

**19AIE202**

**OPERATING SYSTEMS**

**GROUP PROJECT**

REPORT

**TOPIC**

MULTI-LEVEL QUEUE SCHEDULING

**SUBMITTED BY**

**BATCH – 13**

RISHEKESAN S V - CB.EN.U4AIE20058

KARTHIKEYAN M - CB.EN.U4AIE20029

KARTHIK RAJA B - CB.EN.U4AIE20028

NALIN M R - CB.EN.U4AIE20041

PRAGADISHWARAN K - CB.EN.U4AIE20051

**ACKNOWLEGEMENT**

We would like to thank to our professor B. Ganga Gowri who gave us his valuable suggestions and ideas when we needed them to work on our project

“MULTI-LEVEL QUEUE SCHEDULING”

We are also grateful to our university, Amrita Vishwa Vidyapeetham for giving us the opportunity to work on this project. Last but not the least we would like to thank our group members, Nalin MR, Karthikeyan.M, Karthik Raja.B, Pragadishwaran K, Rishekesan S V without their effort and co-operation, we would not have been able to complete the project within the prescribed time.

**ABSTRACT**

With our project “**Multilevel queue scheduling**” we were able to learn many new concept and implementation of different scheduling methods. For this project, each teammate tried to implement three different scheduling method for multilevel queue scheduling. We have tried **Round Robin**, **FCFS**, **Priority Scheduling** algorithms. Through this assignment we got an opportunity to become fluent in C and C++.

It may happen that processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, 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. Now, let us see how it works.

**TABLE OF CONTENTS**

**S. no TITLE Page No.**

1 INTRODUCTION 5

2 ROUND ROBIN 7

3 FCFS 8

4 PRIORITY SCHEDULING 9

5 MULTILEVEL QUEUE SCHEDULING 10

5 CONCLUSION 13

**INTRODUCTION**

* CPU Scheduling is a process of determining which process will own CPU for execution while another process is on hold.
* The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution.
* The selection process will be carried out by the CPU scheduler. It selects one of the processes in memory that are ready for execution.

As we know, a process needs CPU time and I/O time both for its execution. In a multi-programming system, one process can use CPU while another process is waiting for I/O whereas, on the other hand in a uni programming system, all the time get wasted in waiting for I/O whereas CPU is free during that time.

Here are the reasons for using a scheduling algorithm:

* The CPU uses scheduling to improve its efficiency.
* It helps you to allocate resources among competing processes.
* The maximum utilization of CPU can be obtained with multi-programming.
* The processes which are to be executed are in ready queue

**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

**Response time** – amount of time it takes from when a request was submitted until the first response is produced

Diagram

Description automatically generated

A CPU scheduling algorithm tries to maximize and minimize the above conditions.

**CPU SCHEDULING ALGORITHM:**

Types of CPU scheduling Algorithm

There are mainly six types of process scheduling algorithms

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

**ROUND ROBIN :-**

**Round-robin (RR)** is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices (also known as time quanta) are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive). Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an operating system concept.

The algorithm's name is derived from the round-robin principle, which is used in various disciplines to ensure that everyone gets an equal share of something.

**WORKING :-**

* To schedule processes fairly, a round-robin scheduler generally employs time -sharing, giving each job a time slot or quantum (its allowance of CPU time), and interrupting the job if it is not completed by then.
* The job is resumed next time a time slot is assigned to that process. If the process terminates or changes its state to waiting during its attributed time quantum, the scheduler selects the first process in the ready queue to execute. In the absence of time-sharing, or if the quanta were large relative to the sizes of the jobs, a process that produced large jobs would be favoured over other processes.
* Round-robin algorithm is a pre-emptive algorithm as the scheduler forces the process out of the CPU once the time quota expires.

