PROCESS SCHEDULING

Soumya Sahoo
Assistant Professor, CSE
C.V Raman Global University, Bhubaneswar

Topics for discussion

- Motivation
- Types of scheduling
- Short-term scheduling
- Various scheduling criteria
- Various algorithms
  - Priority queues
  - First-come, first-served
  - Round-robin
  - Shortest process first
  - Shortest remaining time and others
  - Multilevel Queue
  - Multilevel feedback Queue

Process Scheduling

- In multiprogramming, several processes are kept in main memory so that when one process is busy in I/O operation, other processes are available to CPU.
- In this way, CPU is busy in executing processes at all times.
- This method of selecting a process to be allocated to CPU is called Process Scheduling.

Process Scheduling

- Process scheduling consists of the following sub-functions:
  - **Scheduling:** Selecting the process to be executed next on CPU is called scheduling.
    - In this function a process is taken out from a pool of ❖ready processes and is assigned to CPU.
    - This task is done by a component of operating system called **Scheduler**.

Main OS Process-related Goals

- Interleave the execution of existing processes to maximize processor utilization.
- Provide reasonable response times.
- Allocate resources to processes.
- Support inter-process communication (and synchronization) and user creation of processes

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 and blocked 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)

Optimization Criteria

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

Types of scheduling

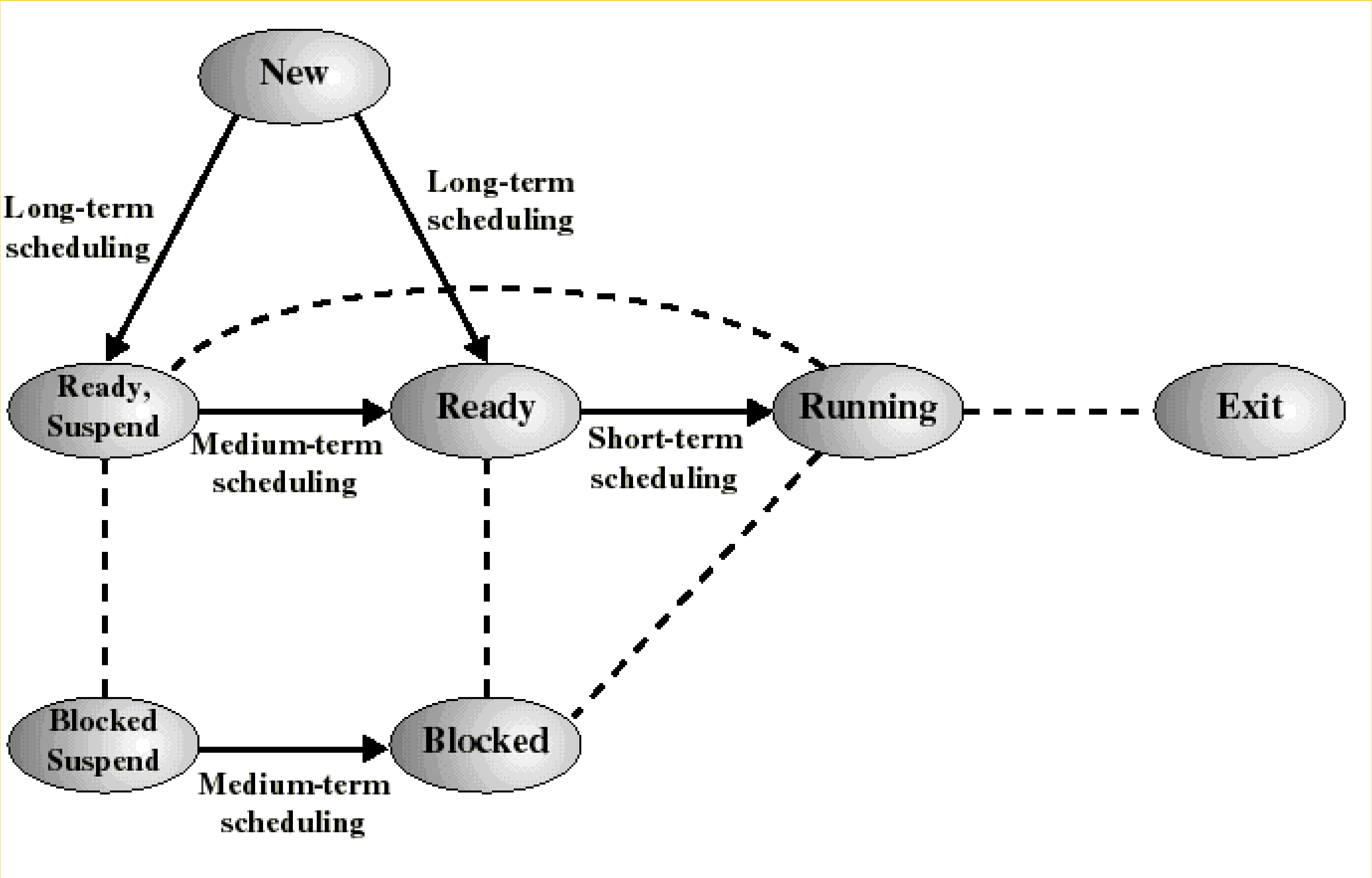
❖**Long-term** : To add to the pool of processes to be executed.

