International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 6 Issue 11|| November 2017 || PP. 67-71

Simulation of Process Scheduling Algorithms

Akanksha Verma[1], Yojna Arora[2]
[1](Student, Department of CSE, Amity University, Haryana, India)
[2](Assistant Professor, Department of CSE, Amiyu University, Haryana, India)
Corresponding Author: Akanksha Verma

ABSTRACT: *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. In this paper, simulation of various scheduling algorithm First Come First Serve (FCFS), Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Priority scheduling anf Round Robin is done.*

KEYWORDS: *CPU scheduling, preemptive, dispatcher, scheduling algorithms*

Date of Submission: 26-10-2017 Date of acceptance: 11-11-2017

I. INTRODUCTION

An operating system is a program that manages the hardware and software resources of a computer. It is the first thing that is loaded into memory when we turn on the computer [1]. Without the operating system, each programmer would have to create a way to send data to a printer, tell it how to read a disk file, and how to deal with other programs. In the beginning programmers needed a way to handle complex input/output operations. The evolution of a computer programs and their complexities required new necessities. Because machines began to become more powerful, the time a program needed to run decreased. However, the time needed for handling off the equipment between different programs became evident and this led to program like DOS. As we can see the acronym DOS stands for Disk Operating System. This confirms that operating systems were originally made to handle these complex input/output operations like communicating among a variety of disk drives. Earlier computers were not as powerful as they are today. In the early computer systems you would only be able to run one program at a time. For instance, you could not be writing a paper and browsing the internet all at the same time. However, today's operating systems are very capable of handling not only two but multiple applications at the same time. In fact, if a computer is not able to do this it is considered useless by most computer users. In order for a computer to be able to handle multiple applications simultaneously, there must be an effective way of using the CPU. Several processes may be running at the same time, so there has to be some kind of order to allow each process to get its share of CPU time.

An operating system must allocate computer resources among the potentially competing requirements of multiple processes. In the case of the processor, the resource to be allocated is execution time on the processor and the means of allocation is scheduling. The scheduling function must be designed to satisfy a number of objectives, including fairness, lack of starvation of any particular process, efficient use of processor time, and low overhead. Over the years, scheduling has been the focus of intensive research, and many different algorithms have been implemented.

In a multiprogramming systems, multiple process exist concurrently in main memory [2]. Each process alternates between using a processor and waiting for some event to occur, such as the completion of an I/O operation. The processor are kept busy by executing one process while the others wait, hence the key to multiprogramming is **Scheduling**.

The CPU scheduling is one of the important problems in operating systems designing and build a program achieve these algorithms has been a challenge in this field , and because of there is many scheduling algorithms so we choose some of them for enforcement one of Incentives for us to build this program.

Implementation of the CPU scheduling algorithms existing in operating systems book and researches on real world and calculate average waiting time and turnaround time with drawing a grant chart for algorithms and compare its performance to discover suitable and the best algorithms. The system provides a clean and convenient way to test the given data and do the analysis of CPU scheduling algorithms.

For the purpose of analysis and testing the user first specifies each process along with its information such as arrival times and burst time and then algorithms can be computed producing output in appropriate format readability. Main objective of Multiprogramming is to keep on running processes all the time for maximum CPU utilization.

- It gives maximum throughput in doing the multiple task at the same time.
- It also helps in avoiding indefinite postponement in the processes.
- Give preference to processes holding key resources
- Give better service to processes that have desirable behaviour patterns
- Degrade gracefully under heavy loads
- Maximize number of users receiving acceptable response times.

II. BASIC CONCEPTS

In a single processor system, only one process can run at a time; any others must wait until the CPU is free and can be rescheduled. The objective of multiprogramming is to have some processes running at all time, to maximize CPU utilization. The idea is relatively simple. A processor is executed until it must wait typically for the completion of some I/O request. In a simple computer system, the CPU then just sits idle. All this waiting time is wasted no useful work is accomplished. With multi programming, we try use this time productively[3].

Several processes are kept in memory at one time. When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process. This pattern continues. Every time one process has to wait, another process can take over use of the CPU. Scheduling of this kind is a fundamental operating system function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating system design.

