A Review on Various Scheduling Algorithms

Saraswathi Seemakuthi1, Venkat Alekhya.Siriki2 , Dr. E.Laxmi Lydia3

1B.TECH: III, Department of Computer Science and Engineering, Vignan's Institute Of Information Technology, Visakhapatnam, saraswathi.seemakurthi@gmail.com, Andhra Pradesh, India.

2B.TECH: III, Department of Computer Science and Engineering, Vignan's Institute Of Information Technology, Visakhapatnam, alekhya.sv96@gmail.com, Andhra Pradesh, India.

3Associate Professor, Department of Computer Science and Engineering, Vignan's Institute Of Information Technology, Visakhapatnam, elaxmi2002@yahoo.com, Andhra Pradesh, India.

Abstract- This research paper describes about scheduling, scheduler, classification of scheduling, main purpose of scheduling and various scheduling algorithms such as first come first serve scheduling algorithm, shortest-job-first scheduling algorithm, priority scheduling algorithm, round robin scheduling algorithm, multilevel-queue scheduling algorithm, multilevel feedback queue scheduling algorithms. This research paper describes how these algorithms are implemented, with the parameters such as average waiting time and average turnaround time, Gantt chart and how average waiting time and average turnaround time are calculated, merits and demerits of the scheduling algorithms.

Keywords: scheduling, scheduler, turnaround time, Gantt chart

1. INTRODUCTION

Why scheduling algorithms are used? When a process is waiting for an I/O request or some other requests that is needed for its execution, the CPU will be sitting idle. To maximize the CPU utilization time the CPU is allocated to another process which is waiting in the ready queue. This can be achieved by scheduling. [1] Scheduling is the method by which work specified by some means is assigned to resources that complete the work. Scheduling can be categorized into two types they are preemptive scheduling and non preemptive scheduling. [2] The preemptive ling scheduling is prioritized. The highest priority process should always be the process that is currently utilized. In non-preemptive scheduling, a running task is executed till completion. It cannot be interrupted. [1]A scheduler is what carries out the scheduling activity. Schedulers are often implemented so they keep all compute resources busy allow multiple users to share system resources effectively, or to achieve a target quality of service. The main purposes of scheduling algorithms are to minimize resource starvation and to ensure fairness amongst the parties utilizing the resources. Scheduling deals with the problem of deciding which of the outstanding requests is to be allocated resources. There are many different scheduling algorithms.

1.1 FIRST – COME, FIRST -SERVED SCHEDULING ALGORITHM

FCFS is a scheduling algorithm which is based on non preemptive scheduling. This is the simplest algorithm as the name indicates the process that request for the CPU first is allocated the CPU first. It is implemented by FIFO queue. [3] When the process enters the ready queue, it's PCB (Process Control Block) is linked onto the tail of the queue. When CPU is free, it is allocated to the process at the head of the queue. The running process is removed from the queue. Figure1 demonstrates the FCFS scheduling algorithm.

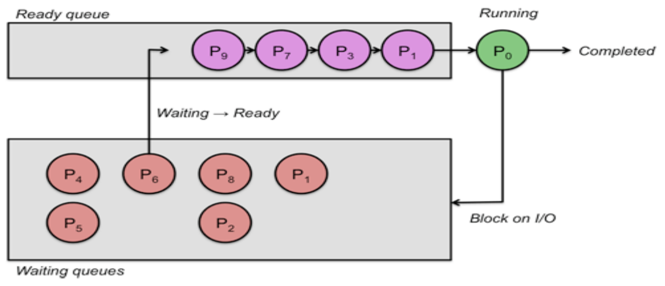


Figure 1 FCFS scheduling algorithm.

A.IMPLEMENTATION OF FCFS

Table1 shows the some processes that arrive at time 0 milliseconds.

Process	Burst Time
P_1	20
P_2	30
P_3	6
P_4	2

Table 1

If these processes arrive in order of P_1, P_2, P_3, and P_4 and served in FCFS order, Figure 2 shows the results of Gantt chart:

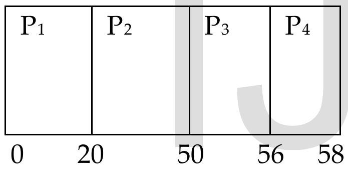


Figure 2 Gantt chart

The waiting time is 0 milliseconds for 'P_1' 20 milliseconds for 'P_2', 50 milliseconds for 'P_3' and 56 milliseconds for 'P_4'

Average waiting time:

(20+50+56) /3=84 milliseconds.

