5	6	2

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
- Priorities can be defined either internally or externally
- Problem ≡ **Starvation** **(indefinite blocking)** – low priority processes may never execute
- Solution ≡ **Aging** – as time progresses increase the priority of the process

Priority Scheduling - Example

Example 1: Consider the following processes with their burst time and priority. Use priority to schedule the processes and compute the average waiting time.

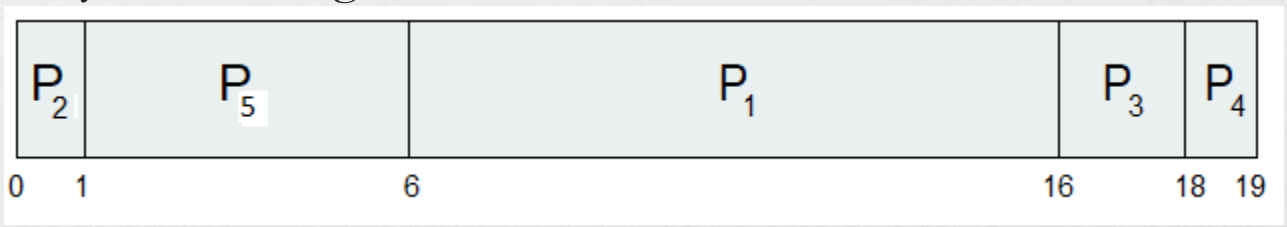
Process ID	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Priority Scheduling - Example

- Example 1 (cont'd) :

Process ID	Burst Time	Priority	TurnAround Time	Wait Time
P1	10	3	16	6
P2	1	1	1	0
P3	2	4	18	16
P4	1	5	19	18
P5	5	2	6	1

- Priority scheduling Gantt Chart:



- Average waiting time = (6+0+16+18+1)/5 = 8.2 msec

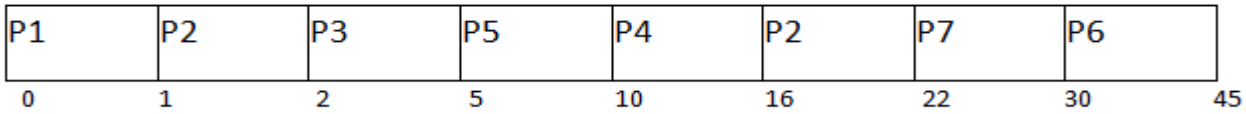
Priority Scheduling - Example

Example 2: Consider the following processes with their burst time and priority. Use priority to schedule the processes and compute the average waiting time.

Process ID	Priority	Arrival Time	Burst Time
P1	2	0	1
P2	6	1	7
P3	3	2	3
P4	5	3	6
P5	4	4	5
P6	10	5	15
P7	9	6	8

Priority Scheduling - Example

- Example 2 (continued): Gantt Chart

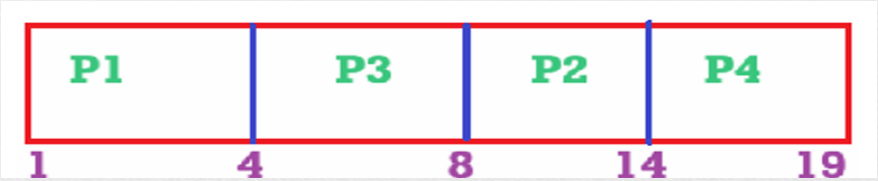


Process ID	Priority	Arrival Time	Burst Time	Completion Time	Turn Around Time	Wait Time
P1	2	0	1	1	1	0
P2	6	1	7	22	21	14
P3	3	2	3	5	3	0
P4	5	3	6	16	13	7
P5	4	4	5	10	6	1
P6	10	5	15	45	40	25
P7	9	6	8	30	24	16

- Average Waiting Time = (0+14+0+7+1+25+16)/7 = 63/7 = 9 units

- Find the average Turn Around Time and Waiting Time of following processes using Non-Preemptive priority scheduling process scheduling algorithm? Consider the following processes with their Arrival Time, Burst Time, and Priority.