❖**Medium-term :** To add to the number of processes that are in the main memory.

❖**Short-term** : Which of the available processes will be executed by a processor?

❖IO scheduling: To decide which process's pending IO request shall be handled by an available IO device.

Classification of Scheduling Activity



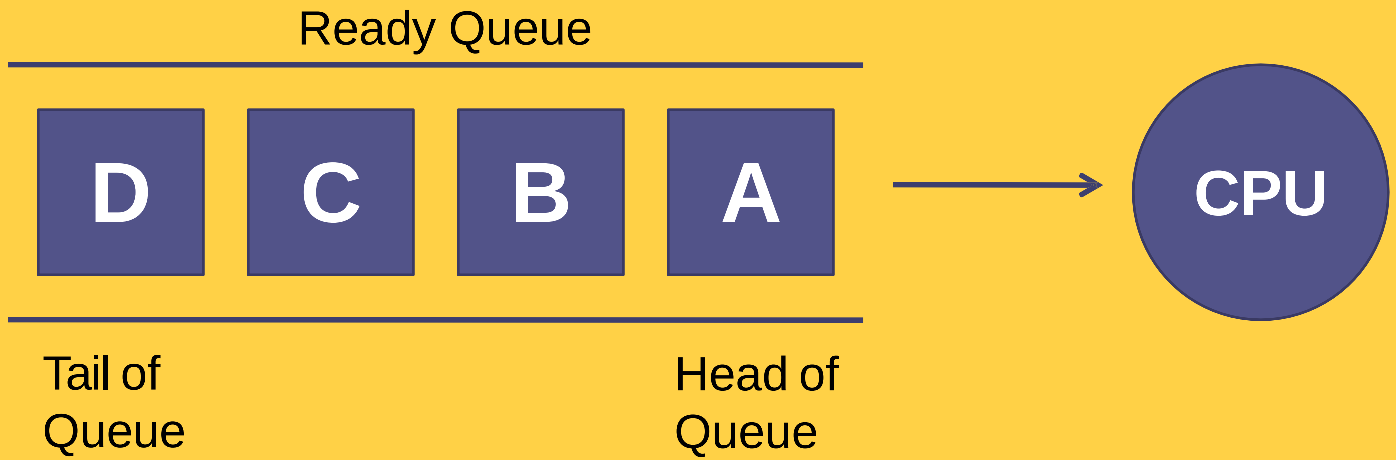
- Long-term: which process to admit
- Medium-term: which process to swap in or out
- Short-term: which ready process to execute next

Scheduling Algorithms

- First-Come-First-Served (FCFS)
- Shortest Job First (SJF)
- Priority Scheduling
- Round-Robin Scheduling (RR)
- Multi-Level Queue Scheduling (MLQ)
- Multi-Level Feedback Queue Scheduling (MFQ)

First-Come-First-Served Scheduling (FCFS)

- In this scheduling, the process that requests the CPU first, is allocated the CPU first.
- Thus, the name *First-Come-First-Served.*
- The implementation of FCFS is easily managed with a FIFO queue.



