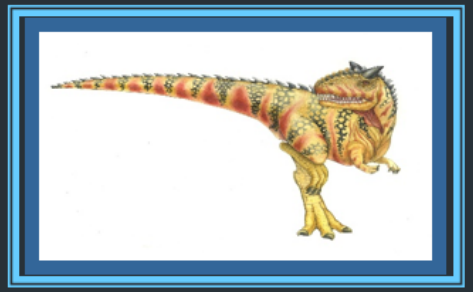
Module3- L1: Processes Scheduling



Dr. Rishikeshan C A
SCOPE, VIT Chennai

Outline

- Scheduling queues
- Representation of Process Scheduling
- Schedulers
- Scheduling Criteria
- Scheduling Algorithms

Objectives

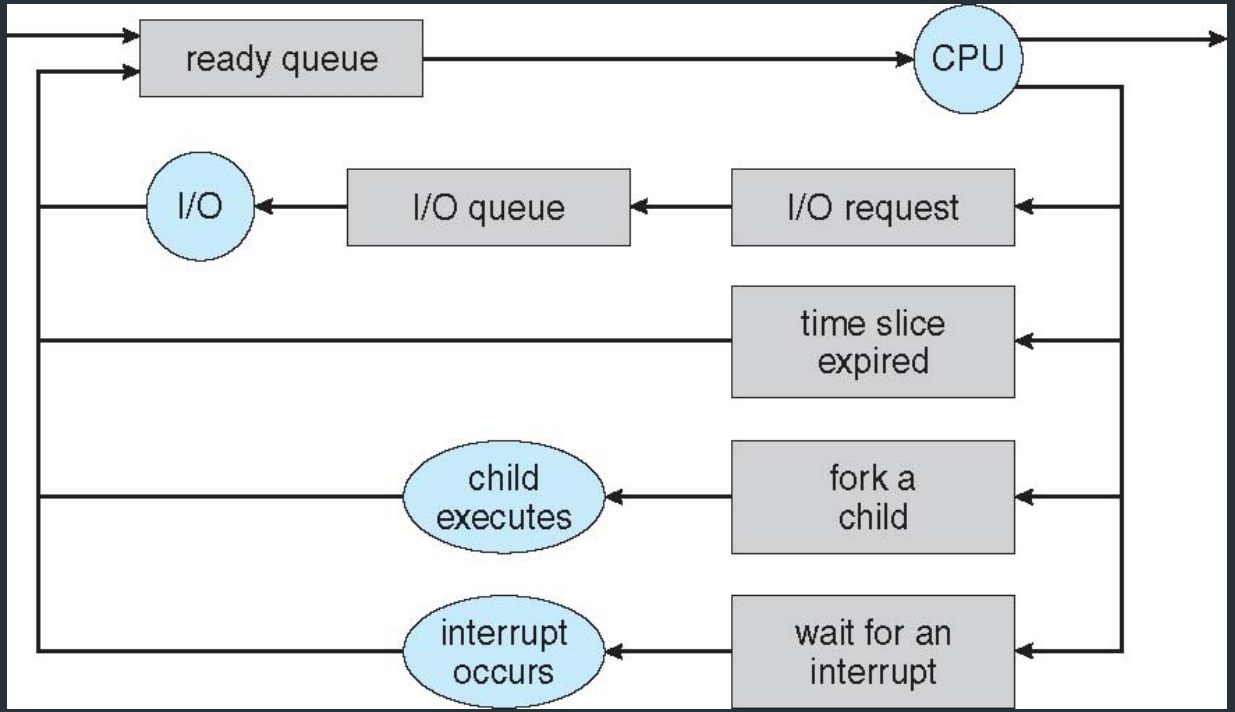
- To introduce CPU scheduling, which is the basis for multiprogrammed operating systems
- To describe various CPU-scheduling algorithms
- To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system
- To examine the scheduling algorithms of several operating systems

Process Scheduling

- Maximize CPU use, quickly switch processes onto CPU for time sharing
- **Process scheduler** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device
  - Processes migrate among the various queues