**FIRST-COME FIRST SERVE :-**

**First Come First Serve (FCFS)** is an operating system scheduling algorithm that automatically executes queued requests and processes in order of their arrival. It is the easiest and simplest CPU scheduling algorithm. In this type of algorithm, processes which requests the CPU first get the CPU allocation first. This is managed with a FIFO queue. The full form of FCFS is First Come First Serve.

As the process enters the ready queue, its PCB (Process Control Block) is linked with the tail of the queue and, when the CPU becomes free, it should be assigned to the process at the beginning of the queue.

The job which comes first in the ready queue will get the CPU first. The lesser the arrival time of the job, the sooner will the job get the CPU. FCFS scheduling may cause the problem of starvation if the burst time of the first process is the longest among all the jobs.

**WORKING :-**

**Step 1** **:** Input the number of processes required to be scheduled using FCFS, burst time for each process and its arrival time.

**Step 2 :** Using enhanced bubble sort technique, sort the all given processes in ascending order according to arrival time in a ready queue.

**Step 3 :** Calculate the Finish Time, Turn Around Time and Waiting Time for each process which in turn help to calculate Average Waiting Time and Average Turn Around Time required by CPU to schedule given set of process using FCFS.

**Step 4 :** Process with less arrival time comes first and gets scheduled first by the CPU.

**Step 5 :** Calculate the Average Waiting Time and Average Turn Around Time.

  Turn Around Time = Completion Time - Arrival Time

   Waiting Time = Turnaround time - Burst Time

It is easy to implement and use. However, this method is poor in performance, and the general wait time is quite high.

**PRIORITY SCHEDULING**

* A CPU algorithm that schedules processes based on priority.
* It used in Operating systems for performing batch processes.
* If two jobs having the same priority are READY, it works on a FIRST COME, FIRST SERVED basis.
* In priority scheduling, a number is assigned to each process that indicates its priority level.
* Lower the number, higher is the priority.
* In this type of scheduling algorithm, if a newer process arrives, that is having a higher priority than the currently running process, then the currently running process is preempted.

**WORKING :-**

* First input the processes with their arrival time, burst time and priority.
* First process will schedule, which have the lowest arrival time, if two or more processes will have lowest arrival time, then whoever has higher priority will schedule first.
* Now further processes will be schedule according to the arrival time and priority of the process. (Here we are assuming that lower the priority number having higher priority). If two process priority are same then sort according to process number.   
  Note: In the question, They will clearly mention, which number will have higher priority and which number will have lower priority.
* Once all the processes have been arrived, we can schedule them based on their priority.