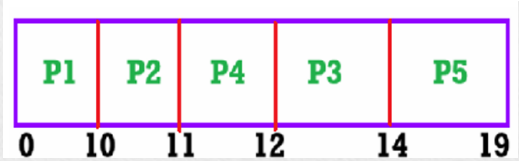
Process	Arrival Time	Burst Time	Priority
P1	1	3	3
P2	2	6	2
P3	3	4	1
P4	5	5	4



- P1P2P3P2P1P4

Find the average Turn Around Time and Waiting Time of following processes using Non-Preemptive SJT scheduling process scheduling algorithm? Consider the following processes with their Arrival Time, Burst Time, and Priority.

Process	Arrival Time	Burst Time
P1	0	10
P2	1	1
P3	2	2
P4	3	1
P5	4	5



P1P2P3P4P3P5P1

Round Robin (RR)

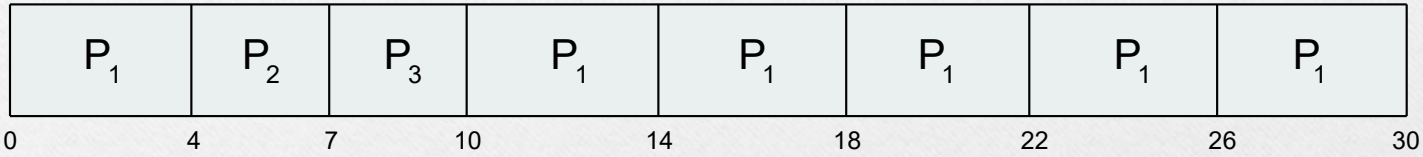
- Each process gets a small unit of CPU time (**time quantum** q), usually 10-100 milliseconds.
  - After this time has elapsed, the process is preempted and added to the end of the ready queue.

- 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 $(n\text{-}1)q$ 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:



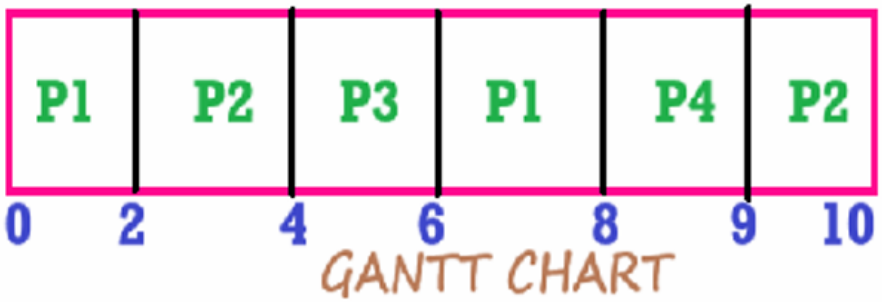
- Typically, higher average turnaround than SJF, but better ***response***
- q should be large compared to context switch time
- q is usually 10ms to 100ms, context switch < 10 usec

Round Robin Scheduling

- Find the average Turn Around Time and Waiting Time of following processes using RR(Round Robin) process scheduling algorithm? Consider the following processes with their Arrival Time, Burst Time (take Quantum Time = 2).

Process	Arrival Time	Burst Time
P1	0	4
P2	1	3
P3	2	2
P4	3	1

Round Robin Scheduling



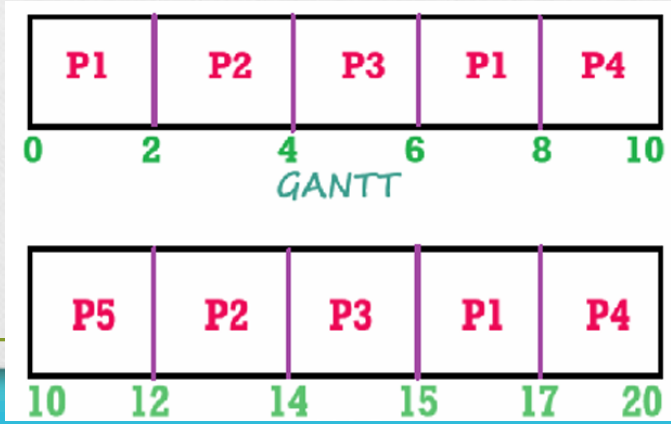
Process	Completion Time	Turnaround Time	Waiting Time
P1	8	8	4
P2	10	9	6
P3	6	4	2
P4	9	6	5

