**CPU SCHEDULING ALGORITHMS**

Report on package submitted by

**Amritha N (18PT02)**

**Swathy Anand (18PT39)**

SUBJECT: Operating Systems

Department of Applied Mathematics and

Computational Sciences

PSG College of Technology,

Coimbatore - 641 004

**ABSTRACT**

The project titled "CPU SCHEDULING ALGORITHMS", is basically a program which simulates the following scheduling algorithms:

* FCFS ( First Come First Served)
* SJF ( Shortest Job First)
* SRTF( Shortest Remaining Time First)
* Round‐Robin

CPU SCHEDULING is a key concept in computer multitasking, multiprocessing operating system and real‐time operating system designs. Scheduling refers to the way processes are assigned to run on the available CPUs, since there are typically many more processes running than there are available CPUs.

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. By switching the CPU among processes, the operating system can make the computer more productive. A multiprogramming operating system allows more than one processes to be loaded into the executable memory at a time and for the loaded processes to share the CPU using time‐multiplexing.

Scheduling algorithm is the method by which threads, processes or data flows are given access to system resources (e.g. processor time, communications bandwidth). The need for a scheduling algorithm arises from requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously).

**TOPICS**

* 1. Introduction
  2. Description
  3. Tools and Technologies
  4. Work flow
  5. Result and discussion
  6. Conclusion
  7. Bibliography
  8. **INTRODUCTION**

A scheduling algorithm is a set of rules that determines the task to be executed at a particular moment. Although there are a number of scheduling algorithms that have been proposed in the literature, the design of those algorithms is challenged by need for supporting different levels of services, fairness, and implementation complexity and so on. A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. It is the 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.

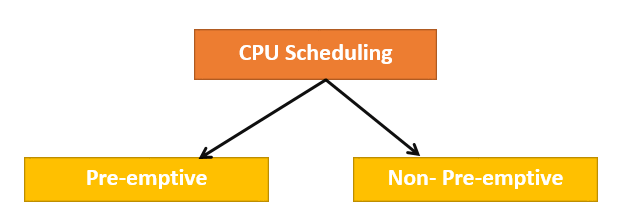


Fig 1.1.1 Types of CPU Scheduling algorithms

These algorithms are either preemptive or non-preemptive as in Fig 1.1.1. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

Almost all programs have some alternating cycle of CPU number crunching and waiting for I/O of some kind. (Even a simple fetch from memory takes a long time relative to CPU speeds.) In a simple system running a single process, the time spent waiting for I/O is wasted, and those CPU cycles are lost forever. A scheduling system thus allows one process to use the CPU while another is waiting for I/O, thereby making full use of otherwise lost CPU cycle.

* 1. **DESCRIPTION**

We are going to be discussing 4 Scheduling Algorithms namely:

* First In First Out (FIFO)
* Shortest Job First (SJF)
* Shortest Remaining Time First (SRTF)
* Round Robin (RR)

to help the user understand how exactly these algorithms work to perform CPU Scheduling.

We used the concept of multi threads in order to achieve simultaneous execution of the Ready Queue, Blocked Queue and the Running State. Semaphores are used to maintain the synchronization between multiple threads. These concepts have been used to show how the processes travel from one state/queue to another, according to that particular Algorithm being used.

* 1. **TOOLS AND TECHNOLOGIES USED**

This is a *C* project, mainly done on the Linux terminal. We have focused largely on the usage of multiple threads to show the exact implementation of the Scheduling Algorithm because it helps to run different functions to run parallelly. Semaphores are used to achieve synchronization of the threads and to avoid segmentation faults.

* 1. **WORK FLOW**

The algorithms are implemented by a menu driven program. In the main function multiple threads are created which are used to execute the different states of the process like ready state, running state, blocked state. The program is capable of getting input from the user dynamically. The user can input CPU burst time and I/O time required for the process. As in *fig 1.4.1*, the inputs are got in an infinite loop in the main function. The main function is made to run parallelly to other threads by having an infinite loop in the threads

.

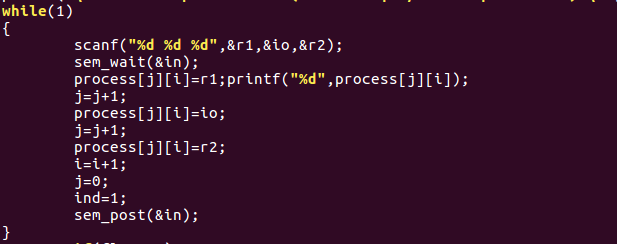


Fig 1.4.1 Infinite loop that dynamically gets input

**First In First Out (FIFO)** algorithm is implemented by creating three threads namely ready\_state, run\_statefifo, blocked\_state. The thread ready\_state helps to push the processes to the ready queue as shown in *fig 1.4.2.* In run\_statefifo, the process are dequeued from the ready queue and a loop is run that decreases the CPU burst time until it becomes zero as in *fig 1.4.3*. The process is pushed to the blocked queue if it has I/O time to wait. Again if the process has some additional execution time it is pushed to the ready queue which is clearly shown in *fig1.4.4.*