**MULTI-LEVEL QUEUE SCHEDULING**

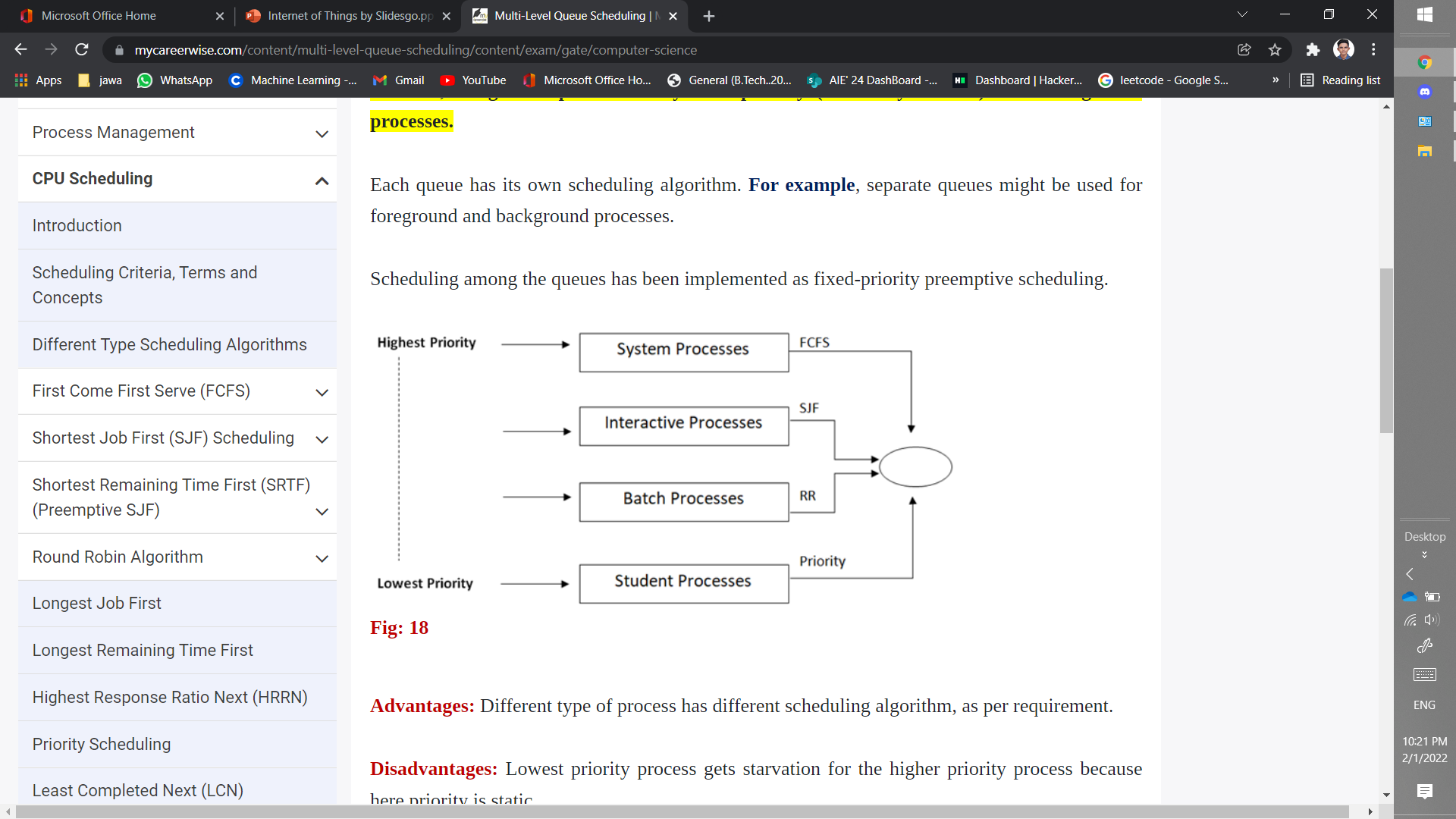
It's possible that processes in the ready queue will be separated into multiple classes, each with their own scheduling requirements. A frequent division is between a front (interactive) process and a background (batch) process, for example. Multilevel Queue Scheduling is employed in this case.

In this scheduling each process are easily classified into different groups. For example, a common division is made between foreground (interactive) process and background (batch) processes. These two types of processes have different response-time requirements and so may have different scheduling needs.

A multilevel queue scheduling algorithm partitions the ready queue into several separate queues .

Processes are permanently assigned to one queue as per group, based on some property.

Multilevel queue scheduling is used when processes in the ready queue can be divided into different classes where each class has its own scheduling needs.



Multilevel Queue Scheduling  classifies the processes according to their types. For example, a multilevel queue scheduling algorithm makes a common division between the interactive processes (foreground) and batch processes (background). These two processes have different response times, so they have different scheduling requirements. Also, the interactive process has higher priority than the batch process.

In this scheduling, ready queue is divided into various queues that are called sub queues. A sub queue is a distinct operational queue.

The method separates the ready queue into various separate queues is Multilevel Queue. The processes are permanently assigned to subqueues, generally based on some property of the process such as memory size, priority or process type.

Each sub queue has its own scheduling algorithm. For example, interactive processes at the foreground may use round robin scheduling while batch jobs at the background may use the FCFS  method.

This scheduling partitions the ready queue in several separate queues and processes are permanently assigned to one queues. It is a common practice to associate some priority depending upon where the process may have or originated.