Average Turnaround Time = (8 + 9 + 4 + 6) / 4 = 6.75

Average Waiting Time = (4 + 6 + 2 + 5) / 4 = 4.25

- Find the average Turn Around Time and Waiting Time of following processes using RR(Round Robin) process scheduling algorithm? Consider the following processes with their Arrival Time, Burst Time (take Quantum Time = 2).

Process	Arrival Time	Burst Time
P1	0	6
P2	1	4
P3	2	3
P4	3	5
P5	4	2

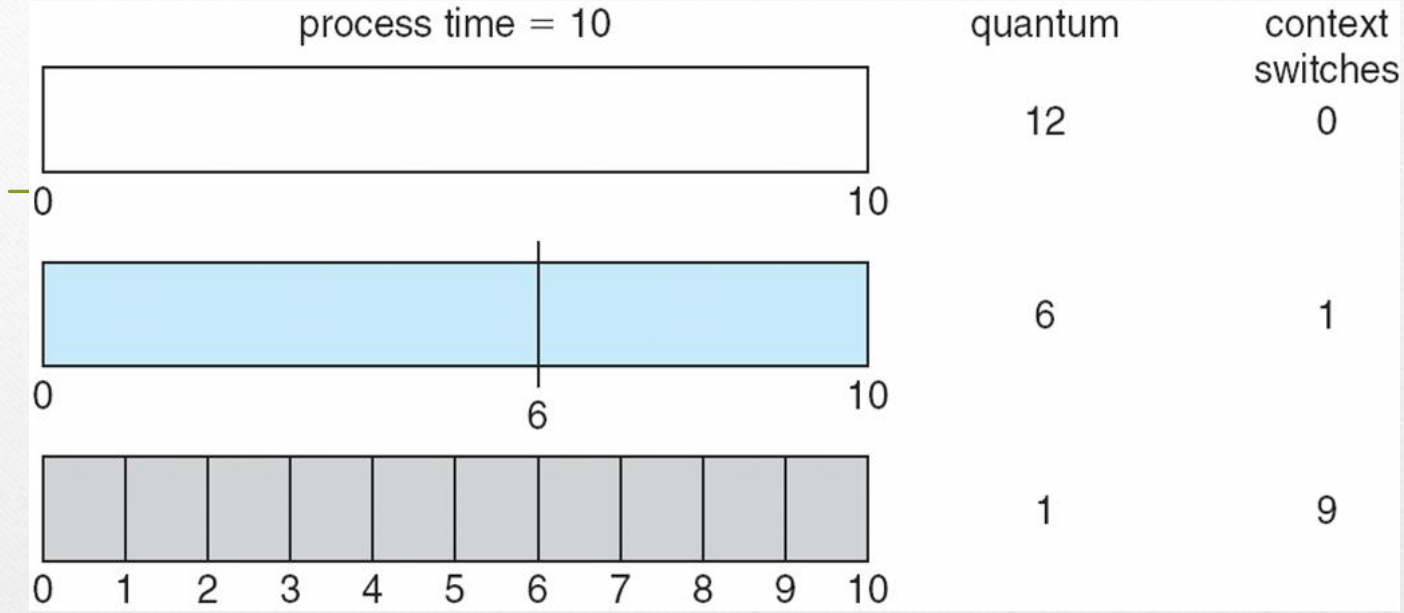


- Find the average Turn Around Time and Waiting Time of following processes using RR(Round Robin) process scheduling algorithm? Consider the following processes with their Arrival Time, Burst Time (take Quantum Time = 1).