If the process arrives in order P_4, P_3, P_1,P_2 the results are shown in the figure3.

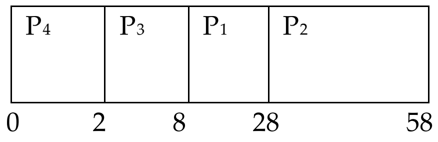


Figure 3 Gantt chart

Average waiting time:

38 / 3=12.666 milliseconds.

The average waiting time under FCFS policy is generally not minimal and may vary substantially if the processes CPU burst time varies greatly .There is a common effect as all other processors wait for the one big process to get off the CPU. This effect results in lower CPU and device utilization than might be possible if the shortest process were allowed to go first. The FCFS scheduling is non-preemptive i.e. once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating the process or by requesting I/O.

B.ADVANTAGES OF FCFS:

- [1] The lack of prioritization means that as long as every process eventually completes, there is no starvation. In an environment where some processes might not complete, there can be starvation.
- It is based on Queuing
- FIFO scheduling is simple to implement. It is also intuitively fair.
- [15]Provably optimal with respect to minimizing the average waiting time.
- [15]I/O bound jobs get priority over CPU bound jobs.

C.DISADVANTAGES OF FCFS:

- [1]Since context switches only occur upon process termination, and no reorganization of the process queue is required, scheduling overhead is minimal.
- Throughput can be low, since long processes can hold the CPU
- Turnaround time, waiting time and response time can be high for the same reasons above
- No prioritization occurs, thus this system has trouble meeting process deadlines.
- [3]It is particularly troublesome for time sharing systems, where it is important that each user get a share of regular intervals.
- It would be disastrous to allow one process to keep the CPU for an extended period.

1.2 SHORTEST-JOB-FIRST SHEDULING

[4]Shortest-Job-First (SJF) is a non-preemptive discipline in which waiting job (or process) with the smallest estimated run-time-to-completion is run next. In other words, when CPU is available, it is assigned to the

process that has smallest next CPU burst. This algorithm associates with each process the length of processes next CPU burst .When the CPU is available, it is assigned to the process that has the smallest next CPU burst. When the next CPU bursts are same, then the tie is beaked using FCFS algorithm. The more appropriate term for this scheduling method would be Shortest-Next-CPU-Burst algorithm, because scheduling depends on the length of the next CPU burst of a process, rather than its total length. Figure 4 demonstrating the shortest job first algorithm.

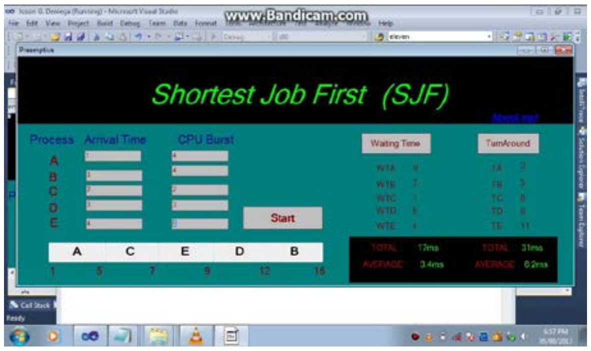


Figure 4 shortest job first

As an example of SJF scheduling, consider the following set of process, with the length of the CPU burst given in milliseconds. Table 2 demonstrates the implementation in SJF scheduling.

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	3

Table 2

Using SJF scheduling, figure 5 shows the results of Gantt chart

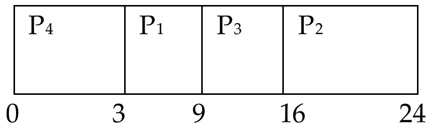


Figure 5 Gantt chart

The waiting time is 3 milliseconds for 'P1', 9 milliseconds for 'P3',16 milliseconds for ' P2' and 0 milliseconds for 'P4'.Average waiting time is 9.333 milliseconds. If the CPU is given to the processes in FCFS order, Figure 6 shows the results of Gantt chart:

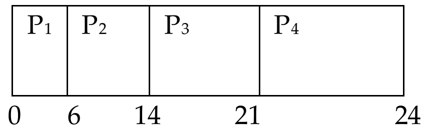


Figure 6 Gantt chart

The waiting time for 'P1' is 0 milliseconds, 'P2' is 6 milliseconds, and 'P3' is 14 milliseconds and 'P4'is 21 milliseconds .Average waiting time is 13.666 milliseconds. From this we can say that SJF decreases the waiting time hence it is provably optimal. Although the SJF algorithm is optimal, it cannot be implemented at the level of short term CPU scheduling .There is no way to know the length of the next CPU burst. One approach is to try to approximate SJF scheduling .In this approach we can predict the next CPU burst. We expect the next CPU burst will be similar in length to previous one. The next CPU burst is generally predicted as an exponential average of the measured lengths of previous ones. Let t_n be the lengths of nth CPU burst, T_{n+1} be the predicted value for the next CPU burst. Then, for α, $0 \leq \alpha \leq 1$ define

$$T_{n+1}=\alpha\ tn + (1-\alpha)\ Tn.$$

This formula defines on exponential average Tn stores the past history. The parameter α controls the relative weight of recent and past history in our prediction. If α =0,then $T_{n+1}=T_n$ recent history has no effect .If α=1,then $T_{n+1}=T_n$,and only the most recent CPU burst matters. More commonly α=1/2 so recent history and past history are equally weighted.

CPU burst (ti) 6 4 6 4 13 13 13 ………

Guess (ti) 10 8 6 6 5 9 11 12 ………….

Figure 7 shows the results of Gantt chart

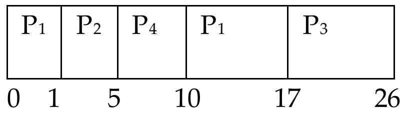
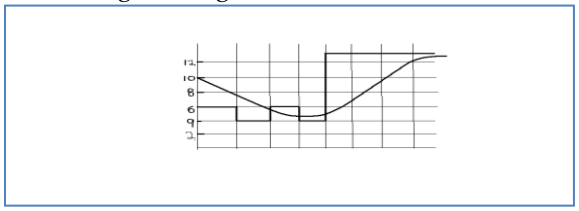


Figure 7 Gantt chart

Graph 1 demonstrates the implementation of SJF scheduling algorithm



Graph 1

The SJF algorithm can be either preemptive or non-preemptive. The choice arises when a new process arrives at ready queue while a previous process is still executing .The next CPU burst of the newly arrived process may shorter than what is left of currently executing process ,where as a non preemptive SJF algorithm will allow the currently running process to finish its CPU burst. Preemptive SJF scheduling is sometimes called shortest – remaining – time –first scheduling.

As an example consider the following four processes, Table 3 demonstrates the implementation in SJF scheduling

Process	Arrival time	Burst time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Table 3

Process P_1 is started at time 0, since it is the only process in the queue. Process P_2 arrives at time .The remaining time for process P_1 (7 milliseconds) is larger than the time required by process P_2 (4 milliseconds) so process P_1 is preempted and process P_2 is scheduled.

The average waiting time is:

((10-1) (1-1)(17-2)(5-3)) /4=26/4=6.5 milliseconds.

A.ADVANTAGES OF SJF:

This scheduling is optimal in that it always produces the lowest mean response time. Processes with short CPU bursts are given priority and hence run quickly (are scheduled frequently).

- [4]The SJF scheduling is especially appropriate for batch jobs for which the run times are known in advance.
- Since the SJF scheduling algorithm gives the minimum average time for a given set of processes, it is probably optimal.
- The SJF algorithm favors short jobs (or processors) at the expense of longer ones.
- The best SJF algorithm can do is to rely on user estimates of run times.
- This algorithm is designed for maximum throughput in most scenarios

B.DISADVANTAGES OF SJF:

- [1] If a shorter process arrives during another process' execution, the currently running process may be interrupted (known as preemption), dividing that process into two separate computing blocks. This creates excess overhead through additional context switching.
- The scheduler must also place each incoming process into a specific place in the queue, creating additional overhead.
- It is not useful in timesharing environment in which reasonable response time must be guaranteed.
- If CPU is allocated in SJF non preemptive then there will be starvation of length of larger CPU burst process.

