Topics to be covered

- What is scheduling
- Objectives of scheduling
- Types of scheduler
- Scheduling criteria
- Scheduling algorithms
  - First Come First Served (FCFS)
  - Shortest Job First (SJF)
  - Shortest Remaining Time Next (SRTN)
  - Round Robin (RR)
  - Priority
    - ✓ Preemptive Priority
    - ✓ Non-Preemptive Priority
  - Other Scheduling algorithms

- **Source:** Internet, Digital Libraries, Reference Books, etc.
- **Disclaimer:** The presentation is prepared from various sources and intended for educational purposes only

What is Process scheduling?

- Process scheduling is the activity of the process manager that **handles suspension of running process** from CPU and **selection of another process** on the basis of a particular strategy.
- The **part of operating system** that **makes the choice** is called scheduler.
- The **algorithm used** by this scheduler is called scheduling algorithm.
- Process scheduling is an essential part of a multiprogramming operating systems.

Objectives (goals) of scheduling

- **Fairness**: **giving each process a fair share of the CPU**.
- **Balance**: **keeping all the parts** of the system **busy** (Maximize).
- **Throughput**: **no of processes** that are **completed per time** unit (Maximize).

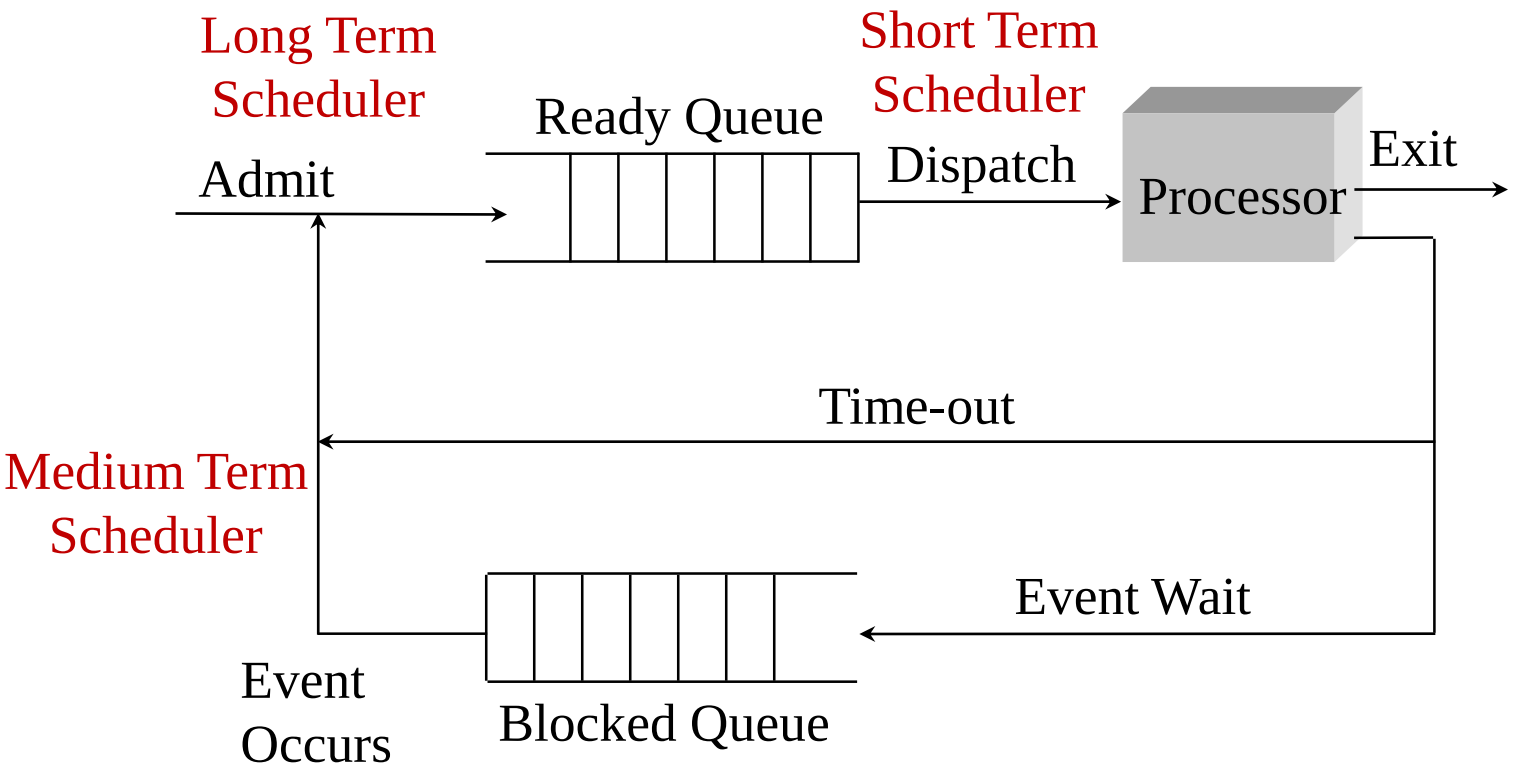
  Rate of production = number of processes / Time
- **Turnaround time**: **time to execute a process** from submission to completion (Minimize).
  - **Turnaround time** = Process finish time – Process arrival time

