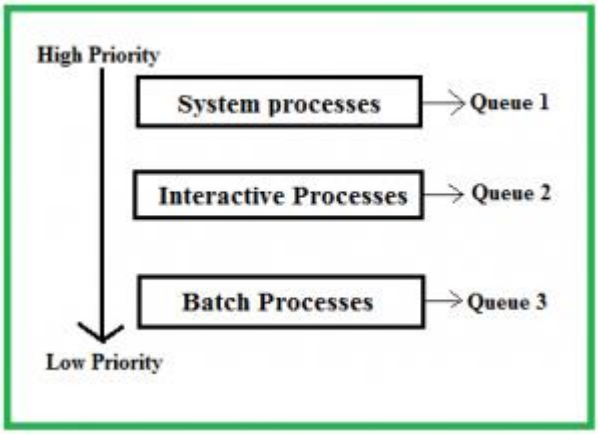
Multilevel Queue Scheduling

Each algorithm supports a different process, but in a general system, some processes require scheduling using a priority algorithm. While some processes want to stay in the system ***(interactive processes),*** others are ***background processes*** whose execution can be delayed.

The number of ready queue algorithms between queues and within queues may differ between systems. A round-robin method with various time quantum is typically utilized for such maintenance. Several types of scheduling algorithms are designed for circumstances where the processes can be readily separated into groups. There are two sorts of processes that require different scheduling algorithms because they have varying response times and resource requirements. ***The foreground (interactive)*** and ***background processes*** (batch process) are distinguished. Background processes take priority over foreground processes.

The ready queue has been partitioned into seven different queues using the multilevel queue scheduling technique. These processes are assigned to one queue based on their priority, such as memory size, process priority, or type. The method for scheduling each queue is different. Some queues are utilized for the foreground process, while others are used for the background process. The ***foreground queue may*** be scheduled using a ***round-robin method,*** and the ***background queue can*** be scheduled using an ***FCFS*** strategy.

Ready Queue is divided into separate queues for each class of processes. For example, let us take three different types of processes System processes, Interactive processes, and Batch Processes. All three processes have their own queue. Now, look at the below figure.



The Description of the processes in the above diagram is as follows:

System Processes: The CPU itself has its own process to run which is generally termed as System Process.

Interactive Processes: An Interactive Process is a type of process in which there should be same type of interaction.

Batch Processes: Batch processing is generally a technique in the Operating system that collects the programs and data together in the form of the batch before the processing starts.

All three different type of processes has 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.

Scheduling among the queues: What will happen if all the queues have some processes? Which process should get the CPU? To determine this Scheduling among the queues is necessary. 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.

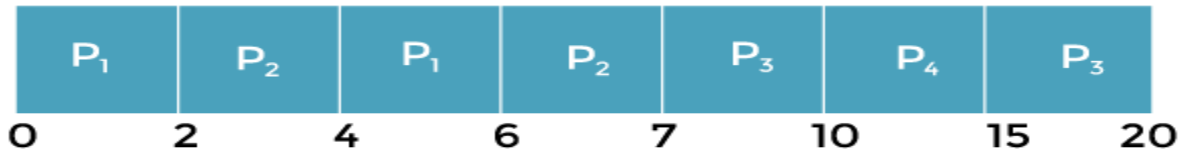
2. **Time slicing** – In this method, each queue gets a certain portion of CPU time and can use it to schedule its own processes. For instance, queue 1 takes 50 percent of CPU time queue 2 takes 30 percent and queue 3 gets 20 percent of CPU time.

Example Problem

Let's take an example of a multilevel queue-scheduling (MQS) algorithm that shows how the multilevel queue scheduling work. Consider the four processes listed in the table below under multilevel queue scheduling. The queue number denotes the process's queue.

Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

Queue 1 has a higher priority than queue **2.** Round Robin is used in queue **1 (Time Quantum = 2),** while FCFS is used in queue **2.**



Working:

1. Both queues have been processed at the start. Therefore, **queue 1 (P1, P2)** runs first (due to greater priority) in a round-robin way and finishes after 7 units.
2. The process in **queue 2 (Process P3)** starts running (since there is no process in queue 1), but while it is executing, P4 enters queue 1 and interrupts P3, and then P3 takes the CPU and finishes its execution.

Advantages and Disadvantages of Multilevel Queue Scheduling

There are various advantages and disadvantages of multilevel queue scheduling. Some of the advantages and disadvantages of multilevel queue scheduling are as follows:

Advantages

1. You can use multilevel queue scheduling to apply different scheduling methods to distinct processes.
2. It will have low overhead in terms of scheduling.

Disadvantages

1. There is a risk of starvation for lower-priority processes.
2. It is rigid in nature.

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling is like Multilevel Queue(MLQ) Scheduling but in this processes can move between the queues. And thus, much more efficient than multilevel queue scheduling.