2.1 CPU I/O BURST CYCLE

The success of CPU scheduling depends on an observed property of processes; process execution consist of a cycle of a CPU execution and I/O wait. Processes alternate between these two states. A process begins with a CPU burst, followed by an I/O burst, followed by another CPU burst, then another I/O burst, and so on. Eventually, last CPU burst ends with a system request to terminate the execution.

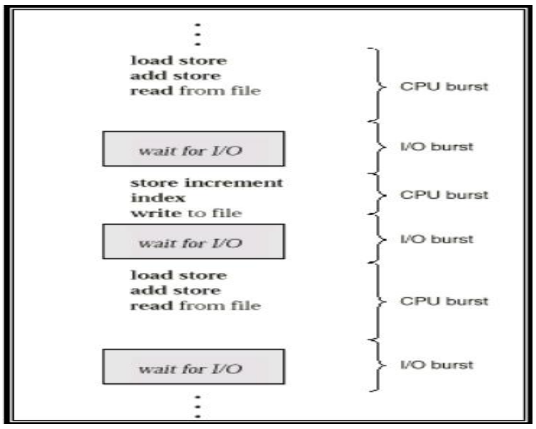


Figure 1 Alternating Sequence of CPU and I/O Burst.

The CPU burst durations have been measured extensively. Although they vary greatly from process to process and from computer to computer, they tend to have a frequency curve as following:

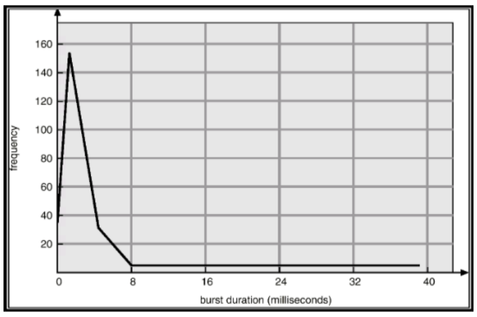


Figure 2 Histogram of CPU- burst durations

The curve is generally characterized as exponential or hyper exponential, with a large number of short CPU bursts and small number of long CPU bursts. An I/O bound program typically has many short CPU bursts. A CPU – bound program might have a few long CPU bursts. This distribution can be important in the selection of an appropriate CPU- scheduling algorithm.

2.2 DISPATCHER

The dispatcher sis the module that gives control of the CPU to the process selected by the short-term scheduler; this involves [4]:

- Switching context
- Switching to user mode
- Jumping to the proper location in the user program to restart that program

2.3 CPU SCHEDULER

Whenever, the CPU becomes idle, the operating system must select one of the processes in the ready-queue to be executed. The selection process is carried out the short-term scheduler or CPU scheduler. The CPU scheduler selects a process from the processes in memory that are ready to execute and allocates the CPU to that process. The ready queue is not necessarily a first-in, first-out (FIFO) queue. It can be implemented as a FIFO queue a priority queue. A tree, or simply an unordered linked list. Conceptually, however, all the processes in the ready queue are lined up waiting for a chance to run on the CPU. The records in the queues are generally process control blocks (PCB) of the processes [5].

2.3.1 PREEMPTIVE SCHEDULING

CPU- scheduling decisions may take place under the following four circumstances:

1. When a process switches from the running state to the waiting state(e.g. as the result of an I/O request or an invocation of wait for the termination of one of the child processes)
2. When a process switches from the running state to the ready state (e.g. when an interrupt occurs)
3. When a process switches from the waiting state to the ready state(e.g. at completion of I/O .
4. When a process terminates for situations 1 and 4, there is no choice in terms of scheduling. A new process (if one existing in the ready queue) must be selected for execution. There is a choice, however, for situations 2 and 3.

When scheduling takes place only under circumstances 1 and 4 , we say that the scheduling is non-preemptive or cooperative ; otherwise, it is preemptive. Under non-preemptive scheduling once the CPU has been allocated to a process, the process keeps the CPU until it release the CPU either by terminating or by switching to the waiting state.

