
PART 3

# CPU SCHEDULING ALGORITHMS

Dr. Soha S. Zaghloul

LAYOUT

- CPU Scheduling Algorithms (cont'd)
  - Priority-Based
  - Round-Robin
  - Multi-Level Queue
  - Multi-Level Feedback Queue

# PRIORITY-BASED SCHEDULING ALGORITHM (1)

The CPU is allocated to processes based on their assigned priority.

Priority-based scheduling algorithm is implemented as follows:

- Each process is assigned a priority. This is saved in its PCB.
- The CPU is allocated to the process with the highest priority.
- A process with a priority 0 has the highest priority.
- Equal priority processes are assigned to the CPU in FCFS order

Priority-based scheduling has the following properties

- (1) May be applied as a preemptive or non-preemptive scheduling:

  Preemptive Priority-Based:

  - When a process *Pi* is appended to the Ready Q, its priority is compared against that of the currently running process *Pj*.
  - If the priority of *Pi* is higher, then *Pj* is preempted after completing the execution of the current instruction.
  - In other words, the arrival of a process to the Ready Q is an event.

  Non-Preemptive Priority-Based:

  - The newly added process *Pi* is simply appended to the Ready Q without checking the priority of the running process *Pj*.
  - When *Pj* voluntarily deallocates the CPU, the priority-based scheduling algorithm is applied on all processes of the Ready Queue
  - In other words, the arrival of a process to the Ready Q is not an event.

# PRIORITY-BASED SCHEDULING ALGORITHM (2)

Priority-based scheduling also has the following properties:

- (2) Priorities may be assigned in one of two ways:
  - Internally-defined priority
    - A measurable quantity is used to define the process priority.
    - Examples include the length of the next CPU burst, memory requirements, or number of open files.
  - Externally-defined priority
    - Set outside of the OS by the System Administrator.
    - The System Administrator may define a process priority based on its importance, type and amount of funds being paid for computer use, political factors, etc…

The main advantage of the Priority-Based Scheduling Algorithm lies in its simplicity.

The disadvantages of the algorithm may be summarized in the following:

- The starvation problem:
  - Low-priority processes may wait infinitely.
  - This is solved by applying aging.
    - Each process is provided with a counter to its waiting time in the Ready Queue.
    - When the waiting time exceeds a pre-determined threshold, the process priority is incremented periodically

## EXAMPLE ON AGING

**Problem:**

Assume that a process *P1* has a priority of 127. The OS is configured to increment the process priority every 15 seconds. When will the priority of the process be equal to zero?

**Solution:**

*P1* will be incremented (127 – 0) times.

➔ *Time needed = 127 * 15 = 1,905 seconds = 31.75 minutes*

# EXAMPLE ON NON-PREEMPTIVE PRIORITY-BASED SCHEDULING ALGORITHM

**Given: The algorithm is non-preemptive**

| Process Number | Burst Time (in ms) | Priority |
|---|---|---|
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 4 |
| P4 | 1 | 5 |
| P5 | 5 | 2 |

**Result:**

Waiting Time
6
0
16
18
1

Average Waiting Time = (6 +0+16+18+1)/5 = 8.2 ms

**Required:**

Calculate the average waiting time using the Gantt Chart.

**Solution:**

| Process Number | Priority | Burst Time |
|---|---|---|
| P2 | 1 | 1 |
| P5 | 2 | 5 |
| P1 | 3 | 10 |
| P3 | 4 | 2 |
| P4 | 5 | 1 |

# ROUND-ROBIN (RR) SCHEDULING ALGORITHM (1)

The RR Scheduling Algorithm is implemented as follows:

- A small unit of time, called a time quantum or time slice is defined.
- The Ready Queue is handled like a circular queue.
- The CPU is allocated to each process for a time interval of up to 1 time quantum.

RR has the following properties:

- RR is similar to the FCFS, but the preemptive property is added.
- Specially designed for time-sharing systems.
- The performance depends heavily on the value of the time quantum:
  - If the time quantum is too large, then probably no preemption will occur and the RR would work similar to FCFS.
  - If the time quantum is too small, then context switching would frequently occur leading to an overhead in the whole system.
  - In modern OSs, the time quantum ranges from 10 to 100 ms.
  - In general, the time quantum is selected to be large enough with respect to the context switch time.
  - If the context switch time is approximately 10% of the time quantum, then about 10% of the CPU time is spent in context switching.

# ROUND-ROBIN (RR) SCHEDULING ALGORITHM (2)

The advantages of RR Scheduling Algorithm may be summarized as:

- Simple to implement
- Ensures fair shares to multi-users

RR has the following disadvantages:

- The average waiting time is often long
- If there is only one process in the Ready Queue, it will be needlessly preempted at the end of each time quantum.