Representation of Process Scheduling

- **Queueing diagram** represents queues, resources, flows

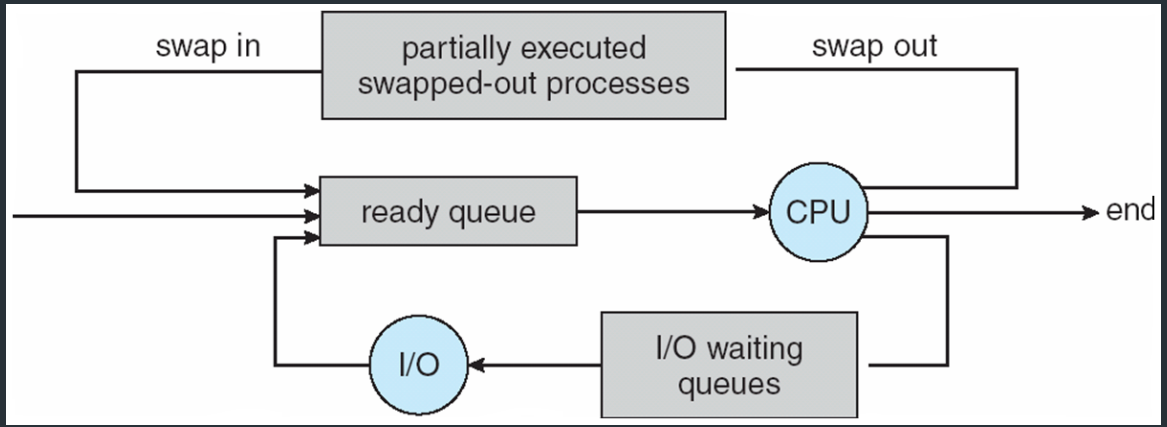


Schedulers

- **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds) ⇒ (must be fast)
- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes) ⇒ (may be slow)
  - The long-term scheduler controls the **degree of multiprogramming**
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts
- Long-term scheduler strives for good ***process mix***

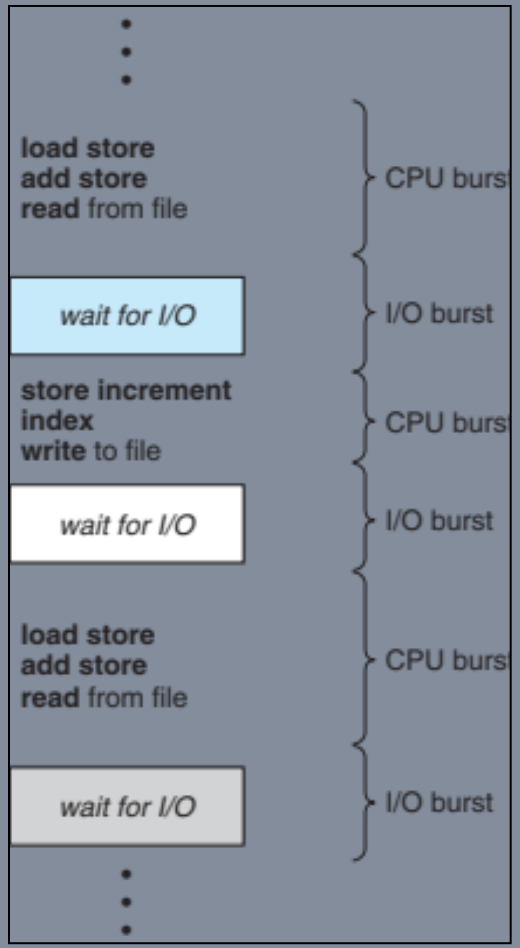
Medium Term Scheduler

- **Medium-term scheduler** can be added if degree of multiple programming needs to decrease
  - Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**



Scheduling Algorithms: 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

- **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 **non-preemptive**
- 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 it takes for the dispatcher to stop one process and start another running

