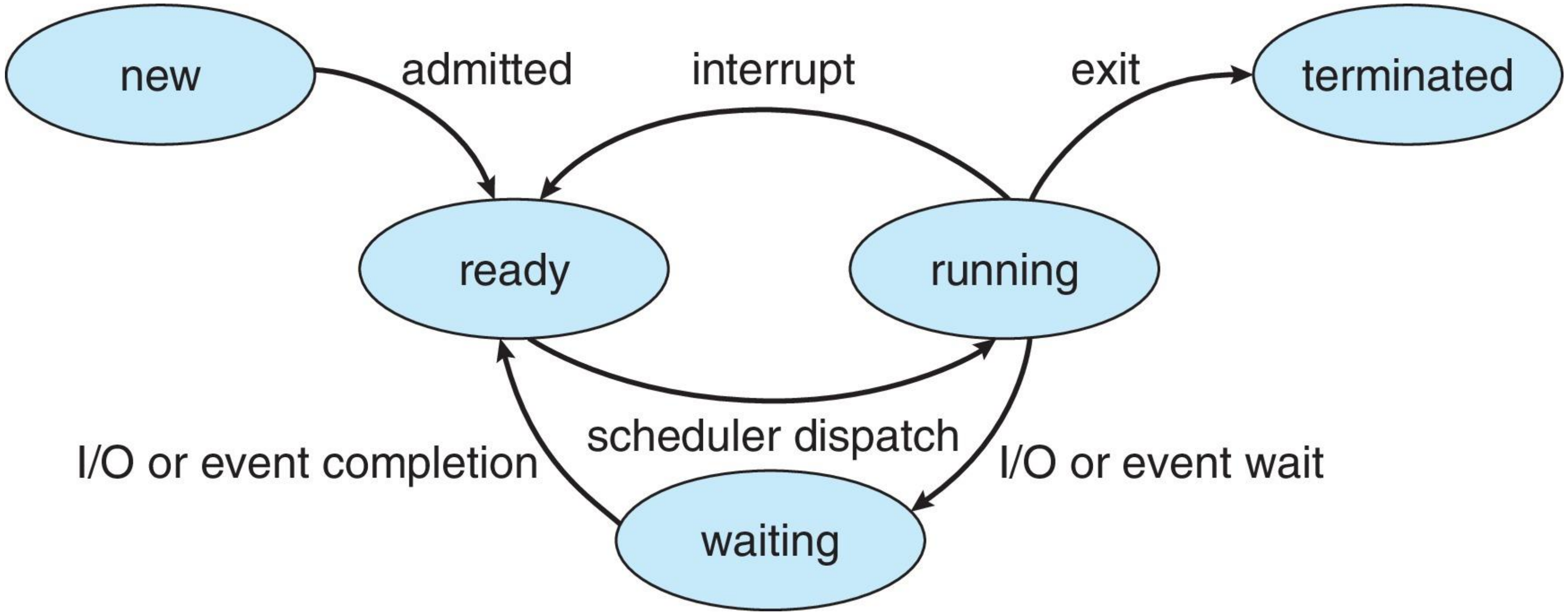
CPU Scheduling

19CSE213 OPERATING SYSTEM

Basics

- An operating system executes a variety of programs that run as a process.
- **Process** – a program in execution;
- Program is **passive** entity stored on disk (**executable file**); process is **active**
- Program becomes process when an executable file is loaded into memory
- As a process executes, it changes **state**
  - **New**: The process is being created
  - **Running**: Instructions are being executed
  - **Waiting**: The process is waiting for some event to occur
  - **Ready**: The process is waiting to be assigned to a processor
  - **Terminated**: The process has finished execution

Diagram of Process State



CPU Scheduler

The **CPU scheduler** selects from among the processes in ready queue, and allocates a CPU core to one of them