First-Come-First-Served Scheduling (FCFS)

- When a process enters the ready queue, its PCB is linked to the tail of the queue.
- When CPU is free, it is allocated to the process which is at the head of the queue.
- FCFS scheduling algorithm is ***non-preemptive***.
- Once the CPU is allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating or by I/O request.

Example of FCFS Scheduling

- Consider the following set of processes that arrive at time 0 with the length of the CPU burst time in milliseconds:

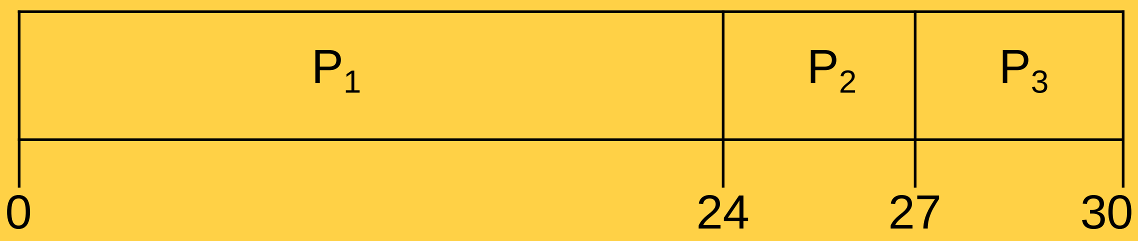
Process	Burst Time (in milliseconds)
P_1	24
P_2	3
P_3	3

Example of FCFS Scheduling

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

P_1	24
P_2	3
P_3	3

- The Gantt Chart for the schedule is:



- Waiting Time for P_1 = 0 milliseconds
- Waiting Time for P_2 = 24 milliseconds
- Waiting Time for P_3 = 27 milliseconds

Example of FCFS Scheduling

- Average Waiting Time = (Total Waiting Time) /
  - No. of Processes
  - = (0 + 24 + 27) / 3
  - = 51 / 3
  - = 17 milliseconds

Example of FCFS Scheduling

- 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_2 = 0 milliseconds
- Waiting Time for P_3 = 3 milliseconds
- Waiting Time for P_1 = 6 milliseconds

Example of FCFS Scheduling

- Average Waiting Time= (Total Waiting Time) / No. of Processes

  = (0 + 3 + 6) / 3

  = 9 / 3

  = 3 milliseconds

- Thus, the average waiting time depends on the order in which the processes arrive.

FCFS CONTD…

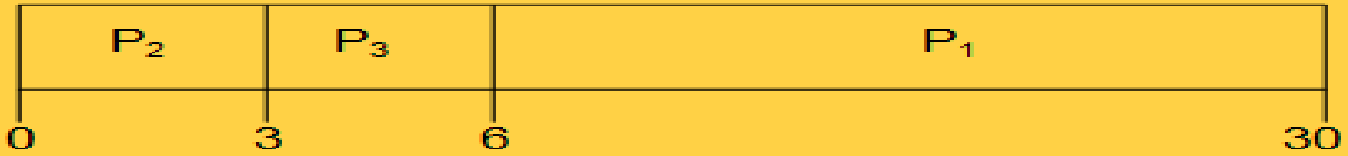
Response time = Time at which the process gets the CPU for the first time - Arrival time

The Turnaround time and the waiting time are calculated by using the following formula.

Turn Around Time = Completion Time - Arrival Time   Or ,TAT = BT + WT

Waiting Time = Turnaround time - Burst Time

- The Gantt chart for the schedule is:



TAT=P2=3-0=3,P3=6-0=6,P1=30-0=30

WT=P2=3-3=0,P3=6-3=3,P1=30-24=6

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
0	0	2	2	2	0
1	1	6	8	7	1
2	2	4	12	10	6
3	3	9	21	18	9
4	6	12	33	29	17

Avg Waiting Time=31/5

P0	P1	P2	P3	P4

0 2 8 12 21 33

Shortest Job First Scheduling (SJF)

- In SJF, the process with the least estimated execution time is selected from the ready queue for execution.
- It associates with each process, the length of its next CPU burst.
- When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
- If two processes have the same length of next CPU burst, FCFS scheduling is used.
- SJF algorithm can be preemptive or non-preemptive.