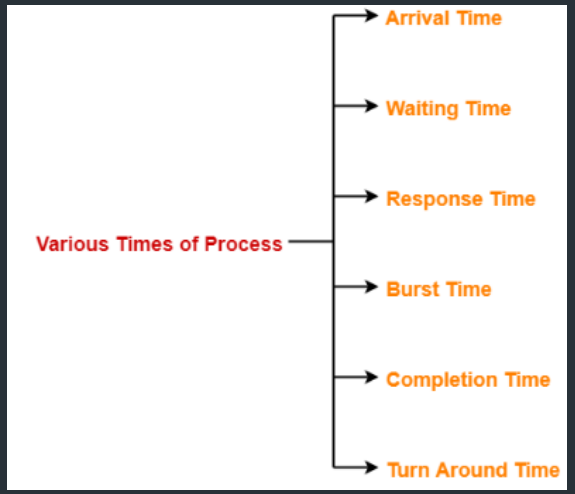
Scheduling Criteria

- **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
- **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

Various Times Related to Process



Burst Time- Burst time is the amount of time required by a process for executing on CPU. It is also called as **execution time** or **running time**.

Arrival Time- Arrival time is the point of time at which a process enters the ready queue.

Completion Time- Completion time is the point of time at which a process completes its execution on the CPU and takes exit from the system.

Waiting Time- Waiting time is the amount of time spent by a process waiting in the ready queue for getting the CPU.

Waiting time = Turn Around time – Burst time

Turn Around Time- Turn Around time is the total amount of time spent by a process in the system. When present in the system, a process is either waiting in the ready queue for getting the CPU or it is executing on the CPU.

Turn Around time = Burst time + Waiting time

OR

Turn Around time = Completion time – Arrival time

Response Time- Response time is the amount of time after which a process gets the CPU for the first time after entering the ready queue.

Response Time = Time at which process first gets the CPU – Arrival 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 , 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

Shortest-Job-First (SJF) Scheduling

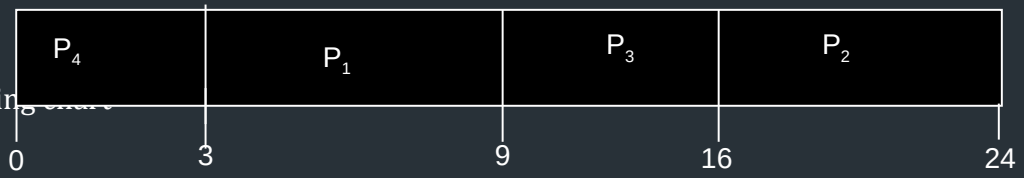
- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time

- 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

Example of SJF

Process	Arrival Time	Burst Time
P_1	0.0	6
P_2	2.0	8
P_3	4.0	7
P_4	5.0	3

- SJF scheduling chart



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

Problem with Non Pre-emptive SJF

Problem with Non Pre-emptive SJF

If the **arrival time** for processes are different, which means all the processes are not available in the ready queue at time `0`, and some jobs arrive after some time, in such situation, sometimes process with short burst time have to wait for the current process's execution to finish, because in Non Pre-emptive SJF, on arrival of a process with short duration, the existing job/process's execution is not halted/stopped to execute the short job first.

This leads to the problem of **Starvation**, where a shorter process has to wait for a long time until the current longer process gets executed. This happens if shorter jobs keep coming, but this can be solved using the concept of **aging**.

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

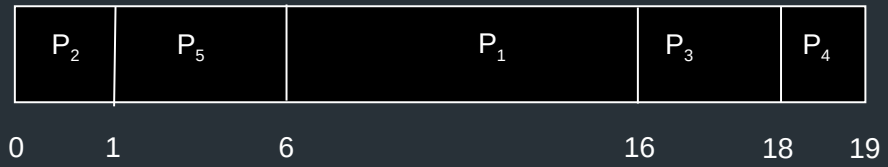
- Average waiting time = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

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
  - Nonpreemptive
- 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 msec