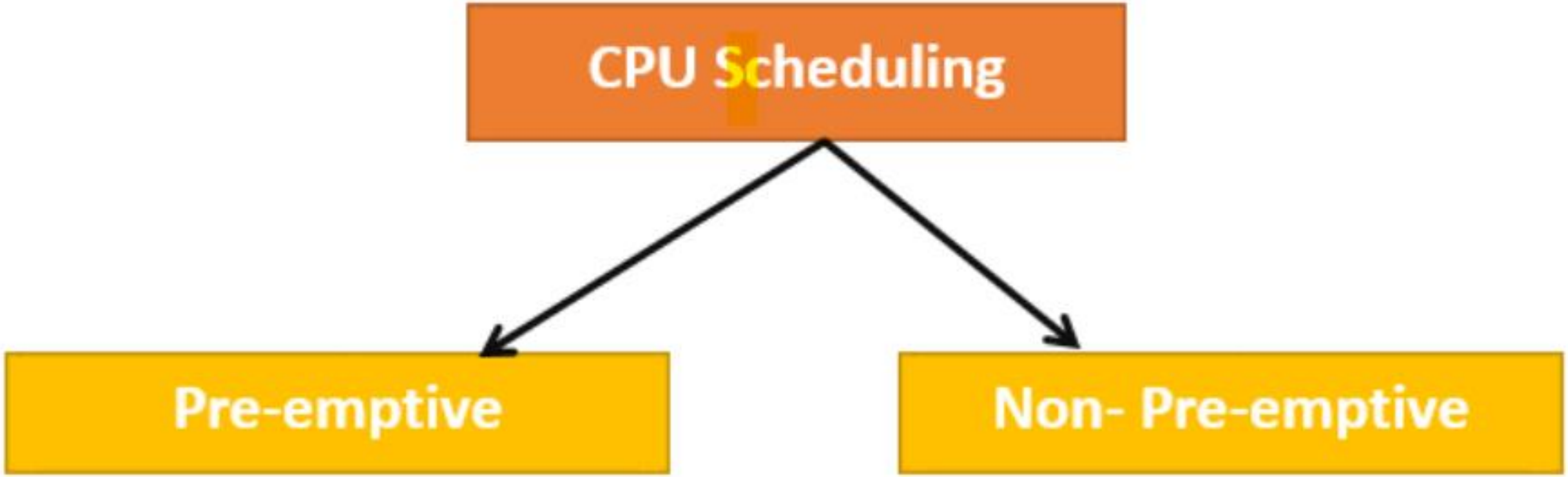
CPU scheduling decisions may take place when a process:

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.

Types of CPU Scheduling



Preemptive and Non-preemptive Scheduling

- When scheduling takes place only under circumstances 1 and 4, the scheduling scheme is **non-preemptive**.

- Otherwise, it is **preemptive**.

- Under Non-preemptive 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, Mac OS, 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 the first 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

- A dispatcher is a special program that comes into play after the scheduler
- Scheduler selects a process out of several processes to be executed
- While the dispatcher allocates CPU for the process selected by the scheduler
- **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** – No: 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.

Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Types of CPU Scheduling Algorithms

- First Come First Serve (FCFS)
- Shortest-Job-First (SJF) Scheduling.
- Shortest Remaining Time First.
- Priority Scheduling.
- Round Robin Scheduling

CPU Scheduling

- **The various types of times in CPU scheduling algorithms are as follows:**
  - **Arrival Time:** Time at which the process arrives in the ready queue
  - **Completion Time:** Time at which process completes its execution
  - **Burst Time:** Time required by a process for CPU execution
  - **Turn Around Time:** Time Difference between completion time and arrival time

    Turn Around Time = Completion Time – Arrival Time
  - **Waiting Time(W.T):** Time Difference between turn around time and burst time

    Waiting Time = Turn Around Time – Burst Time

First- Come, First-Served (FCFS) Scheduling

- First Come First Serve is the full form of FCFS.
- It is the easiest and most simple CPU scheduling algorithm.
- In this type of algorithm, the process which requests the CPU gets the CPU allocation first.
- This scheduling method can be managed with a FIFO queue

