Scheduling Algorithms

Hari Prasad Reddy Sheelam
Computer Engineering
University of North Texas
Texas, US
hariprasadreddysheelam@my.unt.edu

Satishbabu Nalajala
Computer Science
University of North Texas
Texas, US
Satishbabunalajala@my.Unt.edu

Ishitha konda
Computer Engineering
University of North Texas
Texas, US
Ishithakonda@my.unt.edu

Dunnala Naveen
Computer Science
University of North Texas
Texas, US
Naveendunnala@my.unt.edu

***Abstract*—This study, titled ”Scheduling Algorithms,” details the CPU scheduling algorithm concepts and how they were implemented in a C++ context.**

***Index Terms*—FCFS, SJF, RR, Priority-based RR, Multi-level queue, Multi-level priority queue, Priority Scheduling**

I. Introduction

The foundation of multi-programmed operating systems is scheduling. The operating system can increase computer productivity by moving the CPU between processes. In computer science, scheduling is the technique by which threads, processors, or data flows are given access to system resources. CPU time and communication bandwidth are two examples. In order to successfully load balance a system or reach a certain level of service quality, this is typically done.

A. Overview of the project

By creating a CPU simulator, we are implementing the following CPU scheduling techniques in this project.

- FCFS: One of the simplest and most straightforward algorithmic strategies is FCFS (First Come First Served), often known as FIFO (First In First Out). The processes are merely placed up for processing by the FCFS algorithm in the same order that they enter the ready queue. The procedure enters the system at the end of the ready queue. A process that is currently running is dequeued when it finishes, and the process at the top of the ready queue is then set to execute. Due to the non-preemptive nature of FCFS, no process may be compelled to give up resources in the event that a process with a higher priority enters the queue. This algorithm is easy to implement.

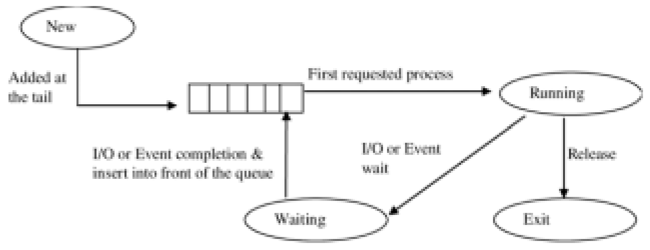


Fig. 1. Process diagram describing FCFS

  With this approach, all processes that are ready are tracked by a single linked list. It is just necessary to choose one process from the front of the queue to run. Simply connecting a new job or unblocked process to the end of the queue will add it.
- SJF: The sole thing that distinguishes SJF (Shortest Job First), also known as SPN (Shortest Process Next), from the SRT (Shortest Remaining Time) method is preemption. The non-preemptive SJF algorithm selects the process with the fastest predicted processing time as its next candidate. Therefore, a quick procedure will beat out longer jobs and move to the front of the line. This calls for in-depth information or calculations of the length of time needed for a process to finish. On the other hand, the SRT algorithm uses the same methods as SJF to introduce the idea of preemption. In this situation, the scheduler always picks the process with the least amount of processing time left to do. It's possible that a new process that joins the ready queue will finish sooner than one that is already running. As a result, when a new process is prepared, the scheduler may preempt the running process. For the selection function to be performed by the scheduler, a processing time estimate is required. In contrast to FCFS algorithms, the SJF algorithm does not prefer longer processes more than shorter ones. In most circumstances, it also offers the highest throughput.
- Round Robin: One of the most crucial and significant algorithms is round robin. Nearly all contemporary algorithmic techniques operate on the same principles as round-robin. It employs the idea of cyclic execution and schedules each task to run at the same time, known as a ”Quantum” or ”Time-slice.” Basically, it builds a queue of processes with all the pending processes and picks, and the first task assigns it a time window to run within. The procedure starts over with the following job in the queue once RR selects it and moves it to the end of the line.

  a) The Quantum Issue: The length of the quantum is the only intriguing problem with round robin. It takes some time to switch from one process to another because of the administrative tasks involved, such as flushing and reloading the memory cache, updating various tables and lists, saving and loading registers and memory mappings, etc. Assume that this process switch, or context switch as it is commonly known, takes 1 msec to complete