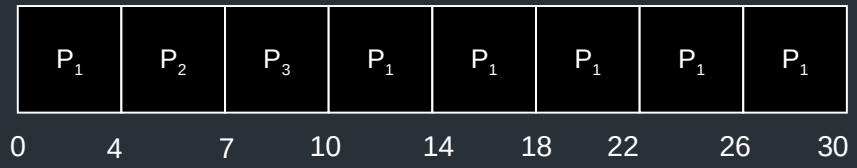
Round Robin (RR)

- 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

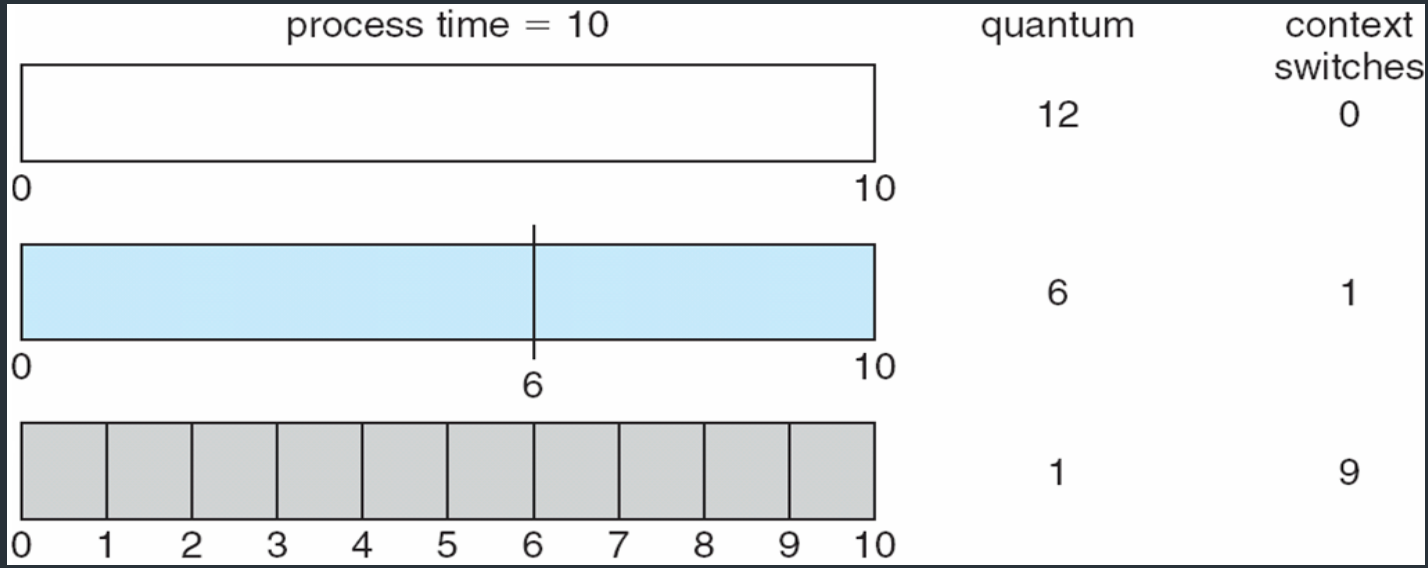
Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