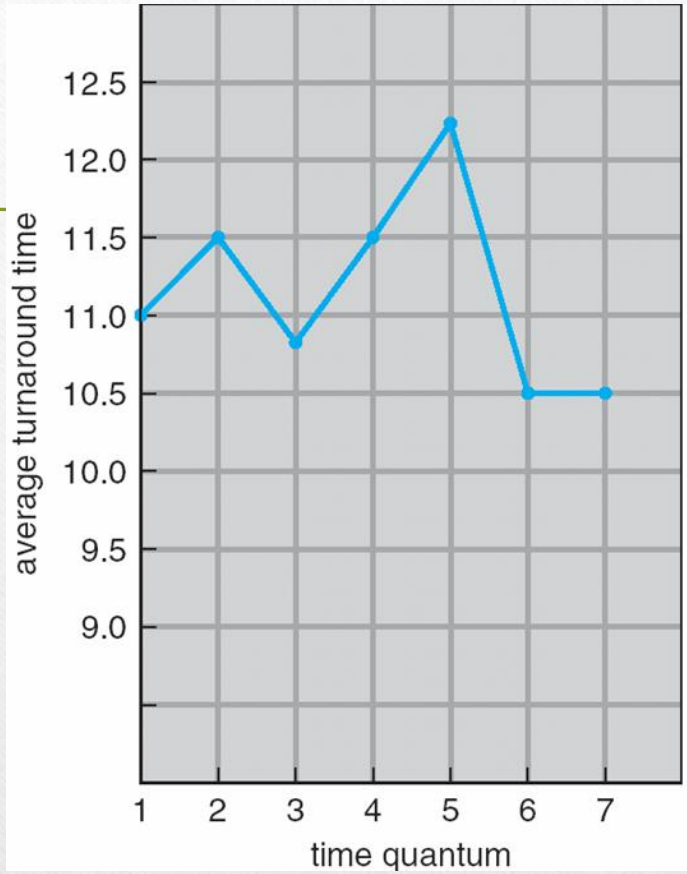
Process	Arrival Time	Burst Time
P1	5	4
P2	1	5
P3	0	6
P4	3	2
P5	4	3
P6	2	7

Process No.	Arrival Time	Burst Time		
		CPU Burst	I/O Burst	CPU Burst
P1	0	3	2	2
P2	0	2	4	1
P3	2	1	3	2
P4	5	2	2	1

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

Multilevel Queue

- Ready queue is partitioned into separate queues, eg:
  - **foreground** (interactive)
  - **background** (batch)

- Process is permanently in a given queue
- Each queue has its own scheduling algorithm:
  - foreground – RR
  - background – FCFS
- Scheduling must be done between the queues:
  - Fixed priority 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., 80% to foreground in RR
  - 20% to background in FCFS

Question 1 : find the average Turn Around Time and Waiting Time of following processes using MULTILEVEL QUEUE Scheduling (MLQ) algorithm? Consider the following processes with their Arrival Time, Burst Time. Question criteria :

(1) There are three queue Q1, Q2, Q3

(2) In Q1 (sjf, non-preemptive), Q2(Round robin, timestamp=2),Q3(FCFS)