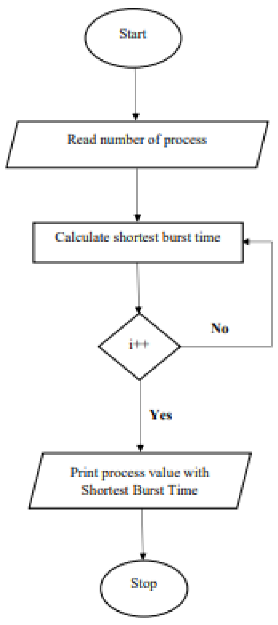

Start
Read number of process
Calculate shortest burst time
i++
No
Yes
Print process value with Shortest Burst Time
Stop

Fig. 2. Process diagram describing SJF

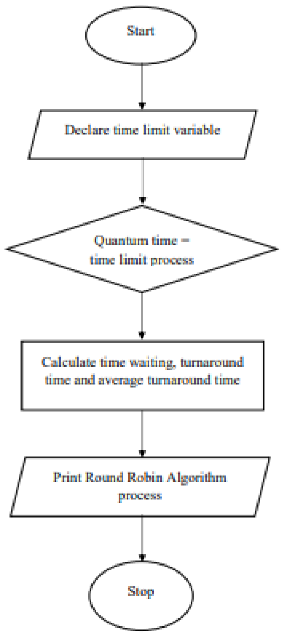

Start
Declare time limit variable
Quantum time = time limit process
Calculate time waiting, turnaround time and average turnaround time
Print Round Robin Algorithm process
Stop

Fig. 3. Process diagram describing RR

and involves changing memory mappings, flushing, and reloading the cache, among other things. Consider the quantum to be set at 4msec as well. With these settings, the CPU will be used for administrative overhead after 4 msec of productive work. This is obviously too much. We could change the quantum setting to, say, 100 msec to increase CPU efficiency. Only 1 percent of the time is currently wasted. But examine what would occur if ten interactive users hit the carriage return key at nearly the same time on a time-sharing system. The list of executable processes will contain ten processes. The first one will begin instantly if the CPU is idle, the second one might not begin for 100 ms, and so on. The majority of consumers will consider a one-second response to a brief instruction to be sluggish.

- Priority Scheduling: Here in the priority scheduling algorithm, every process entered is given a priority and the scheduler arranges all the processes based on their priority level. Always priority is given to the higher priority process. When a lower priority process is getting executed and at the time a higher priority process arrives then the lower one will be interrupted to process the higher priority one. This way the deadline can be achieved easily. And also the waiting time and response time of the higher priority process will be less. This will help in achieving deadlines faster and on time as they are assigned based on priority. There are two types of priority scheduling: Preemptive and non -preemptive. In preemptive scheduling whenever the higher priority process arrives the ongoing process will get interrupted.
- Priority-based Round Robin: This algorithm concentrates on overcoming the drawbacks of round-robin algorithm which gives equal priority to all the processes. If the processor has a smaller CPU burst then round-robin scheduling is not efficient as it increases waiting time and decreases the throughput of the system. Now, this priority-based RR will eliminate the above-discussed defects. This will be achieved in two steps: According to the given priority allocate CPU to the processes for given time quantum for only one time. In the ready queue, processors are organized in ascending order of CPU burst time left. new objectives based on the processes' remaining CPU bursts; the process with the shortest remaining CPU burst is the greatest priority assigned. Depending on the remaining CPU bursts and the new priority established for the processes, each until they have completed their execution, the process has CPU control.
- Multi-level queue: Multi level scheduling algorithm in