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

Example - FCFS

Process	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

Example - FCFS

Using the FCFS scheduling algorithm, these processes are handled as follows:

Step 0) The process begins with P4 which has arrival time 0

Step 1) At time=1, P3 arrives. P4 is still executing. Hence, P3 is kept in a queue

Step 2) At time= 2, P1 arrives which is kept in the queue

Step 3) At time=3, P4 process completes its execution

Step 4) At time=4, P3, which is first in the queue, starts execution

Step 5) At time =5, P2 arrives, and it is kept in a queue

Step 6) At time 11, P3 completes its execution

Step 7) At time=11, P1 starts execution. It has a burst time of 6. It completes execution at time interval 17

Step 8) At time=17, P5 starts execution. It has a burst time of 4. It completes execution at time=21

Step 9) At time=21, P2 starts execution. It has a burst time of 2. It completes execution at time interval 23

Step 10) Let’s calculate the average waiting time for above example.

FCFS example – Waiting time

- Waiting time = Start time – Arrival time

P4=0-0=0

P3=3-1=2

P1=11-2=9

P5=17-4=13

P2=21-5=16

- Average waiting time = (0+2+9+13+16)/5 = 40/5 =8

Pros and Cons of FCFS method

- It offers non-preemptive scheduling algorithm
- Jobs are always executed on a first-come, first-serve basis
- It is easy to implement and use
- The Average Waiting Time is high.
- Short processes that are at the back of the queue have to wait for the long process at the front to finish
- Not an ideal technique for time-sharing systems

Shortest-Job-First (SJF) Scheduling

- Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next
- It may cause starvation if shorter processes keep coming.
- This problem can be solved using the concept of ageing.
- Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time
- SJF is optimal – gives minimum average waiting time for a given set of processes
- Preemptive version called shortest-remaining-time-first

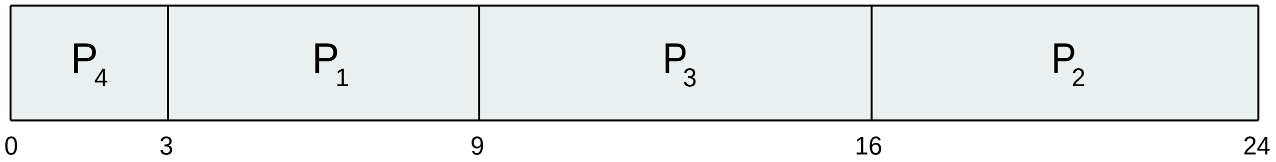
SJF Algorithm

- Sort all the process according to the arrival time
- Then select that process which has minimum arrival time and minimum Burst time
- After completion of process make a pool of process which after till the completion of previous process and select that process among the pool which is having minimum Burst time.

Example of SJF Scheduling

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

SJF scheduling chart



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

Non-Preemptive SJF - Example

Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

Non-preemptive SJF Example – Steps

Step 0) At time=0, P4 arrives and starts execution

Step 1) At time= 1, Process P3 arrives. But, P4 still needs 2 execution units to complete. It will continue execution

Step 2) At time =2, process P1 arrives and is added to the waiting queue. P4 will continue execution

Step 3) At time = 3, process P4 will finish its execution. The burst time of P3 and P1 is compared. Process P1 is executed because its burst time is less compared to P3

Step 4) At time = 4, process P5 arrives and is added to the waiting queue. P1 will continue execution

Step 5) At time = 5, process P2 arrives and is added to the waiting queue. P1 will continue execution.

Non-preemptive SJF Example – Steps

Step 6) At time = 9, process P1 will finish its execution. The burst time of P3, P5, and P2 is compared. Process P2 is executed because its burst time is the lowest.

Step 7) At time=10, P2 is executing and P3 and P5 are in the waiting queue.

Step 8) At time = 11, process P2 will finish its execution. The burst time of P3 and P5 is compared. Process P5 is executed because its burst time is lower.

Step 9) At time = 15, process P5 will finish its execution.