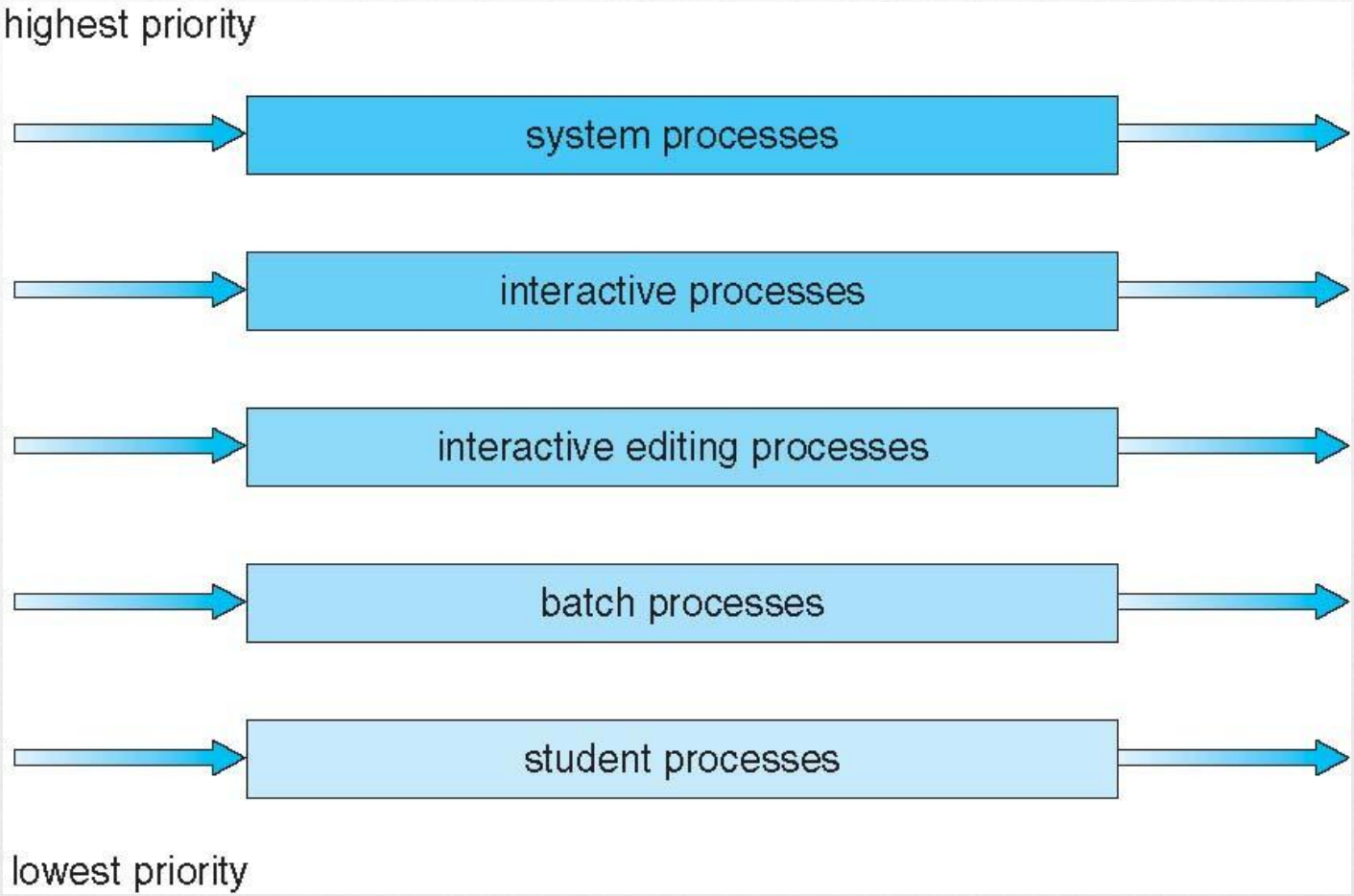
Process	Arrival Time	Burst Time	Queue name
P1	0	8	Q2
P2	1	6	Q1
P3	1	4	Q3
P4	2	3	Q2
P5	2	1	Q1
P6	3	4	Q2
P7	3	1	Q1
P8	4	2	Q2

Question 2: Consider following process, with the cpu burst time given in milliseconds

Process	Burst Time	Arrival Time	Queue name	Priority
P1	4	0.0	2	5
P2	2	2.0	1	----
P3	5	3.0	3	----
P4	1	4.0	2	10
P5	3	5.0	1	----
P6	6	6.0	2	2
P7	4	8	1	----
P8	2	10.0	2	1
P9	7	12.0	1	----
P10	3	12.0	2	1

The system has three queues, first queue is having the highest priority with FCFS scheduling, second queue is having the middle priority with priority scheduling (1 is highest and 100 is lowest priority) and, third queue is having the lowest priority with Round Robin scheduling(T.S = 3). Draw gantt charts to show execution using Multi Level Queue Scheduling (MLQ). Also determine the process turnAround time and Waiting time.