Objectives (goals) of scheduling

- **CPU utilization**: It is **percent of time** that the **CPU is busy** in executing a process.
  - keep CPU as busy as possible (Maximized).
- **Response time**: **time between issuing a command** and **getting** the **result** (Minimized).
- **Waiting time**: **amount of time a process** has been **waiting** in the ready queue (Minimize).
  - **Waiting time** = Turnaround time – Actual execution time

Process finish time - Process arrival time

Types of schedulers

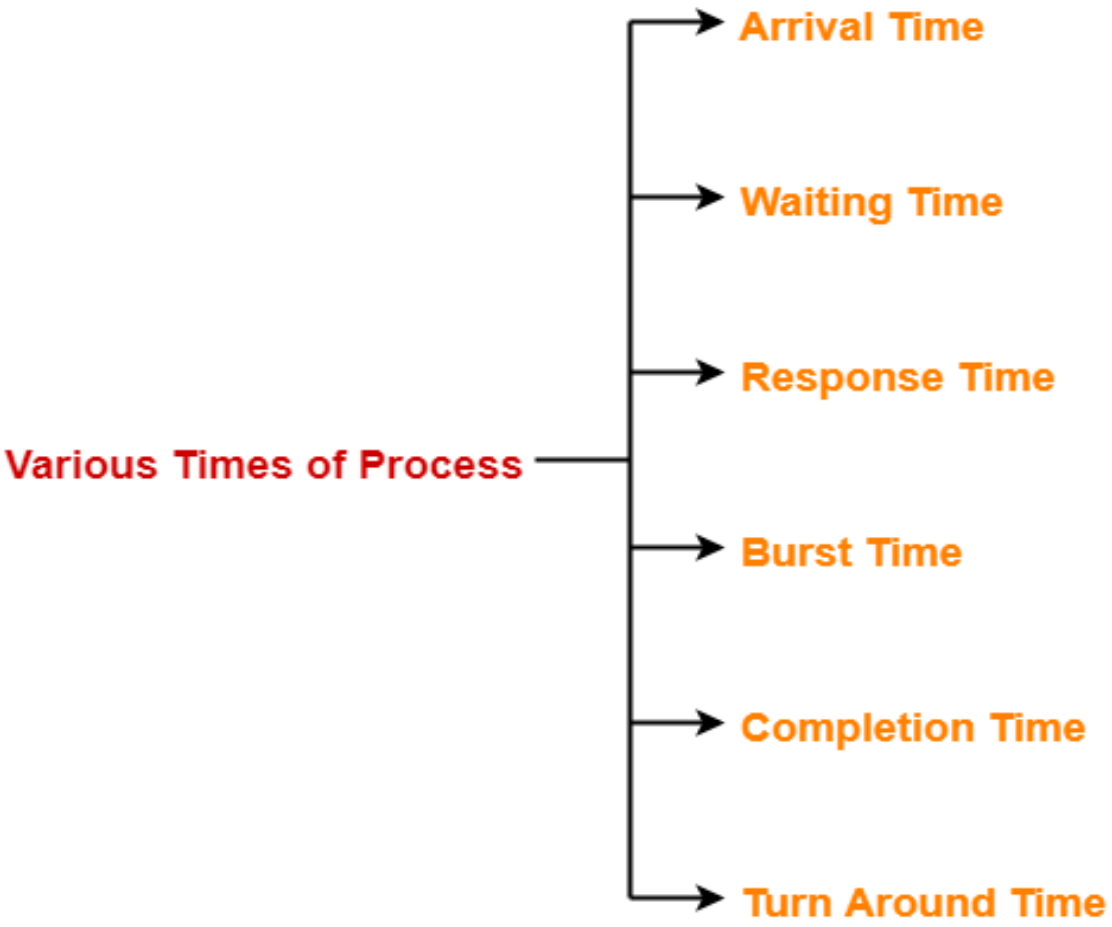


Types of schedulers

Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
It is a job scheduler.	It is a CPU scheduler.	It is a process swapping scheduler.
It selects processes from pool and loads them into memory for execution.	It selects those processes which are ready to execute.	It can re-introduce the process into memory and execution can be continued.
Speed is lesser than short term scheduler. Lowest speed	Speed is fastest among other two schedulers.	Speed is in between both short and long term scheduler.
It is almost absent or minimal in time sharing system.	It is also minimal in time sharing system.	It is a part of time sharing systems.

Scheduling algorithms

1. First Come First Served (FCFS)
2. Shortest Job First (SJF)
3. Shortest Remaining Time Next (SRTN)
4. Round Robin (RR)
5. Priority
    1. Preemptive
    2. Non-Preemptive


Various Times of Process
Arrival Time
Waiting Time
Response Time
Burst Time
Completion Time
Turn Around Time

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

Waiting time : Turn Around time – Burst time

Process finish time - Process arrival time