Characteristics of Multilevel Feedback Queue Scheduling:

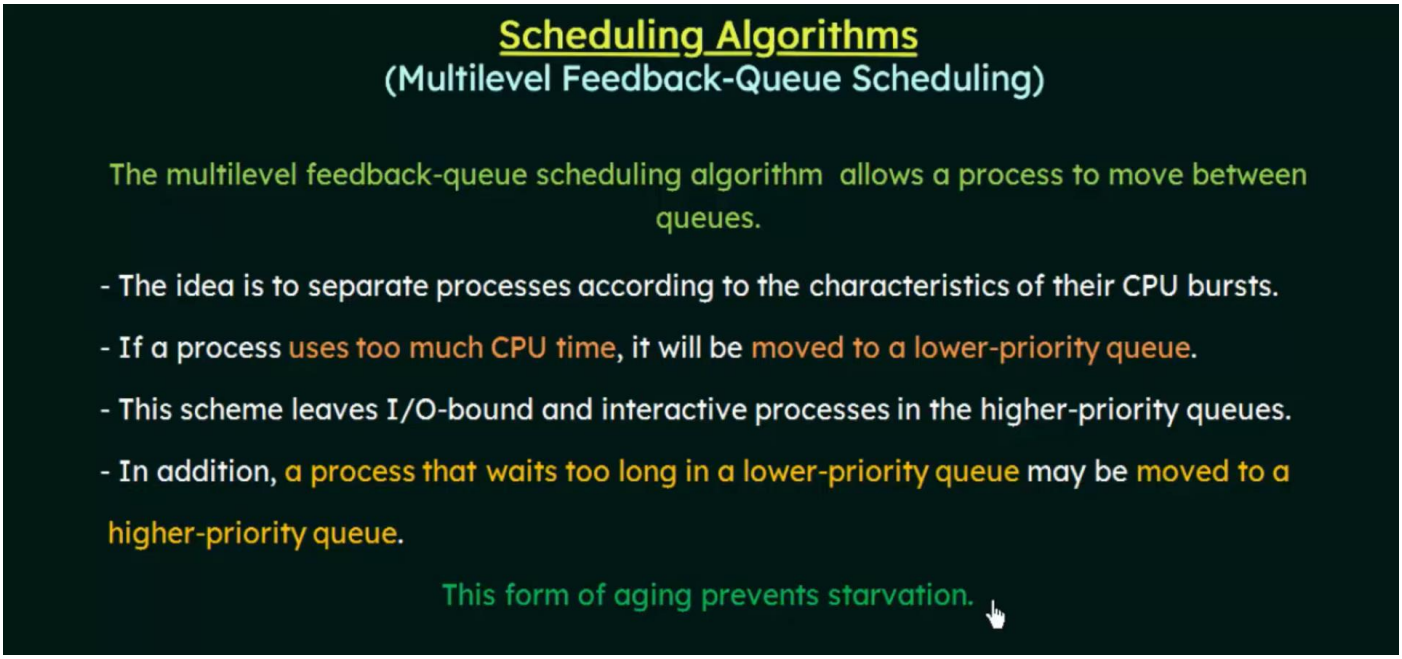
- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system and processes are allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,

Advantages of Multilevel Feedback Queue Scheduling:

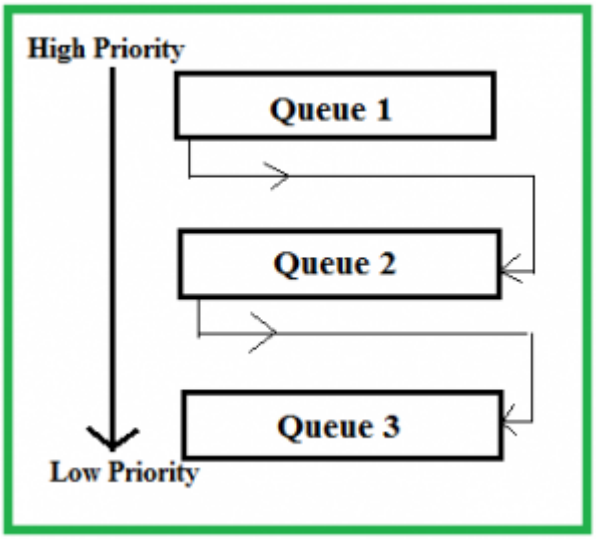
- It is more flexible.
- It allows different processes to move between different queues.
- It prevents starvation by moving a process that waits too long for the lower priority queue to the higher priority queue.

Disadvantages of Multilevel Feedback Queue Scheduling:

- For the selection of the best scheduler, it requires some other means to select the values.
- It produces more CPU overheads.
- It is the most complex algorithm.



Multilevel feedback queue scheduling, however, allows a process to move between queues. Multilevel Feedback Queue Scheduling **(MLFQ)** keeps analyzing the behavior (time of execution) of processes and according to which it changes its priority.
Now, look at the diagram and explanation below to understand it properly.