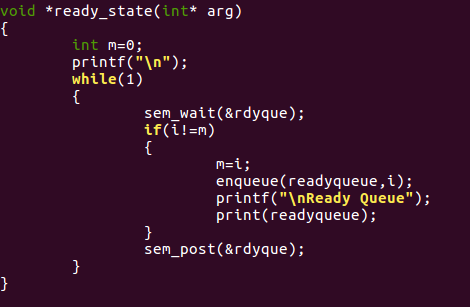


Fig 1.4.2 Infinite loop that dynamically gets input

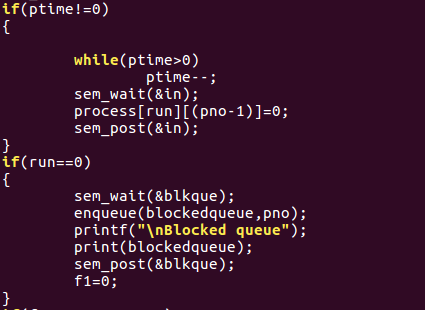


Fig 1.4.3 Decrement of execution time and pushing to blocked queue.

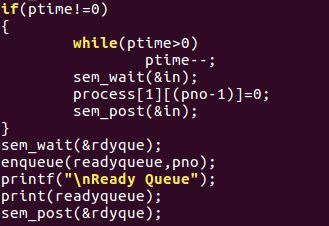


Fig 1.4.4 Decrement of I/O wait time.

**Shortest Job First(SJF) – Non preemptive** algorithm uses same ready\_state thread which is used in FIFO algorithm. In run\_statesjfnp, the shortest process of all the processes in the ready queue is found and the CPU burst time of that process is reduced until it becomes zero which is shown in the *fig 1.4.5*. The process is then pushed to the blocked queue if it has I/O wait time and it gets decreased.

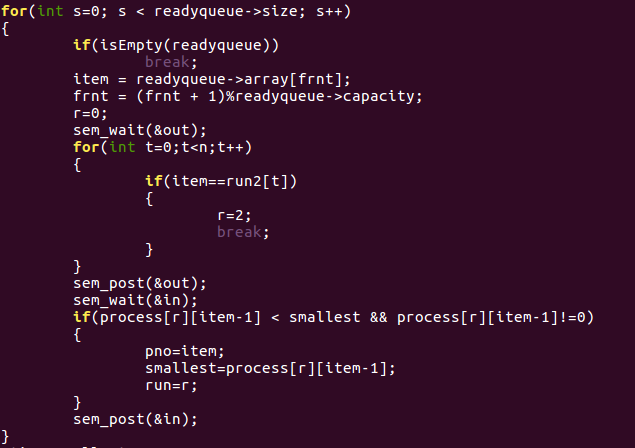


Fig 1.4.5 Finding shortest process of all the processes in queue

**Shortest Job First(SJF) – Preemptive** or the Shortest Remaining Time First algorithm uses the same ready\_state as in FIFO. In the run\_state, the smallest process is found as shown in *fig 1.4.6* , and the execution time for that particular process is decreased by 1(since 1 clock cycle is being executed). If the execution time for that particular process is still not zero, then as per certain conditions, it is enqueued back into the Ready Queue and/or Blocked Queue as shown in *fig 1.4.7*. This process continues for every clock cycle.

****

Fig 1.4.6 Finding the smallest process amongst all processes

**

Fig 1.4.7 Enqueueing the process back to Ready State and/or Blocked State

**Round Robin(RR)** algorithm involves a run state where the CPU burst time is reduced by time quantum as in *fig 1.4.8.* After reducing it by time quantum if the CPU burst time is zero it is pushed to blocked queue for I/O services otherwise it is again pushed to the ready queue. This is performed repeatedly until all processes terminate.



Fig 1.4.8. Reducing the CPU burst time by time quantum.

* 1. **RESULT**

For every algorithm, we input the process (in the order of CPU, I/O, CPU) as and when the user wishes and the status of the ready state and Blocked State will be displayed at every instant. The Output will run indefinitely and the user can input how many ever processes he/she likes.

It is absolutely necessary to note that the 1 CPU clock cycle is not equal to 1 second. Hence only if the user inputs a very large CPU Burst and I/O Burst value, will be see some noticeable change amongst the different algorithms.

* 1. **CONCLUSION**

CPU Scheduling is of utmost importance to make sure that the CPU does not remain idle waste its clock cycles. Through this project, we successfully learnt the implementation of some of the most important Scheduling Algorithms and how it works internally.

* 1. **BIBLIOGRAPHY**

The following book/web references have been used to implement the project:

* “Operating Systems Concepts” by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, 9th Edition
* “Operating Systems-Internals and Design Principles” by William Stallings, 7th Edition
* <https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm>
* <https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/6_CPU_Scheduling.html>