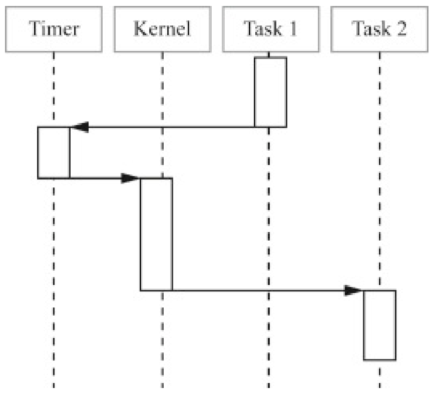


Fig. 4. Preemptive priority scheduling

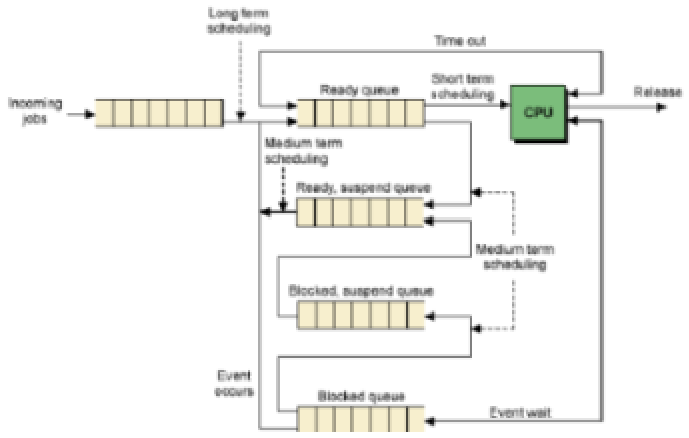


Fig. 5. Priority based RR scheduling

which all the processes in the ready queue are divided into seperate queues based on their scheduling needs. Scheduling needs include memory size, priority, process type. All these seperate queues are processed using different scheduling algorithms. The multilevel queue's key benefit is that it has a minimal scheduling overhead.

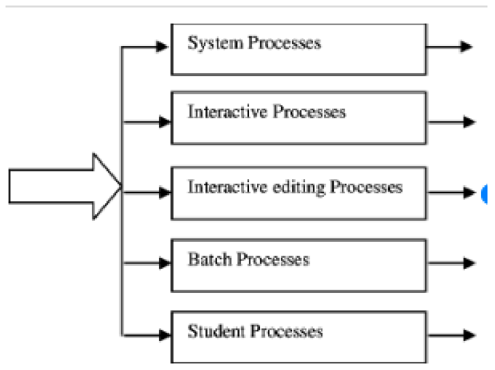


Fig. 6. Multi-level queue scheduling

- Multi-level priority queue: As discussed in the previous section Multi-level priority queue scheduling algorithm in which all the processes in the ready queue are divided into separate queues based on their scheduling needs. Here the change will be in arranging the divided queues. These will be arranged based on their priority and the execution follows the same order. That is after the execution of high-priority queues only the lower ones will execute. As explained earlier every queue will be given different scheduling algorithms depending on its execution flow. All the queues divided from the ready queue will be arranged based on priority and the same flow execution will be done. Depending on the priority of a process when it begins running, the operating system may place it in one of the three queues mentioned above. For instance, if it is a background process, the operating system would prefer that it not be delivered to queues with a higher priority, like queues 1 and 2. It will be immediately added to queue 3, which has a lower priority. Let's assume that our present procedure is given significant priority and is placed in queue 1. The process in queue 1 runs for 4 units before finishing in those units or using those units to provide CPU for I/O operations. If the process completes in those units, its priority remains unchanged, and if it enters the ready queue again, it begins running in queue 1 once more. A process in queue 1 loses priority and is moved to queue 2 if it doesn't finish in 4 units. Here is the diagram explaining the flow of the multi-level priority queue.

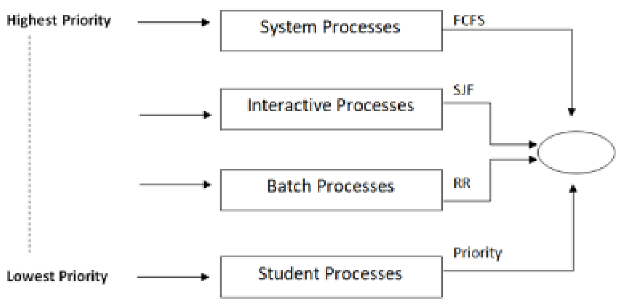


Fig. 7. Multi-level priority queue scheduling

B. What problem being solved

The process of choosing which program will use the CPU while another is suspended is known as CPU scheduling. The primary goal of CPU scheduling is to make sure that the OS has at least chosen one of the processes in the readily available line whenever the CPU is inactive.

