Reprint **ISSN 2076-3972 (Web Version)**

Institutional Engineering and Technology (IET)

(*Inst. Engg. Tech.*)

Volume: 5 **Issue: 1** **April 2015**

Inst. Engg. Tech. 5(1): 25-29 (April 2015)

DEVELOPING A SIMPLE, INTERACTIVE AND EASY SIMULATION TOOL FOR THE TEACHERS/UNDERGRAD STUDENTS WITH A VIEW TO TEACHING/LEARNING CPU SCHEDULING ALGORITHMS

A.M. NITU, M.P. UDDIN, M.M. ISLAM AND M.S. ISLAM

An International Scientific Research Publisher

Green Global Foundation©

Web address: http://ggfjournals.com/e-journals archive
E-mails: editor@ggfjournals.com and editor.int.correspondence@ggfjournals.com

ISSN-2076-3972 (Online)
Inst. Engg. Tech. 5(1):25-29(April 2015)

DEVELOPING A SIMPLE, INTERACTIVE AND EASY SIMULATION TOOL FOR THE TEACHERS/UNDERGRAD STUDENTS WITH A VIEW TO TEACHING/LEARNING CPU SCHEDULING ALGORITHMS

A.M. NITU[1*], M.P. UDDIN[2], M.M. ISLAM[3] AND M.S. ISLAM[4]

[1]Assistant Professor, Department of Computer Science and Information Technology, HSTU, Dinajpur, Bangladesh; [2]Lecturer, Department of Computer Science and Information Technology, HSTU, Dinajpur, Bangladesh; [3]Assocoate Professor, Department of Physics, HSTU, Dinajpur, Bangladesh; [4]Assistant Professor, Department of Electrical and Electronic Engineering, HSTU, Dinajpur, Bangladesh.

[*]Corresponding author & address: Adiba Mahjabin Nitu, E-mail: nitu.hstu@gmail.com
Accepted for publication on 25 March 2015

ABSTRACT

Nitu AM, Uddin MP, Islam MM, Islam MS (2015) Developing a simple, interactive and easy simulation tool for the teachers/undergrad students with a view to teaching/learning CPU scheduling algorithms. *Ins. Engg. Tech.* 5(1), 25-29.

Operating System is one of the most significant courses for under-graduate students of Computer Science. In this paper a very simple, interactive and easy simulator has been developed for the teachers as well as the under-graduate students of computer science to teach as well as learn various CPU scheduling algorithms used for Operating System course. The key objective of using this simulator is to teach the students about the basic CPU scheduling algorithms visually. This will provide a clear and better understanding about how a scheduling policy executes. An under-graduate student will run the simulator and visualize how the processes get attention of the CPU following different scheduling policies.

Key words: *Simulator, Simulation Tool, CPU Scheduling, Operating System, Visualization of Scheduling Algorithms, Computer Science*

INTRODUCTION

An operating system course is one of the significant courses for the under-graduate students of Computer Science to complete his/her degree. However, CPU scheduling is one of the key concepts of this course. A student should have clear understanding about the execution of processes running in an operating system. There is a number of CPU scheduling algorithms whose task is to allocate the CPU to a process for a while based on some criteria. A simulation tool with visual representation of these algorithms definitely aids a student in better understanding of these CPU scheduling algorithms.

This paper is the design and development of a simulation tool to implement CPU scheduling algorithms for a single CPU. The simulator presents the algorithms graphically, and the performance metrics calculated in this simulator allows a student to measure and compare the behavior of those algorithms.

OVERVIEW

A. CPU scheduling algorithms

CPU scheduling is the allocation of the CPU to a process for use while the execution of another process is on hold i.e., in waiting state due to unavailability of any resource like I/O etc. thereby making the full use of CPU. The main goal of CPU scheduling is to make the system efficient, fast and fair (Singh Undated). CPU scheduling is done following some algorithms which are known as CPU scheduling algorithms. CPU scheduling algorithms offer proper and efficient use of the CPU. Different types of criteria are measured to check which scheduling algorithm performs better than others such as CPU utilization, throughput, turnaround time, waiting time, load average, and response time etc. The mostly used CPU scheduling algorithms are First Come First Served (FCFS), Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Priority Scheduling Algorithm (Pri), Round Robin (RR), Multilevel Queue, Multilevel Feedback Queue etc. Some of the basic algorithms are illustrated below (Singh Undated; Dadfar 2002; Silberschatz *et al*. 2006; Mohtashim Undated; Venkateswara 2013).