Step 10) At time = 23, process P3 will finish its execution.

Non-preemptive SJF example – Waiting time

```
Wait time
P4= 0-0=0
P1=  3-2=1
P2= 9-5=4
P5= 11-4=7
P3= 15-1=14
```

```
Average Waiting Time= 0+1+4+7+14/5 = 26/5 = 5.2
```

Shortest Remaining Time First Scheduling

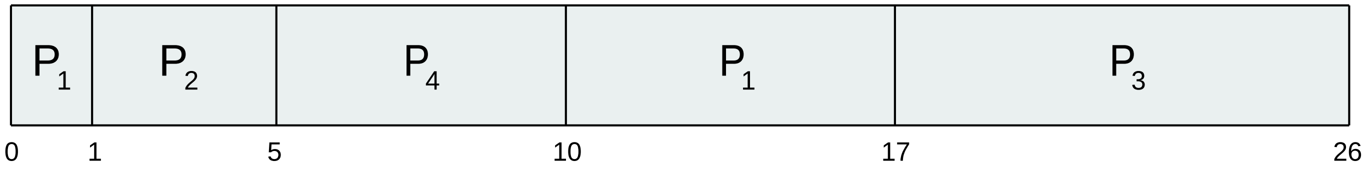
- The Preemptive version of Shortest Job First (SJF) scheduling is known as Shortest Remaining Time First (SRTF)
- In this scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute
- In SRTF, the execution of the process can be stopped after certain amount of time.
- Once all the processes are available in the ready queue, No preemption will be done and the algorithm will work as SJF scheduling.

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



Example - SRTF

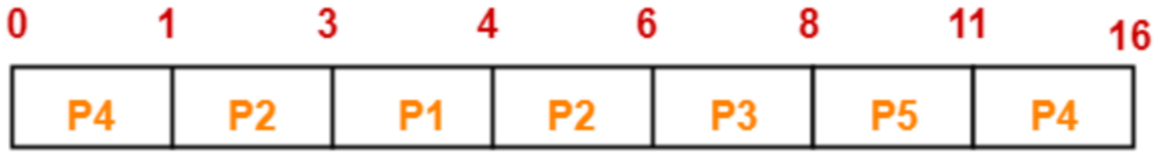
- Turn Around time = Completion time – Arrival time
- Waiting time = Turn Around time – Burst time

Process	Exit Time	Turn Around Time	Waiting time
P1	17	17-0 =17	17 – 8 = 9
P2	5	5-1 = 4	4 – 4 = 0
P3	26	26 –2 = 24	24 – 9 = 15
P4	10	10-3 = 7	7 - 5 = 2

- Average waiting time = [9+0+15+2]/4 = 26/4 = 6.5

Exercise 1 -SRTF

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3



SRTF : Exercise1 –Answer

Process Id	Exit time	Turn Around time	Waiting time
P1	4	4 – 3 = 1	1 – 1 = 0
P2	6	6 – 1 = 5	5 – 4 = 1
P3	8	8 – 4 = 4	4 – 2 = 2
P4	16	16 – 0 = 16	16 – 6 = 10
P5	11	11 – 2 = 9	9 – 3 = 6

- Average Turn Around time = (1 + 5 + 4 + 16 + 9) / 5 = 35 / 5 = 7
- Average waiting time = (0 + 1 + 2 + 10 + 6) / 5 = 19 / 5 = 3.8

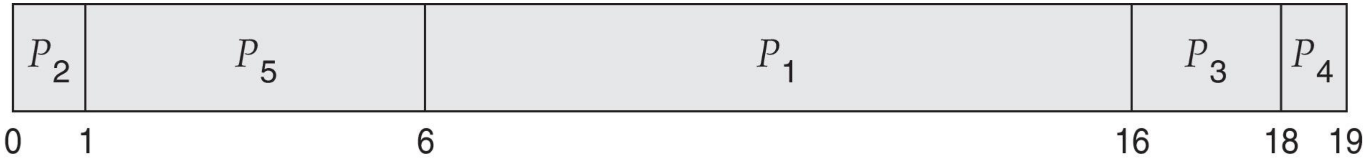
Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer $\equiv$ highest priority)
  - Preemptive
  - Non-preemptive
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Problem $\equiv$ **Starvation** – low priority processes may never execute
- Solution $\equiv$ **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