Cooperative scheduling is the only method that can be used on certain hardware platforms, because it does not require the special hardware such as a timer needed for preemptive scheduling. Unfortunately, preemptive scheduling incurs a cost associated with access to shared data.

2.4 SCHEDULING CRITERIA[6], [7]

Different CPU scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another, In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. Many criteria have been suggested for comparing CPU scheduling algorithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best. The criteria include the following:

CPU utilization: We want to keep the CPU as busy as possible. Conceptually, CPU utilization can range from 0 to 100 percent. In a real system, it should range from 40 percent(for a lightly loaded system) to 90 percent (for a heavily used system).

- **Throughput:** if the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput. For long processes, this rate may be one process per hour; for short transactions, it may be 10 processes per second.
- **Turnaround Time**: From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time.
- **Waiting Time:** The CPU scheduling algorithm does not affect the amount of time during which a process executes or does I/O; it affects only the amount of time that process spends waiting in the ready queue. Waiting time is the sum of the periods spent in the ready queue.
- **Response Time:** In an interactive system , turnaround time may not be the best criterion. Often, a process can produce some output fairly early can continue computing new results while previous results are being output to the user. Thus another measure is the time from the submission of a

request until the first response is produced. This measure, called response time, is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device.

III. IMPLEMENTATION

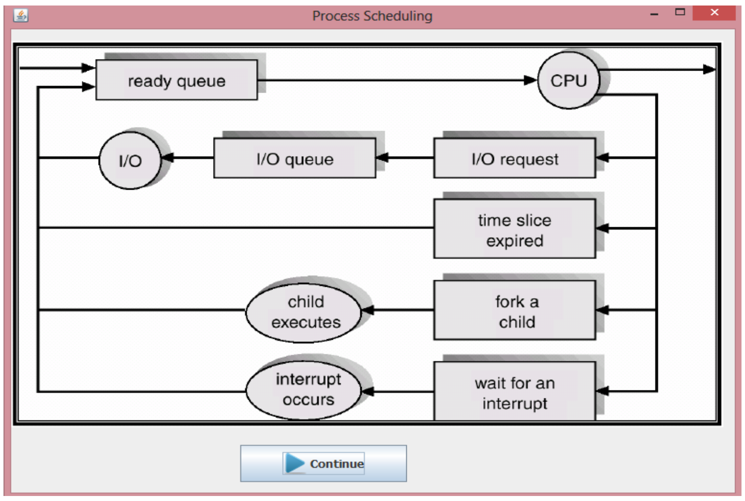
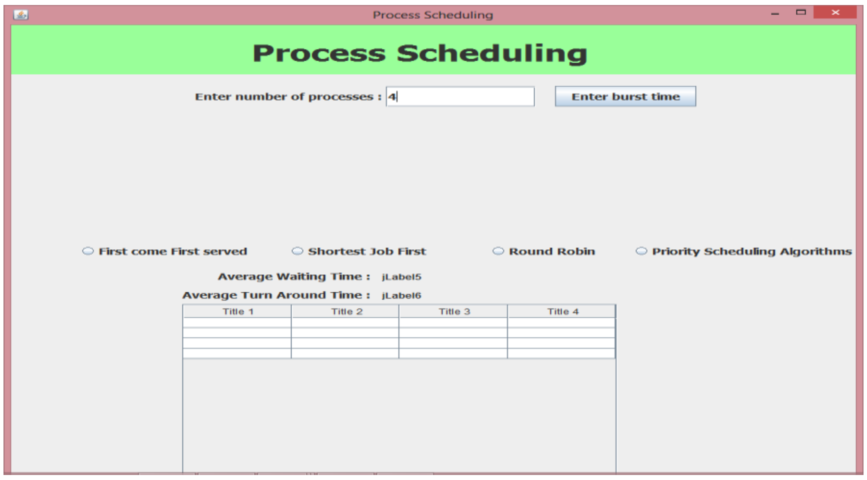