1.3 PRIORITY SHEDULING ALGORITHM

[3] In this algorithm a priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal priority processes are scheduled in FCFS order. Priorities are indicated by some fixed range of numbers, such as 0-7 or 0-4095.However, there is no general agreement on whether 0 is the highest or lowest priority. Some systems use lowest priority numbers to represent highest priority and some use highest numbers to represent highest priority. Priorities are defined either internally or externally. Internally defined priorities use some measurable quantity or quantities to compute the priority of the process. For

example, time-limits, memory requirements, the number of open files. External priorities are set by criteria outside the operating system, such as the importance of process, the type and amount of funds being paid for computer use, the department sponsoring the work and other, often political, factor. Priority scheduling can be either preemptive or non preemptive .A preemptive priority scheduling algorithm will preempt the CPU if the priority of currently running process is lower than priority of newly arrived process. A non preemptive priority scheduling algorithm will simply put the new process at the head of ready queue. Figure 8 demonstrates the implementation in priority scheduling

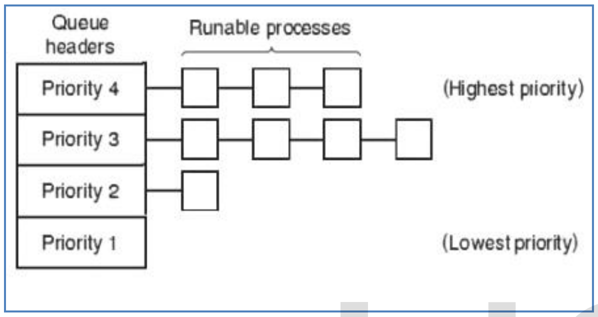


Figure 8 priority scheduling algorithm

As an example consider the following four processes. Table 4 demonstrates the implementation in priority scheduling

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

Table 4

Figure 9 shows the results of Gantt chart

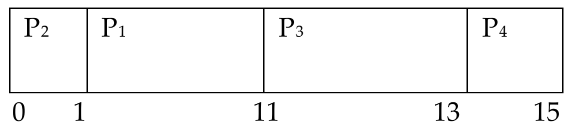


Figure 9 Gantt chart

The average waiting time is 6.25 milliseconds

A.ADVANTAGES:

- [6] Priority scheduling provides a good mechanism where the relative importance of each process may be precisely defined.
- [5] Simplicity: suitable for applications with varying time and resource requirements.
- Reasonable support for priority.

B.DISADVANTAGE:

- A major problem with priority scheduling algorithm is indefinite blocking, starvation
- A process that is ready to run but waiting for the CPU can be considered blocked .A priority scheduling algorithm can leave some low priority processes waiting indefinitely. Solution to this problem is Aging .Aging is a technique of gradually increasing the priority of processes that wait in the system for long time.
- If the system eventually crashes then all unfinished low priority processes gets lost.

1.4 ROUND ROBIN SCHEDULING ALGORITHM:

[3]It is designed especially for time sharing systems. It is similar to FCFS scheduling, but preemption is added to switch between processes. A small unit of time called a time quantum or time slice is defined. A time slice is generally from 1 to 100 milliseconds. The ready queue is treated as circular queue. The CPU scheduler goes around the ready queue, allocating to each process for a time interval of up to 1 quantum. To implement RR scheduling, we keep the ready queue as FIFO queue of processes. New processes are added to tail of queue. The process may have CPU burst less than time quant; in

this case the process itself will release CPU voluntarily. If CPU burst of currently running process is longer than time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed and the process will be put at the tail of ready queue. The CPU scheduler will then select next process in ready queue. Consider the following example which illustrates Round Robin algorithm. Figure 10 demonstrating round robin scheduling algorithm.