Exercise – Theory and Practice Lab 2

- Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

Process	Burst Time	Priority
*P*1	2	2
*P*2	1	1
*P*3	8	4
*P*4	4	2
*P*5	5	3

- The processes are assumed to have arrived in the order *P*1, *P*2, *P*3, *P*4, *P*5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms:
   - FCFS, SJF, non-preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2)

b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Multilevel Queue Scheduling

- It may happen that processes in the ready queue can be divided into different classes where each class has its own scheduling needs.
- A common division is a **foreground (interactive)** process and a **background (batch)** process.
- These two classes have different scheduling needs.
- For this kind of situation Multilevel Queue Scheduling is used.

Multilevel Queue

- The ready queue consists of multiple queues
- Multilevel queue scheduler defined by the following parameters:
    - Number of queues
    - Scheduling algorithms for each queue
    - Method used to determine which queue a process will enter when that process needs service
    - Scheduling among the queues

Multilevel Queue

- **Ready Queue** is divided into separate queues for each class of processes

- For example, let us take three different types of processes
  - Real time processes
  - System processes
  - Interactive processes
  - Batch Processes

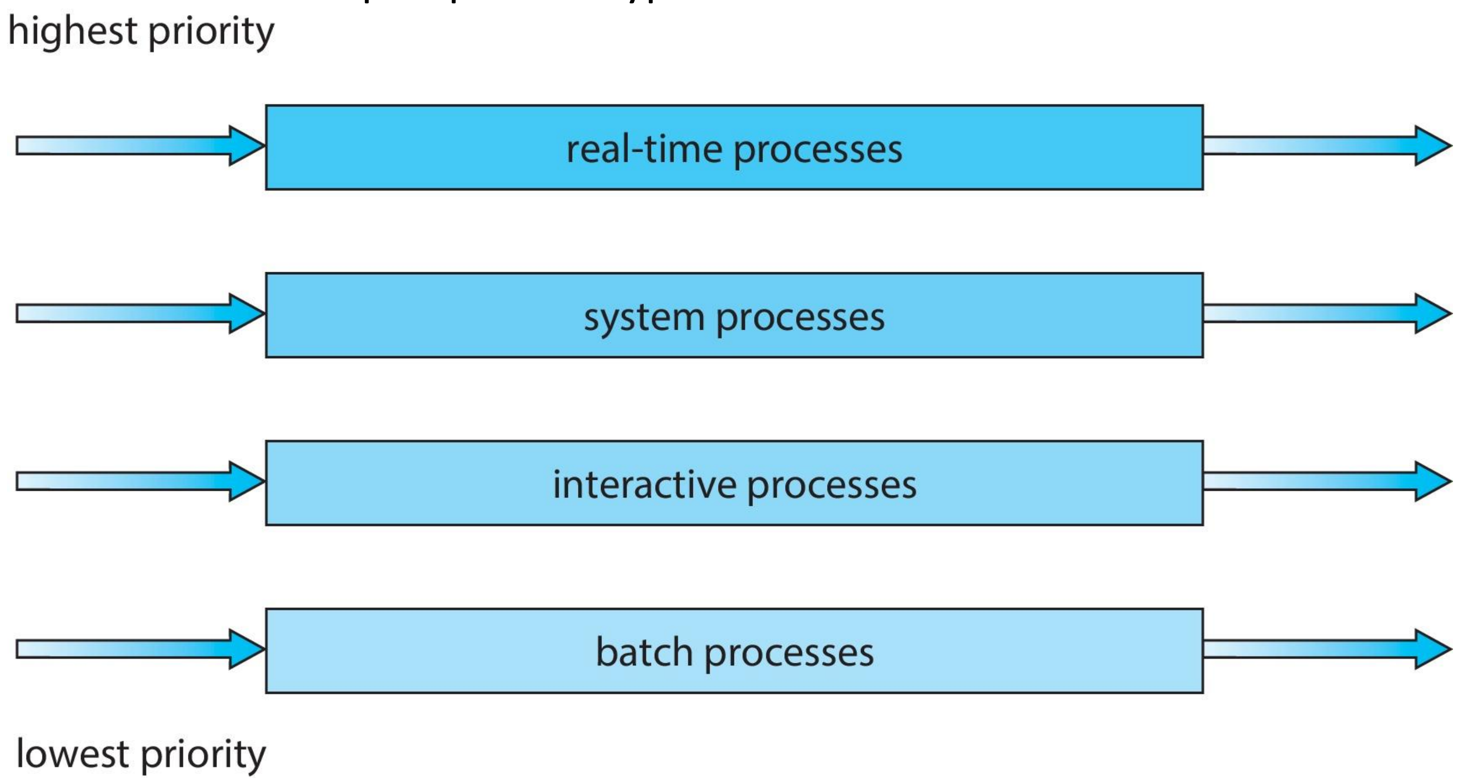
- All processes have their own queue

- Each queue has its own Scheduling algorithm

- For example, queue 1 and queue 2 uses **Round Robin** while queue 3 can use **FCFS** to schedule their processes.

Multilevel Queue

- Prioritization based upon process type



Scheduling among the queues

There are two ways to do so –