When numerous I/O binding processes are chosen by the long-term scheduler in multiprogramming, the CPU is typically idle. Resource usage must be improved for a program to be effective. If the OS has more waiting time there is a chance that the system will fail. So to minimize the waiting time CPU need to be occupied with tasks and thus it also can avoid deadlock scenarios.

C. Why is it important

Modern computer systems rely on multiplexing and multitasking, which is the fundamental justification for the adoption of scheduling algorithms. Multiplexing is the simultaneous transmission of several flows, while multitasking is the process of running many processes at once. The achieve the below goals scheduling algorithms play the main role:

- Efficiency: Keep the CPU active at all times with productive tasks.
- Throughput: Increase the number of jobs processed each hour.
- Waiting time: Reduce the total number of times spent waiting in the ready queue.
- Turn around time: Reduce the amount of time that batch users must wait for output from the time of submission to the time of completion.

- Response time: After submitting a process in the interactive system the response time will be reduced to receive a response.

II. Background

A. What does one need to know to understand the problem

Before understanding schedule algorithms one should know Operating systems concepts like what is a process and the steps in process execution. Process handling involves below states:

- New: The process is set to be in a new state when a secondary memory-allocated process starts to execute.
- Ready: A process is changed to be in a ready state from a new state when it is loaded into the main memory and prepared for execution. The process in a ready state will be waiting for execution by the processor. In a multiprogramming environment, many processes could be in the ready state.
- Waiting: A process switches from the run state to the block state or the wait state if it has to perform an input/output operation or access a blocked resource while running.
- Running: A process moves from the ready state to the run state whenever the CPU is assigned for its execution.
- Terminated: A process transitions from the run state to the terminate state after its execution is complete. After it enters the terminate state, the operating system deletes the process control box (or PCB).

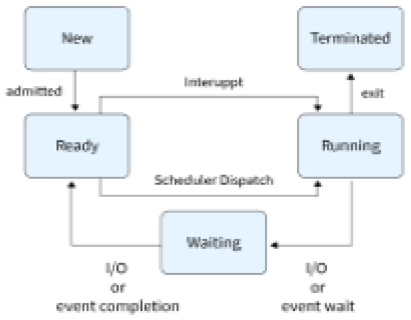


Fig. 8. Stages of Process

By

III. Implementation

A. Solutions of the problem

The problem of deadlock or system failure can be achieved by implementing the below algorithms such that the processor will not be idle for more time at the time of multiprocessing:

- FCFS, which assigns tasks according to the order in which they request the CPU, schedules tasks.
- SJF (shortest-job-first), which orders jobs based on the duration of the subsequent CPU burst.
- Using Priority Scheduling, jobs are scheduled according to importance.
- scheduling using a time quantum for each work, or round-robin (RR) (or for the remainder of its CPU burst).
- Priority with round-robin plans jobs according to the importance and applies round-robin scheduling to activities of similar importance.
- Multi-level queue
- Multi-level priority queue

B. How the implementation is done

- Programming language for implementation: For implementing all the scheduling algorithms we have used C++
- Operating system to test the project: We have used Linux based Operating system for running the implementation code of all algorithms.

C. Test Cases

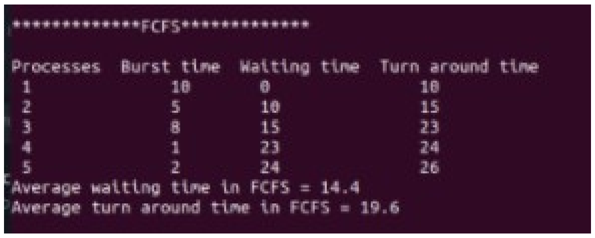
Here we have entered some of the test cases manually by the user and we tested in different situations how the OS will react. we also determined the average waiting time and average turnaround time for every algorithm.

IV. Experimental Results

A. Results or Outputs of implementation

a) FCFS: The below steps show the algorithm for the FCFS algorithm.