# ROUND-ROBIN (RR) – EXAMPLE

| Given: Time quantum = 4 ms. Arrival time= 0 for all. | Result: | |
|---|---|---|
| Process Number Burst Time (in ms)<br>*P1* *24*<br>*P2* *3*<br>*P3* *3* | Waiting Time<br>*10 - 4 = 6*<br>*4*<br>*7* | Average Waiting Time =<br>*(6 + 4 + 7)/3*<br>*= 5.66 ms* |

**Required:**

Calculate the average waiting time using the Gantt Chart. Ignore context switch time.

**Solution:**


| Remaining Time | | |
|---|---|---|
| P1 | P2 | P3 |
| 20 | 3 | 3 |
| 20 | 0 | 3 |
| 20 | 0 | 0 |
| 16 | 0 | 0 |
| 12 | 0 | 0 |
| 8 | 0 | 0 |
| 4 | 0 | 0 |
| 0 | 0 | 0 |

## ROUND-ROBIN (RR) – EXAMPLE WITH CONTEXT SWITCHING


If the context switch is taken into consideration:

- The context switch is added after the end of each quantum time
- The context switch is added even there is only one process in the Ready Q
- The context switch time (given) is added to the waiting time of each process

# MULTI-LEVEL QUEUE (MLQ) SCHEDULING (1)

The MLQ is implemented as follows:

- Processes are divided into categories based on their scheduling needs.
- Each category is permanently assigned to one queue.
- Each queue has its own scheduling algorithm
- In addition, there is scheduling among queues: this is commonly implemented as fixed-priority preemptive scheduling.
- No process in the low-priority queue can run unless the higher-priority Q is empty.
- If a process *Pi* enters a high-priority Q while the CPU is running a process *Pj* from a lower priority, then *Pj* finishes the current instruction, preempted to leave room to the high-priority one.
- The preempted process *Pj* returns to its original queue.
- Another possibility for queue scheduling is to assign a quantum time for each queue.

Therefore, two levels of scheduling algorithms should be defined:

- The first level is amongst the queues.
- The second level is amongst the processes in the same queue.
- Multiple algorithms may be assigned at this level to different queues

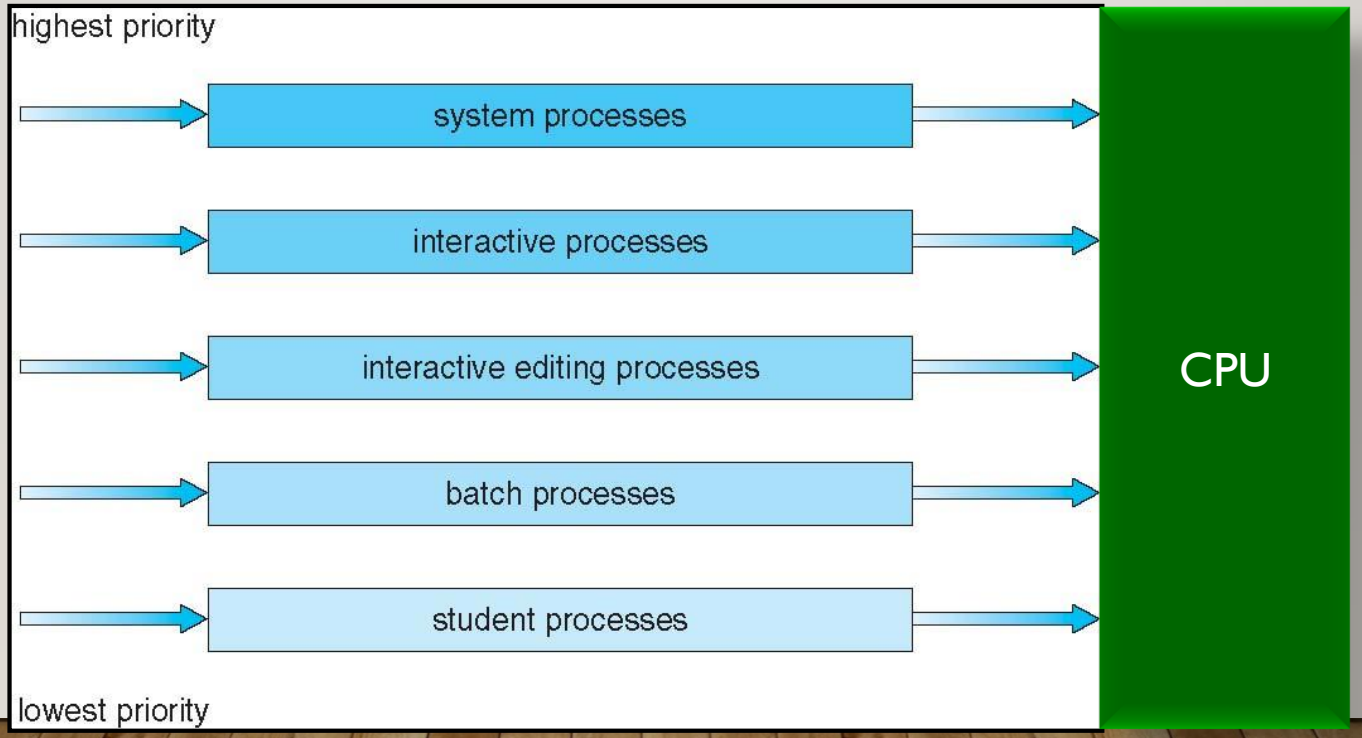
# MULTI-LEVEL QUEUE (MLQ) SCHEDULING ALGORITHM (2)

The main advantage of MLQ is that it ensures that important processes are handled first before the low-priority ones.