For instance, systems programs may have a higher priority over the user programs.

The processes are permanently assigned to one another, based on some property of the process, such as

* Memory size
* Process priority
* Process type

Algorithm choose the process from the occupied queue that has the highest priority, and run that process either preemptive or non-preemptively Each queue has its own scheduling algorithm or policy.

**ADVANTAGE OF MULTI-LEVEL QUEUE SCHEDULING :-**

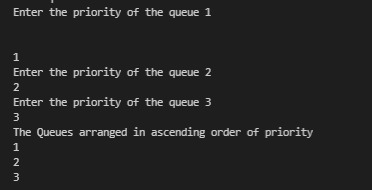
* Since processes do not move between queues so, this policy has the advantage of low scheduling overhead
* It covers all the disadvantages of all other scheduling algorithms such as overhead during context switching, low throughput
* Enables short CPU-bound jobs to be prioritized and therefore processed quickly
* Can be preemptive or non-preemptive
* Flexible implementation with respect to movement between queues

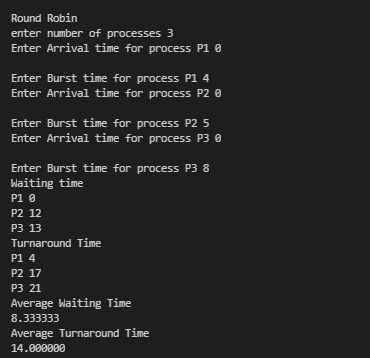
**DISADVANTAGE OF MULTI-LEVEL QUEUE SCHEDULING :-**

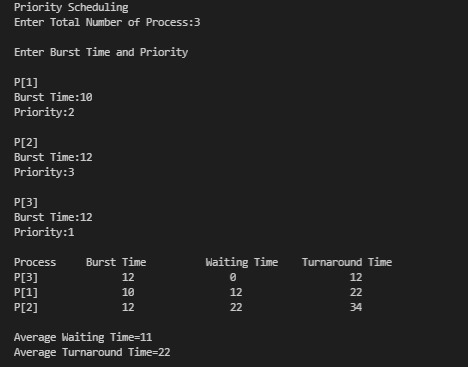
* Running process can be preempted from a low priority queue when process arrives at high priority queue. Scheduling can lead to starvation.
* If dynamically process is entering in system queue, then other processes are lead to starvation. Since we have scheduled the process queue as high priority and low priority, neither process can jump across the queue nor process can switch over.
* The process needs to be assigned to the most suitable priority queue *a* priori. If a CPU-bound process is assigned to a short-quantum, high-priority queue, that’s not optimal for either the process nor for overall throughput.

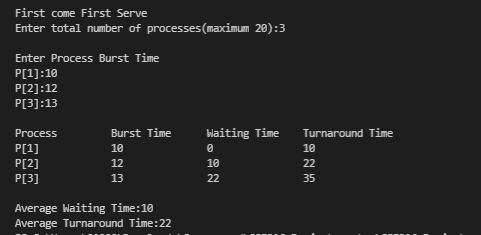
These disadvantages are overcome by multilevel feedback queue scheduling

**CODE OUTPUT**

****

****

****

****

**CONCLUSION**

We have used different algorithm to implement **multilevel queue scheduling**.

We were able to compare the advantages and disadvantages of those algorithms.

Since processes do not move between queues so, this policy has the advantage of low scheduling overhead

It covers all the disadvantages of all other scheduling algorithms such as overhead during context switching, low throughput

• Enables short CPU-bound jobs to be prioritized and therefore processed quickly

Can be preemptive or non-preemptive

• Flexible implementation with respect to movement between queues

**REFERENCE**

1. <https://www.i2tutorials.com/os-introduction/os-multilevel-queue-scheduling/>
2. <https://en.wikipedia.org/wiki/Multilevel_queue>
3. <https://www.cs.sjtu.edu.cn/~fwu/teaching/res/CS307-handout05.pdf>