Response Time : Time at which process first gets the CPU – Arrival time

BT: Amount of time required by a process for executing on CPU

Burst time

CT: Process completes its execution on the CPU and takes exit from the system.

Completion time

Turn Around time: Burst time + Waiting time

First Come First Served (FCFS)

- Selection criteria
  - The **process** that **request first is served first**.
  - It means that **processes are served in the exact order of their arrival**.
- Decision Mode

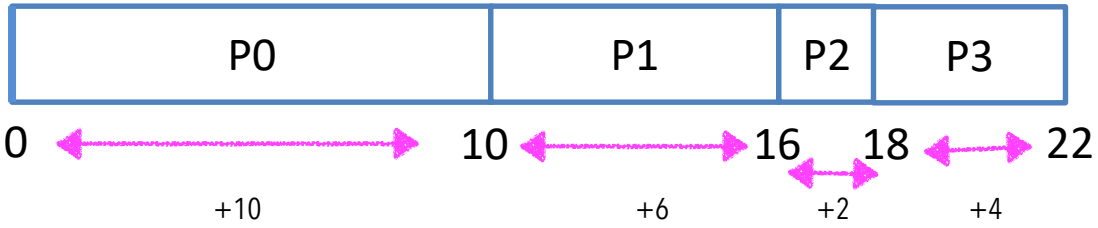
  - **Non preemptive**: Once a process is selected, it runs until it is blocked for an I/O or some other event or it is terminated.
- Implementation:
  - This strategy can be **easily implemented** by using FIFO (First In First Out) queue.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

First Come First Served (FCFS)

- Example

Process	Arrival Time (T0)	Time required for completion (ΔT) (CPU Burst Time)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

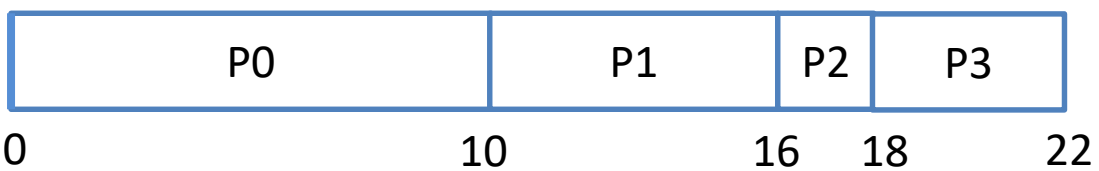
- Gantt Chart



First Come First Served (FCFS)

Process	Arrival Time (T0)	Burst Time (ΔT)	Process finish time
P0	0	10	10
P1	1	6	16
P2	3	2	18
P3	5	4	22

- Gantt Chart



Process finish time - Process arrival time || Bust time + waiting time

- Average Turnaround Time: (10+15+15+17)/4 = 14.25 ms.
- Average Waiting Time: (0+9+13+13)/4 = 8.75 ms.

Turnaround time - burst time

First Come First Served (FCFS)

- Advantages
  - **Simple** and **fair**.
  - **Easy to understand** and **implement**.
  - Every process will get a chance to run, so **starvation doesn't occur**.