The disadvantages of MLQ may be summarized in:

- Inflexible: A process cannot change its assigned queue; ie. its priority.
- Processes in the lowest priority queue suffer from starvation.

The following figure illustrates the MLQ algorithm:



# MULTI-LEVEL FEEDBACK QUEUE (MLFQ) SCHEDULING (1)

The MLFQ is implemented as follows:

- Like MLQ, multiple Ready Queues are provided.
- A process just arriving at the Ready Q is appended to the highest-priority Q.
- Each queue is given its own quantum time.
- When the running process is preempted or interrupted, it returns to the next lower priority queue: it does not return to the same Q it came from.
- Low-priority queues are investigated only if the higher priority ones are empty.
- If a process waits for too long time in a lower-priority queue, it is moved upwards to prevent starvation. This is a different implementation of aging.

Therefore, two levels of scheduling algorithms should be defined:

- The first level is amongst the queues.
- The second level is amongst the processes in the same queue.
- Multiple algorithms may be assigned at this level to different queues

MLFQ implementation gives priority to the processes with CPU burst time less than or equal to the time quantum of the highest priority queue.

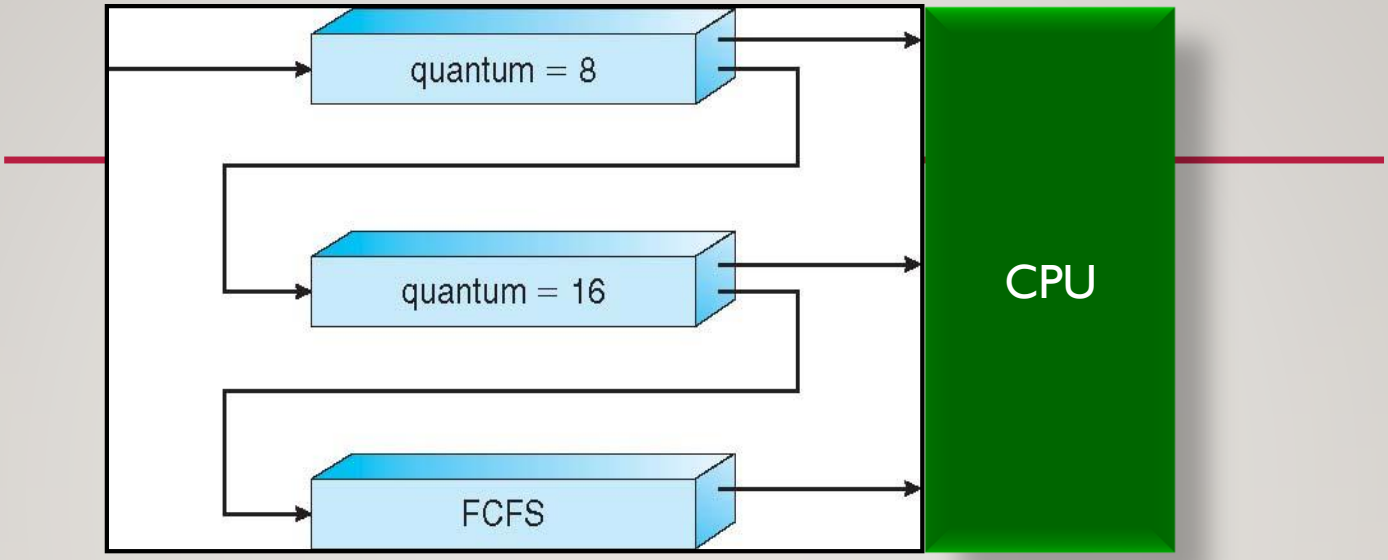
The main advantages of MLFQ are:

- Flexible
- Avoids starvation

The main disadvantage of MLFQ lies in its complexity.

# MULTI-LEVEL FEEDBACK QUEUE (MLFQ) SCHEDULING ALGORITHM (2)

Consider the following figure:




In the example above:

- The highest priority Q is given a quantum time of 8 ms.
- The next priority Q is given a quantum time of 16 ms.
- The lowest priority Q is served as FCFS. No quantum time should be given here.

Therefore, we have the following:

- Processes with CPU burst time <= 8 ms take the most advantage.
- These are followed by those with CPU burst time is between 8 and 16.

As a conclusion, this algorithm is suitable for I/O-bound processes since they have short CPU bursts.