Multilevel Queue Scheduling



Multilevel Feedback Queue

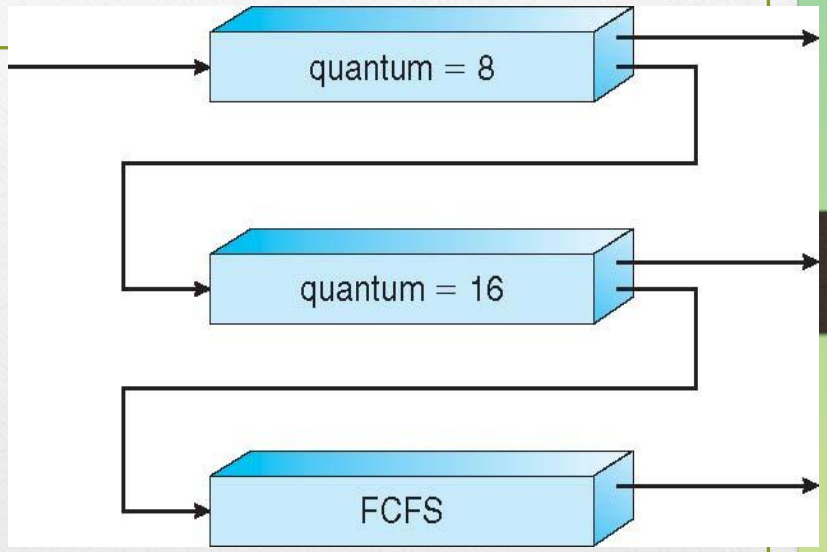
- A process can move between the various queues; aging can be implemented this way

---

- Multilevel-feedback-queue scheduler is 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:
  - Q_0 – RR with time quantum of 8 milliseconds
  - Q_1 – RR with time quantum of 16 milliseconds
  - Q_2 – FCFS



- Scheduling
  - A new job enters queue Q_0 which is served by FCFS
    - 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 by FCFS and receives 16 additional milliseconds
    - If it still does not complete, it is preempted and moved to queue Q_2

Question 1 : find the average Turn Around Time and Waiting Time of following processes using MULTILEVEL FEEDBACK QUEUE Scheduling (MLFQ) algorithm. Consider the following processes with their Arrival Time, Burst Time. Question criteria :

(1) There are four queue Q1, Q2, Q3, Q4

(2) Queue with their time stamp Q1(T.S=4), Q2(T.S=8), Q3(T.S=16), Q4(FCFS)

(in Q1, Q2, Q3, we use Round Robin and in Q4 we use first come first serve (FCFS) scheduling algorithms)

Process	Arrival Time	Burst Time
P1	1	6
P2	4	8
P3	5	12
P4	8	14
P5	12	6
P6	10	10
P7	8	8
P8	16	6

Consider following process, with the cpu burst time given in milliseconds. The system has three queues. first queue is having the T.S = 2, Second queue is having the T.S = 4 and, third queue is having the FCFS scheduling

Process	Burst Time	Arrival Time
P1	4	0.0
P2	7	1.0
P3	5	1.0
P4	6	2.0
P5	12	3.0
P6	9	5.0
P7	3	6.0
P8	10	8.0
P9	3	12.0
P10	2	16.0
P11	4	22.0
P12	2	24.0

Multiple-Processor Scheduling

- CPU scheduling is more complex when multiple CPUs are available
- **Homogeneous processors** within a multiprocessor
- **Asymmetric multiprocessing** – only one processor accesses the system data structures, alleviating the need for data sharing
- **Symmetric multiprocessing (SMP)** – each processor is self-scheduling, all processes are in common ready queue, or each has its own private queue of ready processes
  - Currently, most common
- **Processor affinity** – process has affinity for processor on which it is currently running
  - **soft affinity**
  - **hard affinity**
- **Soft Affinity**: The system tries to keep a process running on the same processor but does not guarantee it.
- **Hard Affinity**: The process specifies a subset of processors on which it may run
  - Variations including **processor sets**