- Take the process arrival times
- Sort all the processes in the order of arrival time in ascending order.
- Start the queue and execute the processes in the sorted order.



```
*************FCFS*************

Processes  Burst time  Waiting time  Turn around time
 1          10          0             10
 2          5           10            15
 3          8           15            23
 4          1           23            24
 5          2           24            26
Average waiting time in FCFS = 14.4
Average turn around time in FCFS = 19.6
```

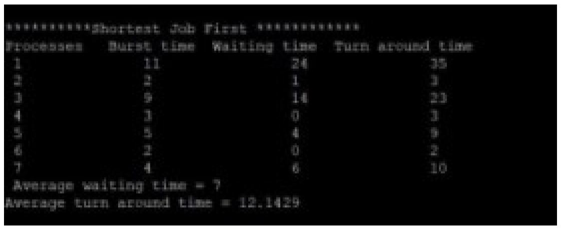
Fig. 9. Output for FCFS scheduling

Above shows the output for the FCFS scheduling algorithm i.e we have provided the CPU with 5 processes with the burst time and waiting time. It completed the execution by following the above algorithm steps and provided the output with turnaround time, averages of waiting and turnaround times.

b) SJF: The below points indicate the algorithm for SJF scheduling.

- Take the process arrival times
- Sort all the processes in the order of arrival time in ascending order.
- Start the queue and execute the processes in the sorted order.
- If the process is Non-preemptive then all the processes execute all the steps. If the process is preemptive it will check for any process arrivals in between execution. If any process arrives in such a way the scheduler will stop the ongoing process by saving its work and add the newly

arrived process to the ready queue by considering burst time and arrival time.
- Again scheduler will start the processing of the queue. The output for SJF algorithm for the given input burst



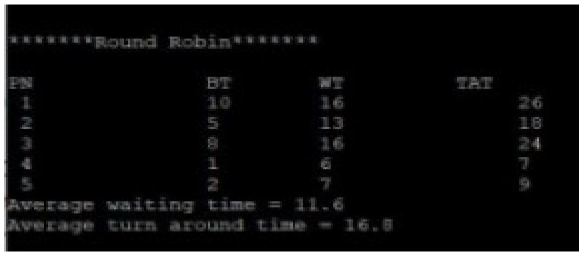
```
**********Shortest Job First ************
Processes   Burst time  Waiting time  Turn around time
 1              11            24             35
 2              2             1              3
 3              9             14             23
 4              3             0              3
 5              5             4              9
 6              2             0              2
 7              4             6              10
 Average waiting time = 7
Average turn around time = 12.1429
```

Fig. 10. Output for SJF scheduling

times and waiting time is shown above. we have provided the CPU with 5 processes with the burst time and waiting time. It completed the execution by following the above algorithm steps and provided the output with turnaround time, averages of waiting and turnaround times.

c) Round-robin: The below steps indicate the algorithm for round-robin scheduling:

- Take the process arrival times
- Quantum time is given for execution
- Start loop with quantum time value Insert the arriving process into the ready queue by assigning quantum time, such that the particular process will execute till the quantum time and gives chance to the next process. Repeat the process till the queue is empty.



```
*******Round Robin*******

PN              BT        WT           TAT
 1              10        16                  26
 2              5         13                  18
 3              8         16                  24
 4              1         6                   7
 5              2         7                   9
Average waiting time = 11.6
Average turn around time = 16.8
```

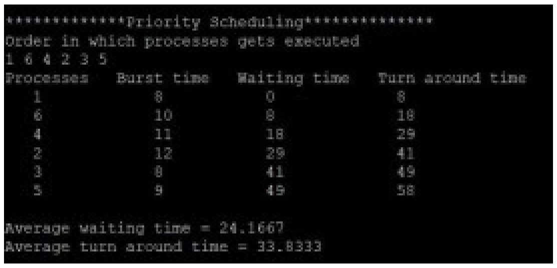
Fig. 11. Output for Round Robin scheduling

The output for the Round-robin algorithm for the given input burst times and quantum time is shown above. we have provided the CPU with 5 processes with burst time and quantum times. It completed the execution by following the above algorithm steps and provided the output with turnaround time, averages of waiting and turnaround times.