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

Time Quantum and Context Switch Time



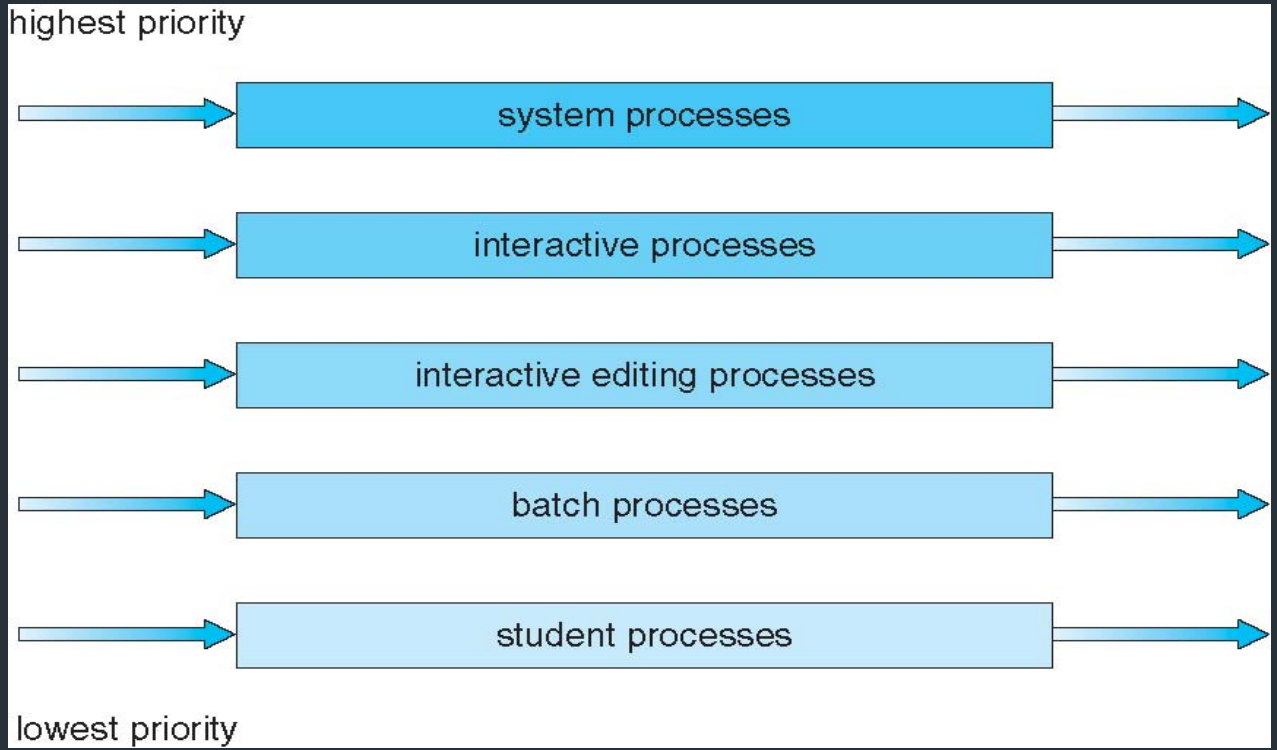
Multilevel Queue

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

Multilevel Queue Scheduling



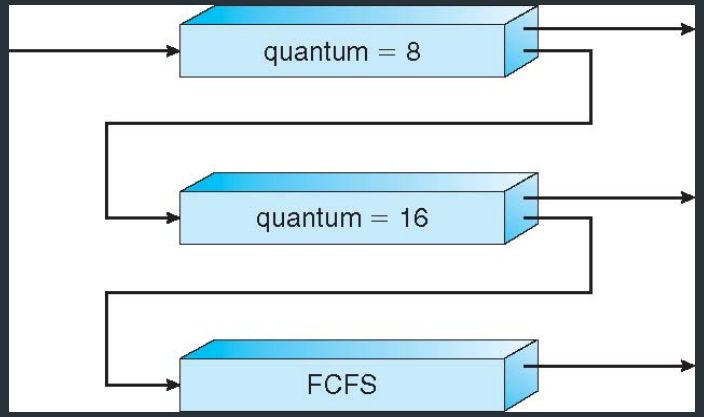
Multilevel Feedback Queue

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

- 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:
  - 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 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 FCFS and receives 16 additional milliseconds
    - If it still does not complete, it is preempted and moved to queue Q_2

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

Problem

- Calculate the predicted burst time using exponential averaging for the fifth process if the predicted burst time for the first process is 10 units and actual burst time of the first four processes is 4, 8, 6 and 7 units respectively. Given $\alpha = 0.5$.