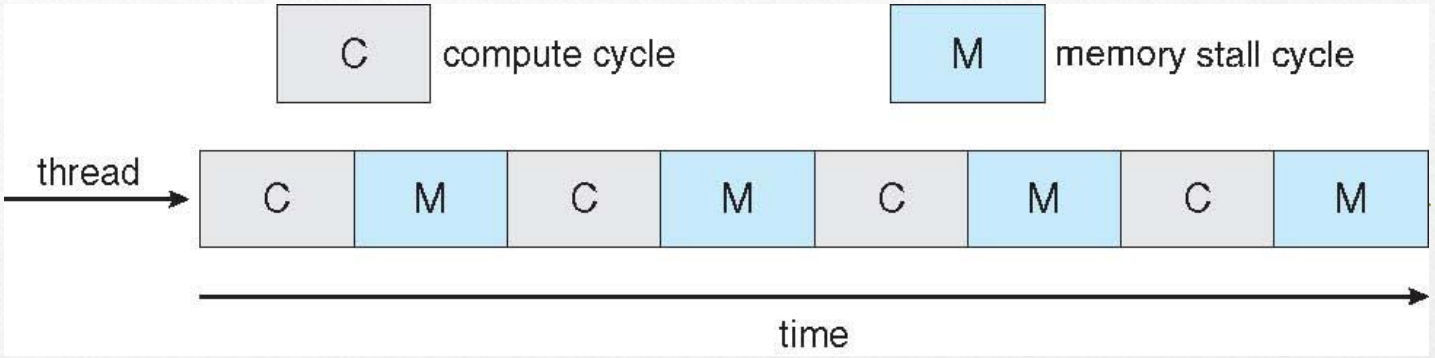
Multiple-Processor Scheduling – Load Balancing

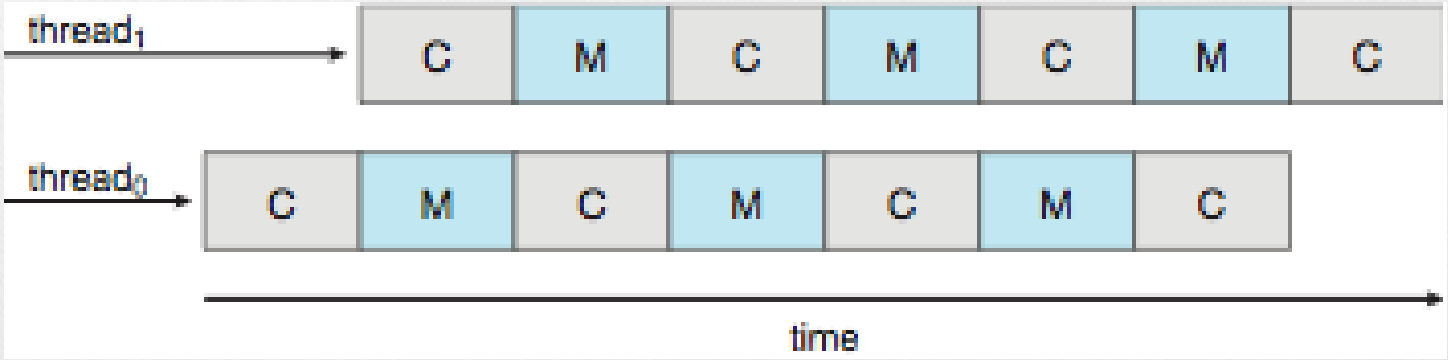
- In SMP, we need to keep all CPUs loaded for efficiency
- **Load balancing** attempts to keep workload evenly distributed
  - **Push migration** – periodic task checks load on each processor, and if found pushes task from overloaded CPU to other CPUs
  - **Pull migration** – idle processors pulls waiting task from busy processor

Multicore Processors

- Trend to place multiple processor cores on same physical chip
- Faster and consumes less power
- Multiple threads per core is also growing
  - Takes advantage of memory stall to make progress on another thread while memory retrieve happens