1. **First Come First Serve (FCFS)**

It is the simplest CPU scheduling algorithm. Here, processes are executed on first come, first serve basis. The process that requests the CPU first is allocated the CPU first. This algorithm is easy to understand and implement. It is a non-preemptive scheduling algorithm. The main advantage of FCFS is less context switching overhead. But there are some limitations which are the throughput can be low, turnaround time, waiting time and response time can be high, and no prioritization occurs etc.

2. **Shortest Job First (SJF)**

This algorithm is used to overcome the limitations of FCFS. SJF chooses the process with smallest burst time to execute the next. It can be preemptive as well as non-preemptive scheduling algorithm. The main limitation is that it is very difficult to know the burst time of next CPU request. Although SJF is optimal, it cannot be implemented at the level of short-term CPU scheduling.

3. **Round Robin (RR)**

In this algorithm, each process gets a small unit of CPU time, known as time quantum q, usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue. If there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time

in chunks of at most q time units at once. No process waits more than (n-1)q time units. Here, timer interrupts every quantum to schedule next process.

For example, consider a scheduling task consisting of four processes in the ready queue to be processed as given in the following table:

Process	Burst Time (ms)
P1	21
P2	3
P3	6
P4	2

If the FCFS algorithm is used, then the average waiting time of a process will be (0+21+24+30)/4=18.75 ms. The Gantt chart is shown below:

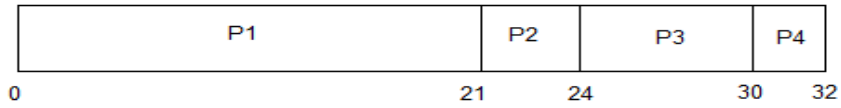

Fig. 1. Gantt chart using FCFS algorithm

If the non-preemptive SJF algorithm is used, then the average waiting time of a process will be (0+2+5+11)/4=4.5 ms. The Gantt chart is shown below:

Fig. 2. Gantt chart using non-preemptive SJF algorithm

If the RR algorithm with q=5ms is used, then the average waiting time of a process will be [(31-21)+5+(20-6)+13]/4=10.5 ms. The Gantt chart is shown below:

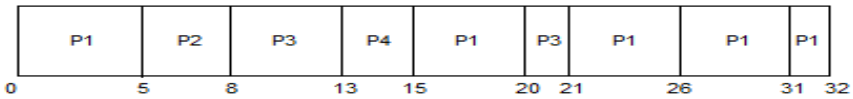

Fig. 3. Gantt chart using RR algorithm

B. Computer simulation

Simulation is the imitation of a real world system. The way of presenting a real world system by a computer program is called computer simulation (Oren 2011). The computer program is known as a simulator or a simulation tool. This is a very powerful technique to model an existing or proposed system. Computer simulation is widely used to understand and analyze the behavior of a system (Pidd and Carvalho, 2006 and Breitenecker and Troch, Undated). Gradually computer simulation is getting more acceptances due to its flexibility to present complex systems with various scenarios.

Depending on how the states of the system are changed, simulation is classified broadly in two categories: continuous system and discrete system. If the system variables change continuously over time, it is called continuous simulation. In discrete simulation, the system variables change only at discrete point of time (Banks 2000).

In this paper, a discrete event simulation tool has been developed to simulate CPU scheduling algorithms for a single CPU for teaching and learning purposes.

RELATED WORKS

Developing simulation tool for Computer Science education is one of the common and useful works. Some simulators developed for CPU scheduling algorithms are mentioned here. The author in (Suranauwarat 2012) developed a Java based CPU scheduling simulator with two different modes: simulation mode and practice mode. The user can execute the algorithms in simulation mode and check his prediction about execution of the algorithms in practice mode. The CPU scheduling simulator described in (Salih and Fadil, 2009) has graphical presentation of some scheduling algorithms with short description of each of them. A CPU scheduling algorithm simulator has been developed by (Suranauwarat 2007). (Saleem and Javed, 2000) has developed another simulation tool for CPU scheduling algorithms. GangSim, a simulator for grid scheduling studies has been proposed by (Dumitrescu and Foster, 2005). (Goyal *et al*. 1996) proposed a Hierarchical CPU scheduler for multimedia operating systems. (Gbarnett 2008) proposed a CPU scheduling simulator which is available at online. (Casanova 2001) has developed a toolkit for the simulation of application scheduling and named Simgrid.

The user has to enter a lot of or detail process information as input for most of the simulators mentioned above in this section. This is time consuming and the user might lose interest in using the simulators. Taking this issue in consideration, this paper plans to design a simulator with minimal input settings; the simulator generates process information automatically and randomly.