Now let us suppose that queues 1 and 2 follow round robin with time quantum 4 and 8 respectively and queue 3 follow FCFS.
Implementation of MFQS is given below –

- When a process starts executing the operating system can insert it into any of the above three queues depending upon its **priority**. For example, if it is some background process, then the operating system would not like it to be given to higher priority queues such as queue 1 and 2. It will directly assign it to a lower priority queue i.e. queue 3. Let's say our current process for consideration is of significant priority so it will be given **queue 1**.
- In queue 1 process executes for 4 units and if it completes in these 4 units or it gives CPU for I/O operation in these 4 units then the priority of this process does not change and if it again comes in the ready queue then it again starts its execution in Queue 1.
- If a process in queue 1 does not complete in 4 units then its priority gets reduced and it shifted to queue 2.
- Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 units. In a general case if a process does not complete in a time quantum then it is shifted to the lower priority queue.
- In the last queue, processes are scheduled in an FCFS manner.
- A process in a 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.

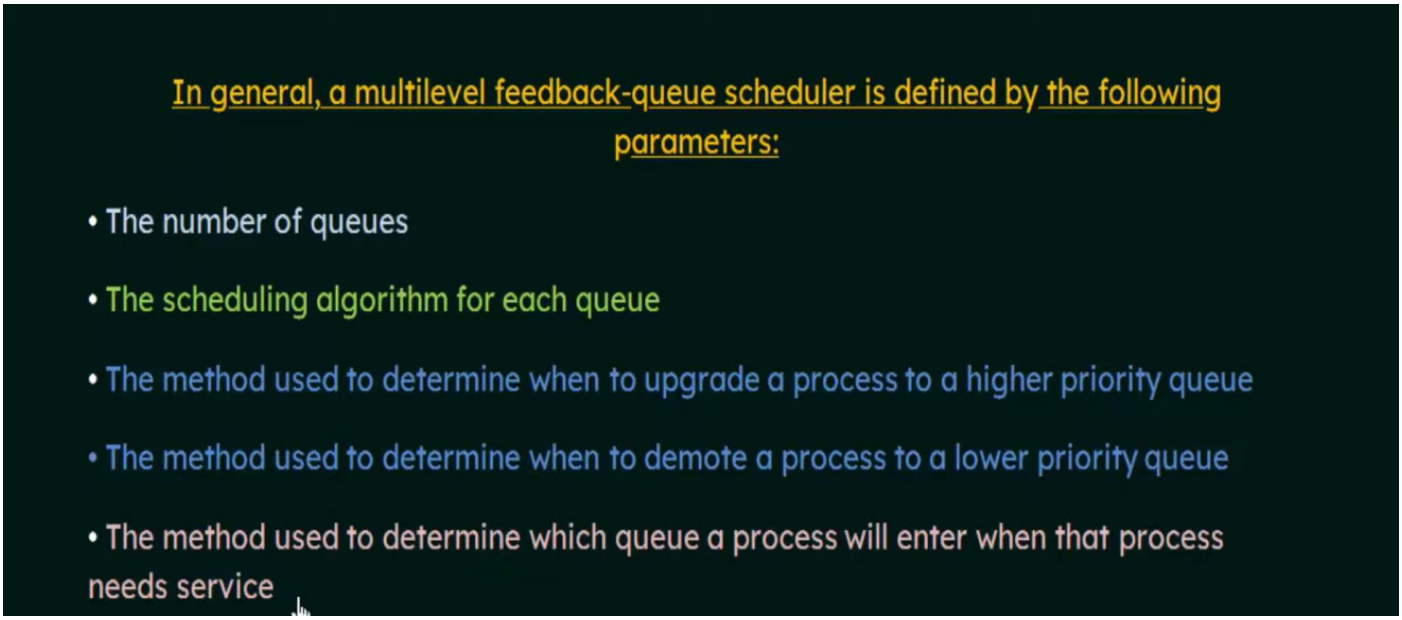
Well, the above implementation may differ for example the last queue can also follow Round-robin Scheduling.

Problems in the above implementation: *A process in the lower priority queue can suffer from starvation due to some short processes taking all the CPU time.*

Solution: *A simple solution can be to boost the priority of all the processes after regular intervals and place them all in the highest priority queue.*

What is the need for such complex Scheduling?

- Firstly, it is more flexible than multilevel queue scheduling.
- To optimize turnaround time algorithms like SJF are needed which require the running time of processes to schedule them. But the running time of the process is not known in advance. MFQS runs a process for a time quantum and then it can change its priority(if it is a long process). Thus it learns from past behavior of the process and then predicts its future behavior. This way it tries to run a shorter process first thus optimizing turnaround time.
- MFQS also reduces the response time.



Example: Consider a system that has a CPU-bound process, which requires a burst time of 40 seconds. The multilevel Feed Back Queue scheduling algorithm is used and the queue time quantum '2' seconds and in each level it is incremented by '5' seconds. Then how many times the process will be interrupted and in which queue the process will terminate the execution?

Solution:

- Process P needs 40 Seconds for total execution.
- At Queue 1 it is executed for 2 seconds and then interrupted and shifted to queue 2.
- At Queue 2 it is executed for 7 seconds and then interrupted and shifted to queue 3.
- At Queue 3 it is executed for 12 seconds and then interrupted and shifted to queue 4.
- At Queue 4 it is executed for 17 seconds and then interrupted and shifted to queue 5.
- At Queue 5 it executes for 2 seconds and then it completes.
- Hence the process is interrupted 4 times and completed on queue 5.