Non-Preemptive SJF

- In non-preemptive scheduling, CPU is assigned to the process with least CPU burst time.
- The process keeps the CPU until it terminates.
- **Advantage:**
  - It gives minimum average waiting time for a given set of processes.
- **Disadvantage:**
  - It requires knowledge of how long a process will run and this information is usually not available.

Preemptive SJF

- In preemptive SJF, the process with the smallest estimated run-time is executed first.
- Any time a new process enters into ready queue, the scheduler compares the expected run-time of this process with the currently running process.
- If the new process's time is less, then the currently running process is preempted and the CPU is allocated to the new process.

Example of Non-Preemptive SJF

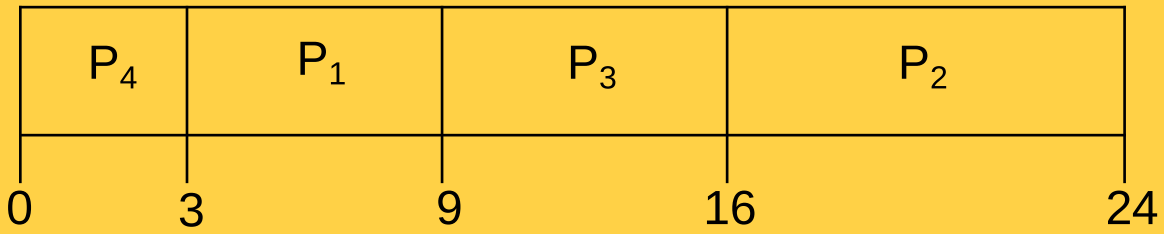
- Consider the following set of processes that arrive at time 0 with the length of the CPU burst time in milliseconds:

Process	Burst Time (in milliseconds)
P_1	6
P_2	8
P_3	7
P_4	3

Example of Non-Preemptive SJF

P_1	6
P_2	8
P_3	7
P_4	3

- The Gantt Chart for the schedule is:



- Waiting Time for P_4 = 0 milliseconds,
- Waiting Time for P_1 = 3 milliseconds
- Waiting Time for P_3 = 9 milliseconds
- Waiting Time for P_2 = 16 milliseconds

Example of Non-Preemptive SJF

- Average Waiting Time = (Total Waiting Time) /
- No. of Processes
- = (0 + 3 + 9 + 16) / 4
- = 28 / 4
- = 7 milliseconds

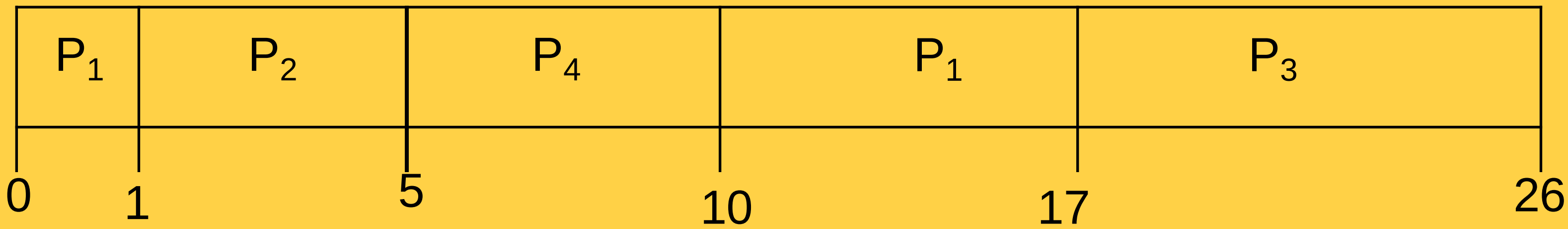
Example of Preemptive SJF

- Consider the following set of processes. These processes arrived in the ready queue at the times given in the table:

Process	Arrival Time	Burst Time (in milliseconds)
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Example of Preemptive SJF

- The Gantt Chart for the schedule is:



- Waiting Time for $P_1 = 10 – 1 – 0 = 9$
- Waiting Time for $P_2 = 1 – 1 = 0$
- Waiting Time for $P_3 = 17 – 2 = 15$
- Waiting Time for $P_4 = 5 – 3 = 2$

P	AT	BT
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Example of Preemptive SJF

- Average Waiting Time= (Total Waiting Time) /
  - No. of Processes
  - = (9 + 0 + 15 + 2) / 4
  - = 26 / 4
  - = 6.5 milliseconds

Explanation of the Example

- Process P_1 is started at time 0, as it is the only process in the queue.
- Process P_2 arrives at the time 1 and its burst time is 4 milliseconds.
- This burst time is less than the remaining time of process P_1 (7 milliseconds).
- So, process P_1 is preempted and P_2 is scheduled.

Explanation of the Example

- Process P_3 arrives at time 2. Its burst time is 9 which is larger than remaining time of P_2 (3 milliseconds).
- So, P_2 is not preempted.
- Process P_4 arrives at time 3. Its burst time is 5.
  - Again it is larger than the remaining time of P_2 (2 milliseconds).
- So, P_2 is not preempted.

Explanation of the Example

- After the termination of P_2, the process with shortest next CPU burst i.e. P_4 is scheduled.
- After P_4, processes P_1 (7 milliseconds) and then P_3 (9 milliseconds) are scheduled.

Priority Scheduling

- In priority scheduling, a priority is associated with all processes.
- Processes are executed in sequence according to their priority.
- CPU is allocated to the process with highest priority.
- If priority of two or more processes are equal than FCFS is used to break the tie.

Priority Scheduling

- Priority scheduling can be preemptive or non-preemptive.
- **Preemptive Priority Scheduling:**
  - In this, scheduler allocates the CPU to the new process if the priority of new process is higher tan the priority of the running process.
- **Non-Preemptive Priority Scheduling:**
  - The running process is not interrupted even if the new
    ❖process has a higher priority.
  - In this case, the new process will be placed at the head of the ready queue.

Priority Scheduling

- **Problem:**
  - In certain situations, a low priority process can be blocked infinitely if high priority processes arrive in the ready queue frequently.
  - This situation is known as ***Starvation***.

Priority Scheduling

- **Solution:**
  - ***Aging*** is a technique which gradually increases the priority of processes that are victims of starvation.
  - For e.g.: Priority of process X is 10.
  - There are several processes with higher priority in the ready queue.
  - Processes with higher priority are inserted into ready queue frequently.
  - In this situation, process X will face starvation.

Priority Scheduling

(Cont.):

- The operating system increases priority of a process by 1 in every 5 minutes.
- Thus, the process X becomes a high priority ❖process after some time.
- And it is selected for execution by the scheduler.

S.Sahoo,CGU

Example of Priority Scheduling

- Consider the following set of processes that arrive at time 0 with the length of the CPU burst time in milliseconds. The priority of these processes is also given:

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

S.Sahoo,CGU

Example of Priority Scheduling

- The Gantt Chart for the schedule is:

P_2	P_5	P_1	P_3	P_4

0 1 6 16 18 19

- Waiting Time for $P_2 = 0$
- Waiting Time for $P_5 = 1$
- Waiting Time for $P_1 = 6$
- Waiting Time for $P_3 = 16$
- Waiting Time for $P_4 = 18$

P	BT	Pr
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Example of Priority Scheduling

- Average Waiting Time = (Total Waiting Time) / No. of Processes

  = (0 + 1 + 6 + 16 + 18) / 5

  = 41 / 5

  = 8.2 milliseconds

S.Sahoo,CGU

Another Example of Priority Scheduling

- Processes P_1, P_2, P_3 are the processes with their arrival time, burst time and priorities listed in table below:

Process	Arrival Time	Burst Time	Priority
P_1	0	10	3
P_2	1	5	2
P_3	2	2	1

Another Example of Priority Scheduling

- The Gantt Chart for the schedule is:

P_1	P_2	P_3	P_2	P_1
0–1	1–2	2–4	4–8	8–17

Waiting Time for P1 = 0 + (8 – 1) = 7

Waiting Time for P2 = (1 -1)+ (4 – 2) = 2

Waiting Time for P3 = 2-2=0

P	AT	BT	Pr
P_1	0	10	3
P_2	1	5	2
P_3	2	2	1

Another Example of Priority Scheduling

- Average Waiting Time = (Total Waiting Time) /
- No. of Processes
- = (7 + 2 + 0) / 3
- = 9 / 3
- = 3 milliseconds

Round Robin Scheduling (RR)

- In Round Robin scheduling, processes are dispatched in FIFO but are given a small amount of CPU time.
- This small amount of time is known as ***Time Quantum*** or ***Time Slice***.
- A time quantum is generally from 10 to 100 milliseconds.

Round Robin Scheduling (RR)

- If a process does not complete before its time slice expires, the CPU is preempted and is given to the next process in the ready queue.
- The preempted process is then placed at the tail of the ready queue.
- If a process is completed before its time slice expires, the process itself releases the CPU.
- The scheduler then proceeds to the next process in the ready queue.

Round Robin Scheduling (RR)

- Round Robin scheduling is always preemptive as no process is allocated the CPU for more than one time quantum.
- If a process’s CPU burst time exceeds one time quantum then that process is preempted and is put back at the tail of ready queue.
- The performance of Round Robin scheduling depends on several factors:
  - Size of Time Quantum
  - Context Switching Overhead

Example of Round Robin Scheduling

❖Consider the following set of processes that arrive at time 0 with the length of the CPU burst time in milliseconds:

Process	Burst Time
P_1	10
P_2	5
P_3	2

Time quantum is of 2 milliseconds.

Example of Round Robin Scheduling

- The Gantt Chart for the schedule is:

P_1	P_2	P_3	P_1	P_2	P_1	P_2	P_1	P_1

0 2 4 6 8 10 12 13 15 17

- Waiting Time for $P_1 = 0 + (6 – 2) + (10 – 8) + (13 – 12)$
  - $= 4 + 2 + 1 = 7$
- Waiting Time for $P_2 = 2 + (8 – 4) + (12 – 10)$
  - $= 2 + 4 + 2 = 8$
- Waiting Time for $P_3 = 4$