Multithreaded Multicore System





Windows Scheduling

- Windows uses priority-based preemptive scheduling
- Highest-priority thread runs next
- **Dispatcher** is scheduler
- Thread runs until (1) blocks, (2) uses time slice, (3) preempted by higher-priority thread
- Real-time threads can preempt non-real-time
- 32-level priority scheme
- **Variable class** is 1-15, **real-time class** is 16-31
- Priority 0 is memory-management thread
- Queue for each priority
- If no run-able thread, runs **idle thread**

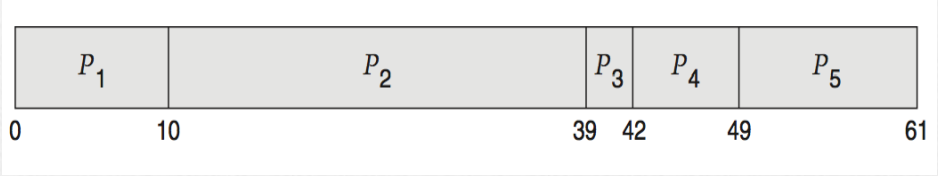
Algorithm Evaluation

- How to select CPU-scheduling algorithm for an OS?
- Determine criteria, then evaluate algorithms
- **Deterministic modeling**
  - Type of **analytic evaluation**
  - Takes a particular predetermined workload and defines the performance of each algorithm for that workload
- Consider 5 processes arriving at time 0:

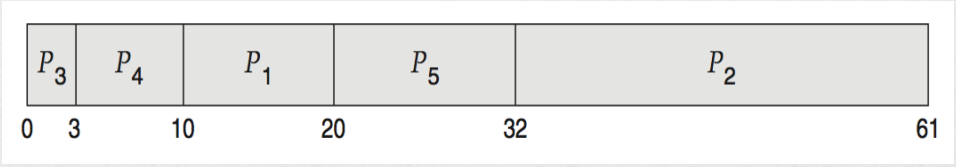
Process	Burst Time
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

Deterministic Evaluation

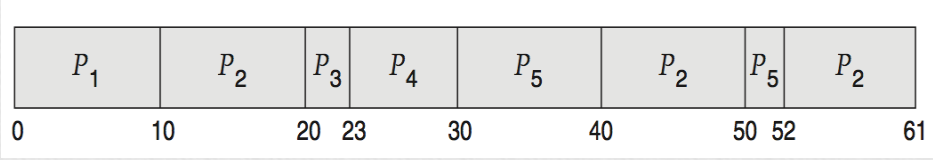
- For each algorithm, calculate minimum average waiting time
- Simple and fast, but requires exact numbers for input, applies only to those inputs
  - FCFS is 28ms:



  - Non-preemptive SFJ is 13ms:



  - RR is 23ms:



Queueing Models

- Describes the arrival of processes, and CPU and I/O bursts probabilistically
  - Commonly exponential, and described by mean
  - Computes average throughput, utilization, waiting time, etc
- Computer system described as network of servers, each with queue of waiting processes
  - Knowing arrival rates and service rates
  - Computes utilization, average queue length, average wait time, etc

Little's Formula

- n = average queue length
- W = average waiting time in queue
- λ = average arrival rate into queue
- Little's law – in steady state, processes leaving queue must equal processes arriving, thus:

  $n = \lambda \times W$

  - Valid for any scheduling algorithm and arrival distribution
- For example, if on an average 7 processes arrive per second, and normally 14 processes in queue, then average wait time per process = 2 seconds

Simulations

- Queueing models are limited
- **Simulations** are more accurate
  - Programmed model of computer system
  - Clock is a variable
  - Gather statistics indicating algorithm performance
  - Data to drive simulation gathered via
    - Random number generator according to probabilities
    - Distributions defined mathematically or empirically
    - Trace tapes record sequences of real events in real systems

Other References

For more practice problems

- https://www.gatevidyalay.com/round-robin-round-robin-scheduling-examples/