Extremem hunger

- Disadvantages
  - **Not efficient** because average waiting time is too high.
  - **Convoy effect is possible**. All small I/O bound processes wait for one big CPU bound process to acquire CPU.
  - **CPU utilization may be less efficient** especially when a CPU bound process is running with many I/O bound processes.

Shortest Job First (SJF)

- Selection criteria
  - The process, that **requires shortest time to complete execution**, is **served first**.
- Decision Mode

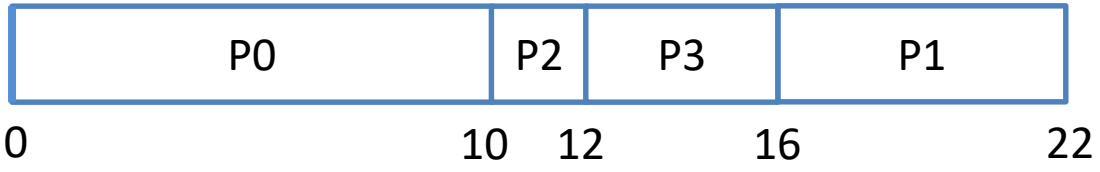
  Preemptive : Prevent attack by disabling the enemy
  - **Non preemptive**: Once a process is selected, it runs until either it is blocked for an I/O or some other event or it is terminated.
- Implementation:
  - This strategy can be **easily implemented** by using FIFO (First In First Out) queue.
  - All processes in a queue are **sorted in ascending order** based on their required CPU bursts.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

Shortest Job First (SJF)

- Example

Process	Arrival Time (T0)	Time required for completion (ΔT) (CPU Burst Time)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

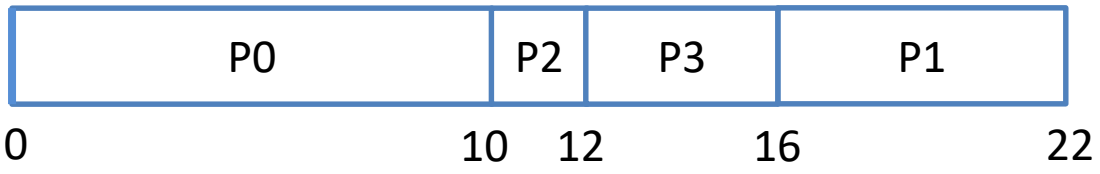
- Gantt Chart



Shortest Job First (SJF)

Process	Arrival Time (T0)	Burst Time (ΔT)	Process finish time
P0	0	10	22
P1	1	6	12
P2	3	2	2
P3	5	4	6

- Gantt Chart



- Average Turnaround Time: (10+21+9+11)/4 = 12.75 ms.
- Average Waiting Time: (0+15+7+7)/4 = 7.25 ms.

Shortest Job First (SJF)

- Advantages:
  - **Less waiting time**.
  - **Good response** for **short processes**.
- Disadvantages :
  - It is **difficult to estimate time required** to complete execution.
  - **Starvation is possible for long process**. Long process may wait forever.

Shortest Remaining Time Next (SRTN)

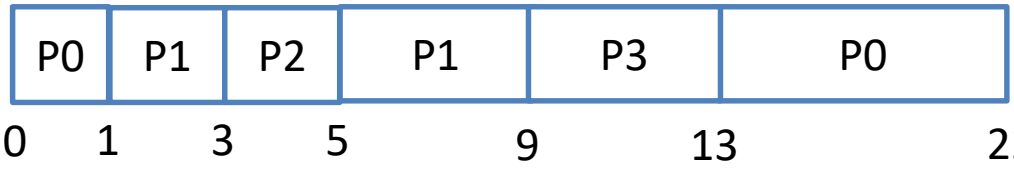
- Selection criteria :
  - The process, **whose remaining run time is shortest**, is **served first**. This is a **preemptive version of SJF scheduling**.
- Decision Mode:
  - **Preemptive**: When a new process arrives, its total time is compared to the current process remaining run time.
  - If the new process needs less time to finish than the current process, the current process is suspended and the new job is started.
- Implementation :
  - This strategy can also be implemented by using sorted FIFO queue.
  - All processes in a queue are **sorted in ascending order on their remaining run time**.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

Shortest Remaining Time Next (SRTN)

- Example

Process	Arrival Time (T0)	Time required for completion (ΔT) (CPU Burst Time)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart

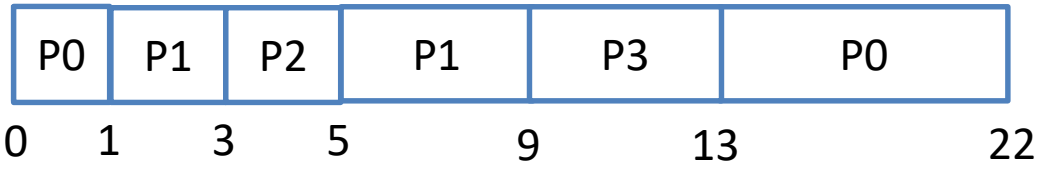


Proce	Proce	Process	Process	Process	Remaining Time
P1	P0	P0	P0	P0	9
P0	P2	P1	P3	4	
	P1	P3	4		

Shortest Remaining Time Next (SRTN)

Process	Arrival Time (T0)	Burst Time (ΔT)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart



- Average Turnaround Time: (22+8+2+8) / 4 = 10 ms.
- Average Waiting Time: (12+2+0+4)/4 = 4.5 ms.

Shortest Remaining Time Next (SRTN)

- Advantages :
  - **Less waiting time**.
  - **Quite good response** for **short processes**.
- Disadvantages :
  - Again it is **difficult to estimate remaining time** necessary to complete execution.
  - **Starvation is possible** for **long process**. Long process may wait forever.
  - **Context switch overhead is there**.

Round Robin (RR)

- Selection Criteria
  - Each selected process is assigned a time interval, called **time quantum or time slice**.
  - Process is **allowed to run only for this time interval**.
  - Here, two things are possible:
    - First, process is either blocked or terminated before the quantum has elapsed. In this case the CPU switching is done and another process is scheduled to run.
    - Second, process needs CPU burst longer than time quantum. In this case, process is running at the end of the time quantum.
    - Now, it will be preempted and moved to the end of the queue.
    - CPU will be allocated to another process.
    - Here, length of time quantum is critical to determine.

Round Robin (RR)

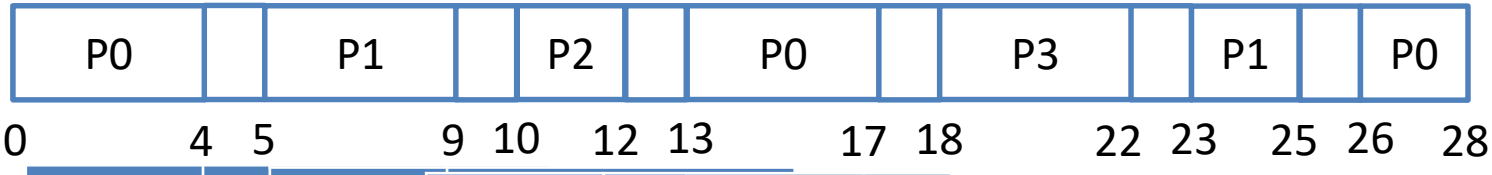
- Decision Mode:
  - **Preemptive**: When quantum time is over or process completes its execution (which ever is earlier), it starts new job.
  - **Selection** of new job is **as per FCFS** scheduling algorithm
- Implementation :
  - This strategy can be implemented by using circular FIFO queue.
  - If any process comes, or process releases CPU, or process is preempted. It is moved to the end of the queue.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

Round Robin (RR)

- Example

Process	Arrival Time (T0)	Time required for completion (ΔT) (CPU Burst Time)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart (Quantum time is 4 ms & context switch overhead is 1 ms)

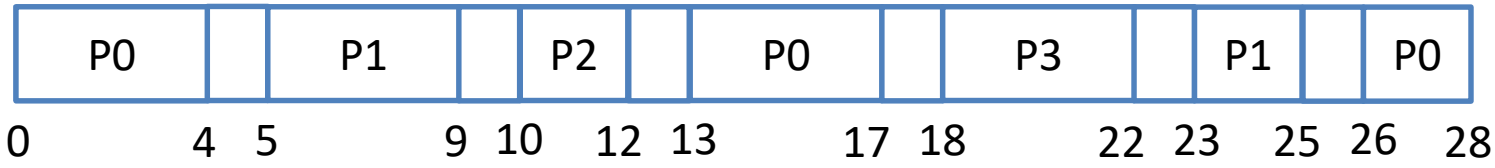


Process	Re	Process	Process	Rem	Process	Rem	Process	Process	Remaining Time
P1	6	P2	P0	6	P3	4	P1	P0	2
P2	2	P0	P3	4	P1	2	P0	2	
P0	6	P3	P1	2	P0	2			
		P1	2						

Round Robin (RR)

Process	Arrival Time (T0)	Burst Time (ΔT)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart (Quantum time is 4 ms & context switch overhead is 1 ms)



- Average Turnaround Time: (28+24+9+17)/4 = 19.5 ms.
- Average Waiting Time: (18+18+7+13)/4 = 14 ms.