P	BT
P_1	10
P_2	5
P_3	2

Example of Round Robin Scheduling

- Average Waiting Time= (Total Waiting Time) /
  - No. of Processes

= (7 + 8 + 4) / 3

= 19 / 3

= 6.33 milliseconds

S.Sahoo,CGU

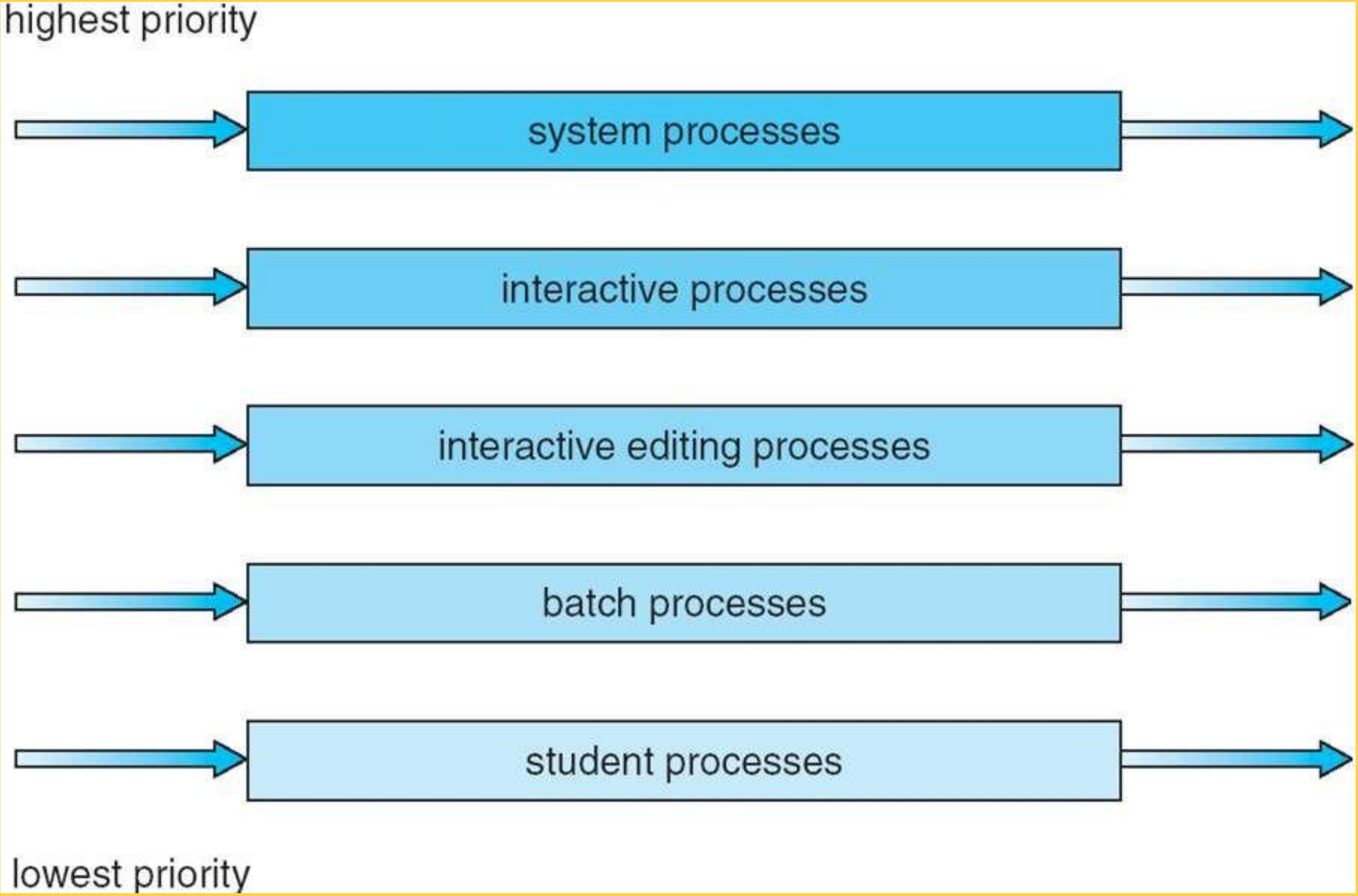
Multi-Level Queue Scheduling (MLQ)

- Multi-Level Queue scheduling classifies the processes according to their types.
- For e.g.: a MLQ makes common division between the interactive processes (foreground) and the batch processes (background).
- These two processes have different response times, so they have different scheduling requirements.
- Also, interactive processes have higher priority than the batch processes.

Multi-Level Queue Scheduling (MLQ)

- In this scheduling, ready queue is divided into various queues that are called subqueues.
- The processes are assigned to subqueues, based on some properties like memory size, priority or process type.
- Each subqueue has its own scheduling algorithm.
- For e.g.: interactive processes may use round robin algorithm while batch processes may use FCFS.