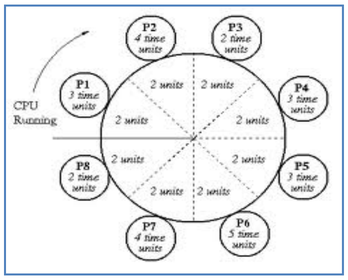


Figure 10 round robin scheduling algorithm

Table 5 demonstrates the implementation in Round robin scheduling

Process	Burst time
P_1	24
P_2	3
P_3	3

Table 5

Figure 11 shows the results of Gantt chart

P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1

0 4 7 10 14 18 22 26 30

Figure 11 Gantt chart

Let us consider time quant of 3 milliseconds. Then average waiting time is 17/3 is 5.66 milliseconds. The performance of RR algorithm depends heavily on the size of time quantum. If it is extremely large then the policy is same as time quantum. If it is extremely small, the RR approach is called process sharing and creates the appearance that each of n process has its own processor.

A. Advantages:

- [6] Round robin scheduling is fair in that every process gets an equal share of the CPU.
- It is easy to implement and, if we know the number of processes on the run queue, we can know the worst-case response time for a process.
- [1] Good average response time, waiting time is dependent on number of processes, and not average process length
- Because of high waiting times, deadlines are rarely met in a pure RR system.
- Starvation can never occur, since no priority is given. Order of time unit allocation is based upon process arrival time, similar to FCFS.
- Round Robin is excellent for parallel computing is because round-robin is great for load balancing if the tasks are around the same lengths.

B.DISADVANTAGES:

- [6]RR scheduling involves extensive overhead, especially with a small time unit.
- Balanced throughput between FCFS and SJF, shorter jobs are completed faster than in FCFS and longer processes are completed faster than in SJF
- [1] Giving every process an equal share of the CPU is not always a good idea. For instance, highly interactive processes will get scheduled no more frequently than CPU-bound processes.
- [8]Low throughput: If round robin is executed in circular way then more context switches occur so throughput will be low
- Context switch leads to the wastage of time, memory and leads to scheduler overhead.
- Round robin made larger response time which is the drawback because system performance will be degraded.

1.5 MULTILEVEL QUEUE SCHEDULING:

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. In addition, foreground processes may have priority (externally defined) over background processes. A multilevel queue scheduling algorithm partitions the ready queue into several queues .The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.

Let us consider an example of a multilevel queue scheduling algorithm with five queues, listed below in order of priority:

- System Processes
- Interactive Processes
- Interactive editing Processes
- Batch Processes
- Student Processes

  Figure 12 demonstrating the multilevel queue scheduling algorithm.

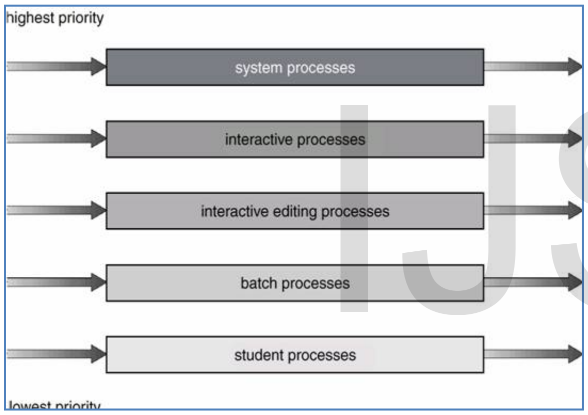


Figure 12 multilevel queue scheduling algorithm

Each queue has absolute priority over low priority queues. No process in the student queue could run unless the queues for system processes, interactive processes, interactive editing processes and batch processes were all empty. If system process entered the ready queue while a batch process was running, the batch process would be preempted. Another possibility is to time slice among the queues. Each queue gets a certain portion of the CPU time.

A.ADVANTAGES:

- [9]Since processes do not move between queues so, this policy has the advantage of low scheduling overhead
- [10]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
- [12]Can be preemptive or non-preemptive
- [14]Flexible implementation with respect to movement between queues.

B.DISADVANTAGES:

- [9]This scheduling algorithm is inflexible
- [10]It is difficult to understand to implement

1.6MULTILEVEL FEEDBACK-QUEUE SCHEDULING:

[3]This algorithm allows the process to move between the queues instead of assigning the process permanently to a certain queue. This algorithm separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it will be moved to a low-priority queue. A process that waits too long in lower –priority queue may be moved to a higher –priority queue. This form of aging prevents starvation.