FIGURE 3. HOME PAGE



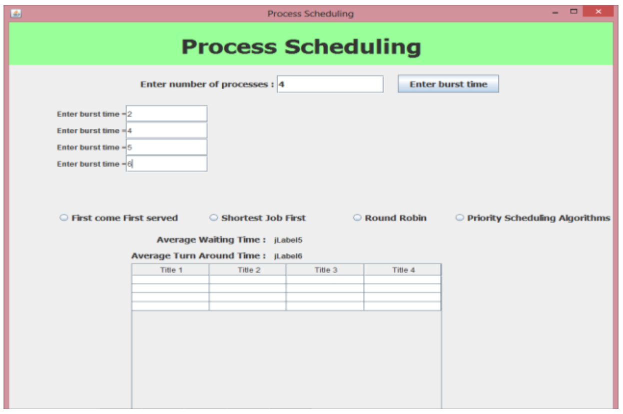
Enter number of processes : 4

Average Waiting Time : jLabel5

Average Turn Around Time : jLabel6

Title 1	Title 2	Title 3	Title 4

Figure 4. Process Selection



Enter number of processes : 4

Enter burst time -2

Enter burst time -4

Enter burst time -5

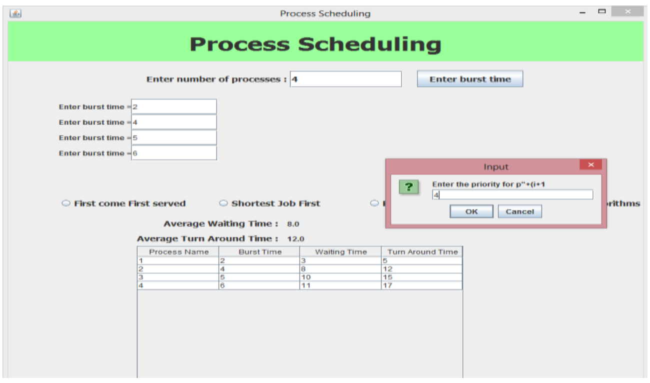
Enter burst time -6

Average Waiting Time : jLabel5

Average Turn Around Time : jLabel6

Title 1	Title 2	Title 3	Title 4

Figure 5. Burst Time Generation



Process Scheduling

Enter number of processes : 4

Enter burst time =2
Enter burst time =4
Enter burst time =5
Enter burst time =6

Enter the priority for p"+(i+1
4

Average Waiting Time : 8.0
Average Turn Around Time : 12.0

Process Name	Burst Time	Waiting Time	Turn Around Time
1	2	3	5
2	4	8	12
3	5	10	15
4	6	11	17

Figure 6. Process Execution

IV. CONCLUSION

The treatment of shortest process in SJF Scheduling tends to result in increased waiting time for long processes. And the long process will never get served, though it produces minimum average waiting time and average turnaround time. It is recommended that any kind of simulation for any CPU scheduling algorithm has limited accuracy. The standard scheduling algorithms FCFS, SJF, RR, and Priority scheduling have formed the basis of industry application of CPU schedulers. The industry implementation of schedulers have employed variations based on flavours and hybrids of the standard algorithms. As with most NP-hard problems, in designing the standard algorithms, scholars did not concern themselves with security matters.

REFERENCES

[1]. Abraham Silberschatz , Peter Baer Galvin , Greg Gagine , "Operating System Concepts", 7th edition 153-166
[2]. Leo J. Cohen , " Operating System", 5th edition 309- 373
[3]. Michael Kifer , Scott A. Smolka , "Introduction To Operating System Design And Implementation" 3rd edition 54-72
[4]. M. Naghibzadeh " Concepts And Techniques" 101- 134
[5]. Sudhir Kumar "Encyclopedia Of Operating System" 160- 205
[6]. Silberschatz, A. P.B. Galvin and G. Gagne (2012), Operating System Concepts .
[7]. Sabrian, F., C.D. Nguyen, S. Jha, D. Platt and F.Safaei, (2005).

International Journal of Engineering Science Invention (IJESI) is UGC approved Journal with Sl. No. 3822, Journal no. 43302.

Akanksha Verma Simulation of Process Scheduling Algorithms." International Journal of Engineering Science Invention(IJESI), vol. 6, no. 11, 2017, pp. 67-71.