Round Robin (RR)

- Advantages:
  - **Simplest, fairest** and **most widely used algorithms**.
- Disadvantages:
  - **Context switch overhead** is there.
  - **Determination of time quantum is too critical**.
    - ✓ If it is too **short**, it **causes frequent context switches** and **lowers CPU efficiency**.
    - ✓ If it is too **long**, it **causes poor response** for **short interactive process**.

Non Preemptive Priority Scheduling

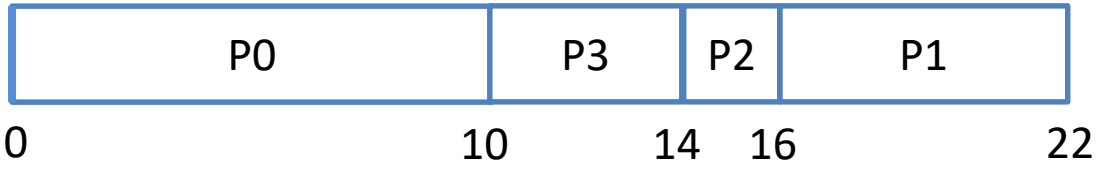
- Selection criteria :
  - The process, that **has highest priority, is served first**.
- Decision Mode:
  - **Non Preemptive**: Once a process is selected, it runs until it blocks for an I/O or some event or it terminates.
- Implementation :
  - This strategy can be implemented by using sorted FIFO queue.
  - All processes in a queue are **sorted based on their priority with highest priority process at front end**.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

Non Preemptive Priority Scheduling

- Example

Process	Arrival Time (T0)	Time required for completion (ΔT) (CPU Burst Time)	Priority
P0	0	10	5
P1	1	6	4
P2	3	2	2
P3	5	4	0

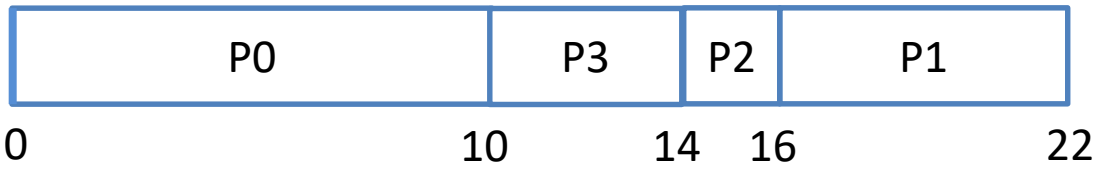
- Gantt Chart (small values for priority means higher priority of a process)



Non Preemptive Priority Scheduling

Process	Arrival Time (T0)	Burst Time (ΔT)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart (small values for priority means higher priority of a process)



- Average Turnaround Time: (10+21+13+9) / 4 = 13.25 ms
- Average Waiting Time: (0+15+11+5) / 4 = 7.75 ms

Non Preemptive Priority Scheduling

- Advantages:
  - Priority is considered so **critical processes can get even better response time**.
- Disadvantages:
  - **Starvation is possible for low priority processes**. It can be overcome by using technique called ‘Aging’.
  - **Aging**: **gradually increases the priority of processes** that wait in the system for a long time.

Preemptive Priority Scheduling

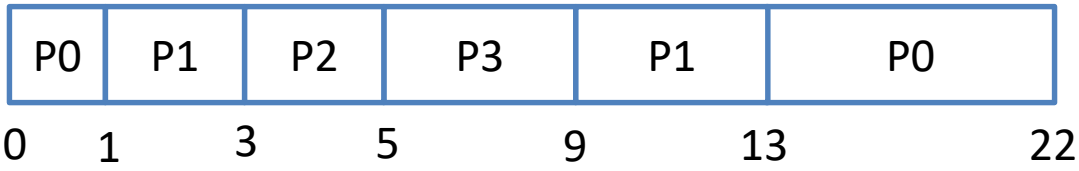
- Selection criteria :
  - The process, that **has highest priority, is served first**.
- Decision Mode:
  - **Preemptive**: When a new process arrives, its priority is compared with current process priority.
  - If the new process has higher priority than the current, the current process is suspended and new job is started.
- Implementation :
  - This strategy can be implemented by using sorted FIFO queue.
  - All processes in a queue are **sorted based on priority with highest priority process at front end**.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