1. Fixed priority preemptive scheduling method: –

- Each queue has absolute priority over the lower priority queue.
- Let us consider following priority order **queue 1 > queue 2 > queue 3**.
- According to this algorithm, no process in the batch queue(queue 3) can run unless queues 1 and 2 are empty.
- If any batch process (queue 3) is running and any system (queue 1) or Interactive process(queue 2) entered the ready queue the batch process is preempted.

2. Time slicing :–

- In this method, each queue gets a certain portion of CPU time and can use it to schedule its own processes.
- For instance, queue 1 takes 50 percent of CPU time queue 2 takes 30 percent and queue 3 gets 20 percent of CPU time.

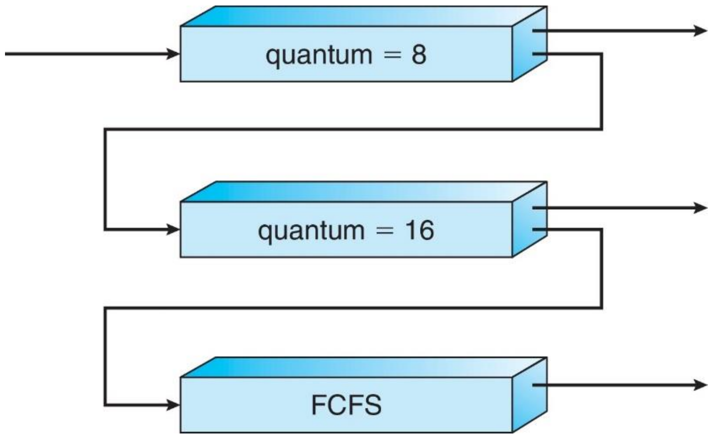
Multilevel Feedback Queue

- Some processes may starve for CPU if some higher priority queues are never becoming empty
- A process can move between the various queues
- Aging can be implemented using multilevel feedback queue
- 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

Multilevel Feedback Queue Implementation

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

Multilevel Feedback Queue Implementation

- In a general case if a process does not complete in a time quantum than it is shifted to the lower priority queue

- In the last queue, processes are scheduled in FCFS manner

- A process in lower priority queue can only execute only when higher priority queues are empty

- A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue

Summary

- Scheduling is important for improving the system performance
- Methods of prediction play an important role in Operating system and network functions
- Simulation is a way of experimentally evaluating the performance of a technique