- A multilevel feedback-queue scheduler is defined by following parameters:
- The number of queues
- The scheduling algorithm for each queue
- The method used to determine when to upgrade a process to a higher priority queue
- The method used to determine when to demote a process to a lower-priority queue
- The method used to determine which queue a process will enter when that process needs service

  The definition of a multilevel feedback-queue scheduler makes it the most general CPU-scheduling algorithm. It can be configured to match a specific system under design.Figure13 demonstrating the multilevel feedback-queue scheduling algorithm.

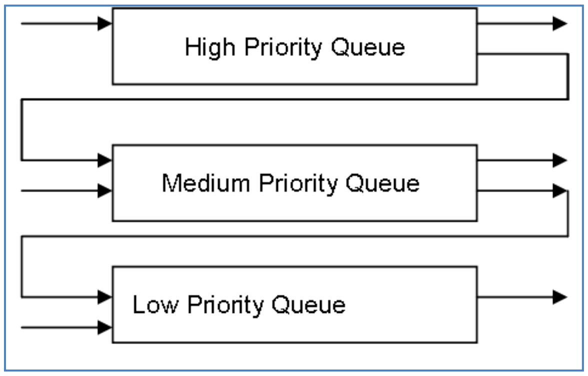


Figure 13 multilevel feedback-queue scheduling algorithm

A.ADVANTAGES:

- [11]This scheme will continue until the process completes or it reaches the base level queue.
- A process that waits too long in a lower priority queue may be moved to a higher priority queue.
- To exploit this behavior, the scheduler can favor jobs that have used the least amount of CPU time, thus approximating SJF.
- This policy is adaptive because it relies on past behavior and changes in behavior result in changes to scheduling decisions.
- [13]A process that waits too long in a lower priority queue may be moved to a higher priority queue.

B.DISADVANTAGES:

- [3]This algorithm is most complex algorithm because defining the best scheduler requires some means by which to select values for all parameters.
- [11]If the process is completed within the time quantum of the given queue, it leaves the system.
- [12]Moving the process around queue produce more CPU overhead.
- If job's time slices expires, drop its priority one level.
- [13]Moving the process around queue produce more CPU overhead.

2. CONCLUSION:

When designing an operating system, a programmer must consider which scheduling algorithm will perform best for the use the system is going to see. There is no universal "best" scheduling algorithm, and many operating systems use extended or combinations of the scheduling algorithms above. First come first serve scheduling algorithm is simple to understand and suitable only for batch system where waiting time is large. The shortest job first scheduling algorithm deals with different approach. In this algorithm, the major benefit is that it gives the minimum average waiting time. The priority scheduling algorithm is based on the priority in which the highest priority job can run first and the lowest priority job need to wait though it will create a problem of starvation. The round robin scheduling algorithm is preemptive which is based on FCFS policy and time quantum. This algorithm is suitable for the time sharing systems. In multilevel queue scheduling, processes are permanently assigned to a queue depending upon its nature and no process in the lower priority queue could run unless the higher priority queues were empty. Also, it is pre-emptive in nature. Multilevel feedback queue scheduling is also pre-emptive in nature and it allows the processes to move between the queues depending upon the given time quantum.

3. REFERENCES:

[1]https://en.wikipedia.org/wiki/Scheduling_(computing)

[2]http://www.careerride.com/OS-preemptive-scheduling.aspx

[3]Operating system principles: Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

[4]http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/sjfSchedule.htm

[5]http://www.answers.com/Q/Advantage_and_disadvantage_of_priority_scheduling

[6]www.cs.rutgers.edu/~pxk/416/notes/07-scheduling.html

[7]www.researchgate.net/post/What_would_be_the_advantages_and_disadvantages_of_using_Dynamic_Time_slicing_concept_for_scheduling

[8]http://shodhganga.inflibnet.ac.in:8080/jspui/bitstream/10603/50607/8/chap4descriptn.pdf

[9]http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/multQueue.htm

[10]http://www.answers.com/Q/What_is_the_advantage_and_disadvantage_of_multilevel_queue_scheduling

[11]https://en.wikipedia.org/wiki/Multilevel_feedback_queue.

[12]https://prezi.com/kzs2ycuj-c65/multilevel-queue-scheduling/

[13]www.sciencehq.com/computing-technology/1353.html

[14]https://prezi.com/kzs2ycuj-c65/multilevel-queue-scheduling/

[15]ijcsmc.com/docs/papers/November2013/V2I11201368.pdf