Preemptive Priority Scheduling

- Example

Process	Arrival Time (T0)	Time required for completion (ΔT) (CPU Burst Time)	Priority
P0	0	10	5
P1	1	6	4
P2	3	2	2
P3	5	4	0

- Gantt Chart (small values for priority means higher priority of a process)

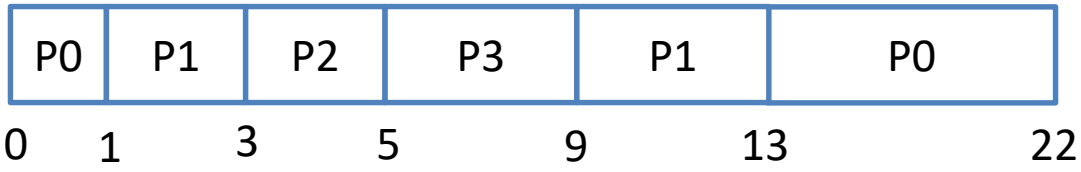


Proces	Proces	Process	Process	Process	Priority
P1	P0	P0	P0	P0	5
P0	P2	P1	P1	4	
	P1	P3	0		

Preemptive Priority Scheduling

Process	Arrival Time (T0)	Burst Time (ΔT)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart (small values for priority means higher priority of a process)



- Average Turnaround Time: (22+12+2+4) / 4 = 10 ms
- Average Waiting Time: (12+6+0+0) / 4 = 4.5 ms

Preemptive Priority Scheduling

- Advantages:
  - Priority is considered so **critical processes can get even better response time**.
- Disadvantages:
  - **Starvation is possible for low priority processes**. It can be overcome by using technique called ‘Aging’.
  - **Aging**: **gradually increases the priority of processes** that wait in the system for a long time.
  - **Context switch overhead is there**.

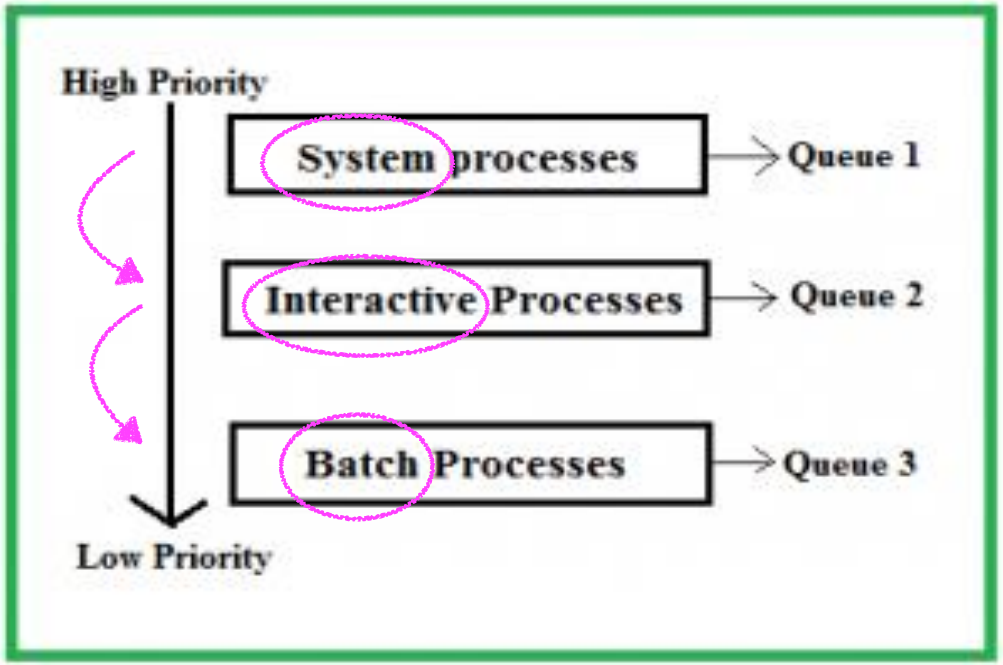
Highest Response Ratio Next (HRRN):

In this scheduling, processes with highest response ratio is scheduled. This algorithm avoids starvation.