d) Priority Scheduling: The below steps indicate the algorithm for round-robin scheduling:

- Read all the burst times and priority of processes
- Arrange all the incoming processes in order of priority.
- Now implement the FCFS on the ready queue that is arranged in priority order

The output for the Priority scheduling algorithm for the given input burst times and priority is shown above. we have provided the CPU with 5 processes with the burst time and



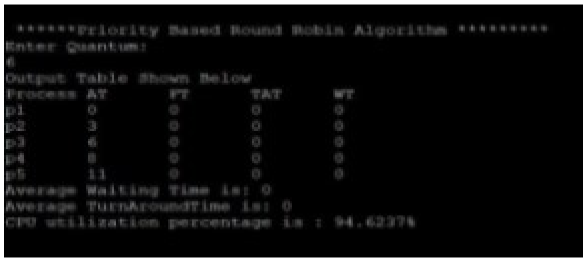
```
*************Priority Scheduling**************
Order in which processes gets executed
1 6 4 2 3 5
Processes   Burst time   Waiting time   Turn around time
   1           8            0              8
   6           10           8              18
   4           11           18             29
   2           12           29             41
   3           8            41             49
   5           9            49             58

Average waiting time = 24.1667
Average turn around time = 33.8333
```

Fig. 12. Output for Priority scheduling

priorities. It completed the execution by following the above algorithm steps and provided the output with turnaround time, averages of waiting and turnaround times.

e) Priority-based round-robin: The below steps indicate the algorithm for Priority-based round-robin scheduling:

- Allocate CPU for all the processes as like for round robin scheduling according to given priority and quantum times.
- After that In the ready queue, processors are organized in ascending order of CPU burst time left. The processes' remaining CPU bursts are used to determine new priorities; the process with the fewest bursts is given the greatest priority.
- Based on the remaining CPU bursts, the processes are performed in accordance with updated priorities, and each process is given CPU control until it has completed its execution.



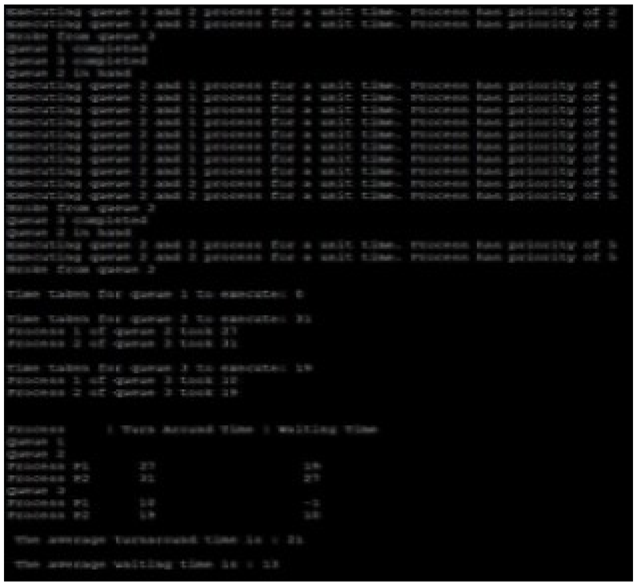
```
 ******Priority Based Round Robin Algorithm *********
Enter Quantum:
6
Output Table Shown Below
Process AT        FT        TAT       WT
p1      0         0         0         0
p2      3         0         0         0
p3      6         0         0         0
p4      8         0         0         0
p5      11        0         0         0
Average Waiting Time is: 0
Average TurnAroundTime is: 0
CPU utilization percentage is : 94.6237%
```

Fig. 13. Output for Priority-based round-robin scheduling

The output for the Priority-based round-robin scheduling algorithm for the given input burst times and priorities and quantum is shown above. we have provided the CPU with 5 processes with the burst time and priorities. It completed the execution by following the above algorithm steps and provided the output with turnaround time, averages of waiting and turnaround times.

f) Multi-level queue: The below steps indicate the algorithm for Multi-level queue scheduling:

- Input the processes.
- Divide the ready queue into separate queues based on their burst times and specifications.
- The processes in the first queue will execute with any of the above mentioned algorithms and second with others and so on.



```
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Executing queue 3 and 2 process for a unit time. Process has priority of 2
Divide from queue 3
Queue 1 completed
Queue 3 completed
Queue 2 in hand
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 1 process for a unit time. Process has priority of 4
Executing queue 2 and 2 process for a unit time. Process has priority of 5
Executing queue 2 and 2 process for a unit time. Process has priority of 5
Divide from queue 2
Queue 3 completed
Queue 2 in hand
Executing queue 2 and 2 process for a unit time. Process has priority of 5
Executing queue 2 and 2 process for a unit time. Process has priority of 5
Divide from queue 2

Time taken for queue 1 to execute: 0

Time taken for queue 2 to execute: 31
Process 1 of queue 2 took 27
Process 2 of queue 2 took 31

Time taken for queue 3 to execute: 19
Process 1 of queue 3 took 10
Process 2 of queue 3 took 19


Process      | Turn Around Time | Waiting Time
Queue 1
Queue 2
Process P1      27                 19
Process P2      31                 27
Queue 3
Process P1      10                 -1
Process P2      19                 10

 The average turnaround time is : 21

 The average waiting time is : 13
```

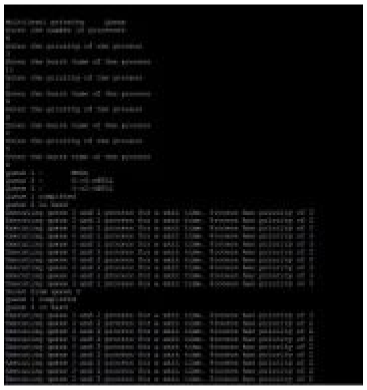
Fig. 14. Output for Multi-level queue scheduling

- This will repeat until all the queues are empty.

The output for the Multi-level queue scheduling algorithm for the given input burst times and waiting time is shown above. we have provided the CPU with processes with the burst time. It completed the execution by following the above algorithm steps and provided the output with turnaround time, averages of waiting and turnaround times.

g) Multi-level priority queue: In Multi-level priority queue, arrangement of queues according to priorities.The queue having the highest priority will be executed first and the lowest priority queue last. The algorithm is as follows:

- Input the processes.
- Divide the ready queue into separate queues based on their burst times and specifications and
- Arrange all the queues based on priority.
- The processes in the first queue will execute with any of the above-mentioned algorithms and the second with others and so on.
- This will repeat until all the queues are empty.



```
[illegible]
```

Fig. 15. Output for Multi-level priority queue scheduling

B. Interpretation of the results

We have run the program in linux server using putty.

V. Conclusion

In the paper we discussed about different algorithms like FCFS, SJF, Priority based RR, Multilevel, Multilevel priority queue, Priority Scheduling. By using these type of algorithms we can reduce time complexity, increase in data consumption, and also in efficiency. Moreover there is still a lot of development in algorithms.

References

[1] M. Jayashree S. Somani M. Pooja and K. Chhatwani ”Comparative Study of Different CPU Scheduling Algorithms” Int. J. Comput. Sci. Mob. Comput. vol. 2 no. 11 pp. 310-318 2013.

[2] M. R. Khan and M. Gaurav Kakhani ”Analysis of Priority Scheduling Algorithm on the Basis of FCFS and amp; SJF for Similar Priority Jobs” Int. J. Comput. Sci. Mob. Comput. vol. 4 no. 9 pp. 324-331 2015.

[3] S. K. Nager and N. S. Gill ”An Improved Shortest Job First Scheduling Algorithm to Decrease Starvation in Cloud Computing” Int. J. Comput. Sci. Mob. Comput. vol. 5 no. 8 pp. 155-161 2016.

[4] El-Sharawy, Enas. (2021). A Review on the CPU Scheduling Algorithms: Comparative Study. International Journal of Network Security. 21. 19-26. 10.22937/IJCSNS.2021.21.1.4.

[5] Adekunle, Yinka and Ogunwobi, Z and Sarumi, Jerry and Efuwape, B and Ainam, Jean-Paul and Ebiesuwa, Oluwaseun. (2014). A Comparative Study of Scheduling Algorithms for Multiprogramming in Real-Time Systems. 12. 180-185.