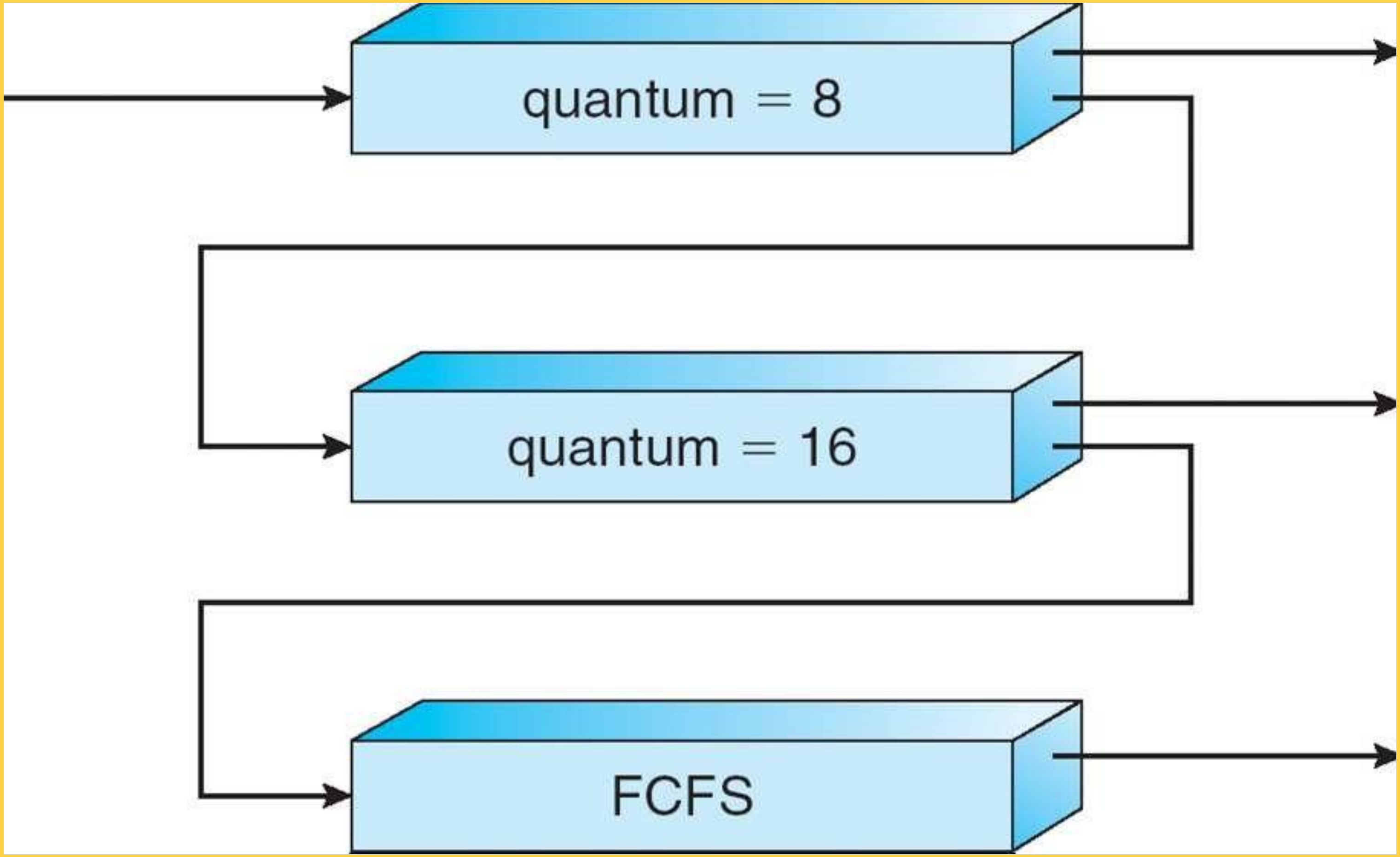
Multi-Level Queue Scheduling (MLQ)



Multi-Level Feedback Queue Scheduling (MFQ)

- Multi-Level Feedback Queue scheduling is an enhancement of MLQ.
- In this scheme, processes can move between different queues.
- The various processes are separated in different
  ❖queues on the basis of their CPU burst times.
- If a process consumes a lot of CPU time, it is placed into a lower priority queue.
- If a process waits too long in a lower priority queue, it is moved into higher priority queue.
- Such an *aging* prevents starvation.

Multi-Level Feedback Queue Scheduling (MFQ)



Multi-Level Feedback Queue Scheduling (MFQ)

- The top priority queue is given smallest CPU time quantum.
- If the quantum expires before the process terminates, it is then placed at the back of the next lower queue.
- Again, if it does not complete, it is put to the last priority queue.
- The processes in this queue runs on FCFS scheduling.
- If a process becomes a victim of starvation, it is promoted to the next higher priority queue.

THANK YOU