Response Ratio = (Waiting Time + Burst time) / Burst time

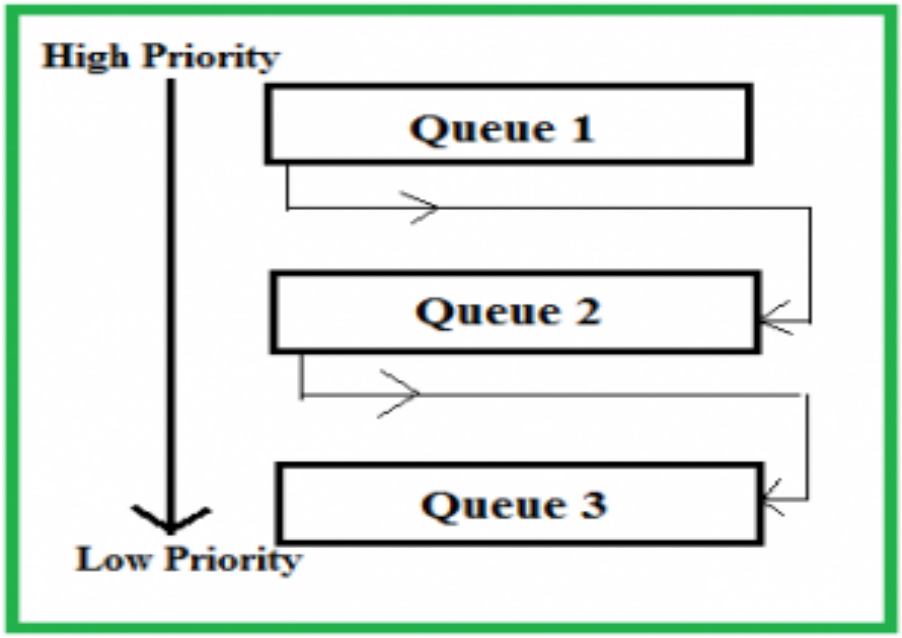
Multilevel Queue (MLQ) CPU Scheduling

- According to the priority of process, processes are placed in the different queues. Generally high priority process are placed in the top level queue.
- Only after completion of processes from top level queue, lower level queued processes are scheduled. It can suffer from starvation.



Multi level Feedback Queue Scheduling:

- It allows the process to move in between queues.
- The idea is to separate processes according to the characteristics of their CPU bursts.
- If a process uses too much CPU time, it is moved to a lower-priority queue.


High Priority
Queue 1
Queue 2
Queue 3
Low Priority

Lottery Process Scheduling

- **Lottery Scheduling** is type of process scheduling, somewhat different from other Scheduling.

- Processes are scheduled in a random manner. Lottery scheduling can be **preemptive or non-preemptive**.

- Giving each process at least one lottery ticket guarantees that it has non-zero probability of being selected at each scheduling operation.

Lottery Process Scheduling

- In this scheduling every process have some tickets and scheduler picks a random ticket

- and process having that ticket is the winner and it is executed for a time slice and then another ticket is picked by the scheduler.

- These tickets represent the share of processes. A process having a higher number of tickets give it more chance to get chosen for execution.

Lottery Process Scheduling

- We have two processes A and B. A has 60 tickets (ticket number 1 to 60) and B have 40 tickets (ticket no. 61 to 100).
- Scheduler picks a random number from 1 to 100. If the picked no. is from 1 to 60 then A is executed otherwise B is executed.
- An example of 10 tickets picked by Scheduler may look like this

```
Ticket number -      73 82 23 45 32 87 49 39 12 09
Resulting Schedule - B  B  A  A  A  B  A  A  A  A
```

Lottery Process Scheduling

- Pros: It also solves the problem of starvation.

How does a process look like in memory?

- A program loaded into memory and executing is called a process. In simple, a process is a program in execution.

- When a program is created then it is just some pieces of Bytes which is stored in Hard Disk as a passive entity.

Inactive

- Then the program starts loading in memory and become an active entity, when a program is double-clicked in windows or entering the name of the executable file on the command line. (i.e. a.out or prog.exe)

How does a process look like in memory?

- Contains temporary data i.e. function parameters, return addresses, local variables

- Dynamic allocation of memory

- Global and static variables that are initialized by the programmer prior to the execution of a program

- Executable instructions of program