- **Solution-**

  Given- Predicted burst time for 1^{st} process = 10 units

  Actual burst time of the first four processes = 4, 8, 6, 7

  $\alpha = 0.5$

- **Predicted Burst Time for 2nd Process-**

  Predicted burst time for 2^{nd} process

  = α x Actual burst time of 1^{st} process + (1-α) x Predicted burst time for 1^{st} process

  = 0.5 x 4 + 0.5 x 10

  = 2 + 5

  = 7 units

Predicted Burst Time for 3rd Process-

Predicted burst time for 3rd process
= α x Actual burst time of 2nd process + (1-α) x Predicted burst time for 2nd process
= 0.5 x 8 + 0.5 x 7
= 4 + 3.5
= 7.5 units

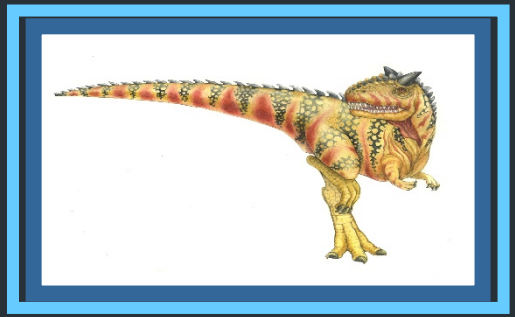
Predicted Burst Time for 4th Process-

Predicted burst time for 4th process
= α x Actual burst time of 3rd process + (1-α) x Predicted burst time for 3rd process
= 0.5 x 6 + 0.5 x 7.5
= 3 + 3.75
= 6.75 units

Predicted Burst Time for 5th Process-

Predicted burst time for 5th process
= α x Actual burst time of 4th process + (1-α) x Predicted burst time for 4th process
= 0.5 x 7 + 0.5 x 6.75
= 3.5 + 3.375
= 6.875 units

Additional Problems



Example problems

Job	Arrival Time	Burst Time
A	0	6
B	1	2
C	2	5
D	3	7
E	7	1

The **average wait time** would be using a FCFS algorithm:

(6-6)+(7-2)+(11-5)+(17-5)+(14-1) -> 0+5+6+10+13 -> 34/5 = 7 (6.8)

What would the **average turnaround time** be?

the turnaround time is TAT= Completion time-arrival time for A=6-0 B=8-1 C=13-2 D=20-3 E=21-7 that gives 6+7+11+17+14/5=11

so average TAT=11

Exercises

Consider the below processes available in the ready queue for execution, with **arrival time** as `0` for all and given **burst times**.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

In Shortest Job First Scheduling, the shortest Process is executed first.

Now, the average waiting time will be = (0 + 2 + 5 + 11)/4 = 4.5 ms

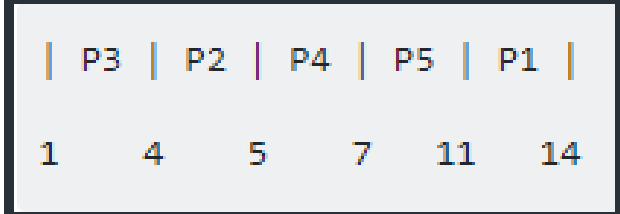
As you can see in the **GANTT chart** above, the process **P4** will be picked up first as it has the shortest burst time, then **P2**, followed by **P3** and at last **P1**.

We scheduled the same set of processes using the First come first serve algorithm in the previous tutorial, and got average waiting time to be `18.75 ms`, whereas with SJF, the average waiting time comes out `4.5 ms`.

Calculate Average Waiting Time and
average Turn-around time in SJF Scheduling

In SJF (Shortest Job First) Scheduling method.

Process	Arrival Time (ms)	Processing Time (ms)
P1	8	3
P2	2	1
P3	1	3
P4	3	2
P5	4	4



Average waiting time=(0+2+2+3+3)/5=2

Average Turnaround time=(3+3+4+7+6)/5=4.6

Consider the following set of processes, with the arrival times and the CPU-burst times given in milliseconds (GATE-CS-2004)

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

What is the average turnaround time for these processes with the preemptive shortest remaining processing time first (SRPT) algorithm ?

(A) 5.50

(B) 5.75

(C) 6.00

(D) 6.25

Answer (A)

Solution:

The following is Gantt Chart of execution

P1	P2	P4	P3	P1
1	4	5	8	12

Turn Around Time = Completion Time – Arrival Time

Avg Turn Around Time = (12 + 3 + 6+ 1)/4 = 5.50

Thank you