SIMULATION RESULT

The simulator can simulate three CPU scheduling algorithms: FCFS, RR and SJF. The simulator is very simple and easy to use. The simulator is developed using Java programming language on Netbeans 8.0.1 IDE. Figure 4 shows the structure of the simulator.

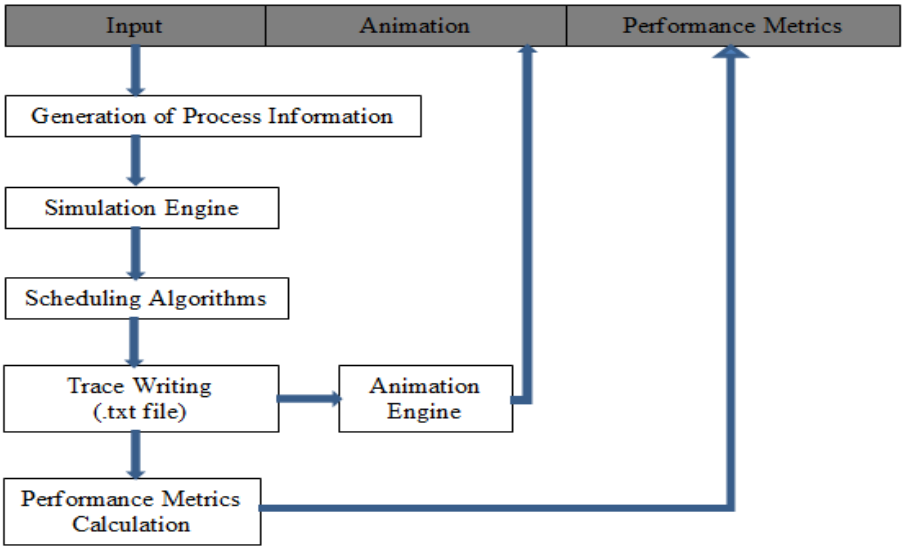

Fig. 4. Structure of the simulator

The simulator starts with a Graphical User Interface (GUI) (indicated as gray boxes). At first the user enters the number of process in the Input Panel (Figure 5). The user can set any value starting from zero. The simulator displays an error message if a negative number is given. Then the user has to press the "Generate" button. At this point, process information (process ID, arrival time and expected execution time) is generated randomly by the simulator for each process. Process ID starts with 1. This information is saved in a text file (.txt) according to process ID. Next the simulation engine reads the text file and executes the scheduling algorithms till all the processes' execution is done. Finally the execution trace for all processes is saved in individual text file for each algorithm. The process execution trace file has the following information for each process: process ID, execution start time for each quantum, and execution end time for each quantum. Each simulation clock unit is considered as a time quantum. Execution trace file serves as input for animation and performance metrics. The simulator displays a message if both traces are generated successfully.

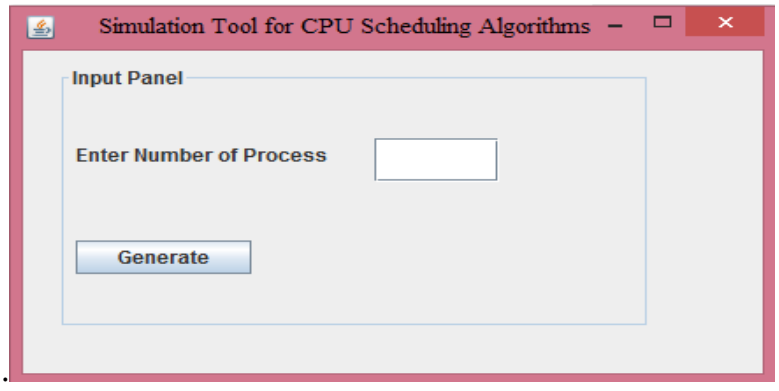

Fig. 5. Input panel

Once the trace generation is done, a new window appears to display the animation and the performance metrics. This window has three buttons, two for animation; "Complete Scheduling" and "Individual Process Scheduling", and the last button for "Performance Observation". In complete scheduling, the execution of all processes will be displayed for all three scheduling algorithms. Figure 6 shows a screenshot of complete scheduling. The animation is displayed in terms of a time scale. Each unit of the time scale is a time quantum. The green lines show the execution sequence of each algorithm, i.e., a green line with a process number (the

number shown in the upper side of the green lie) shows that which process is using the CPU for that time quantum. A scroll bar will be appeared if there is more number of processes. The user can scroll the animation panel left and right to see the execution sequences of a process for all algorithms.

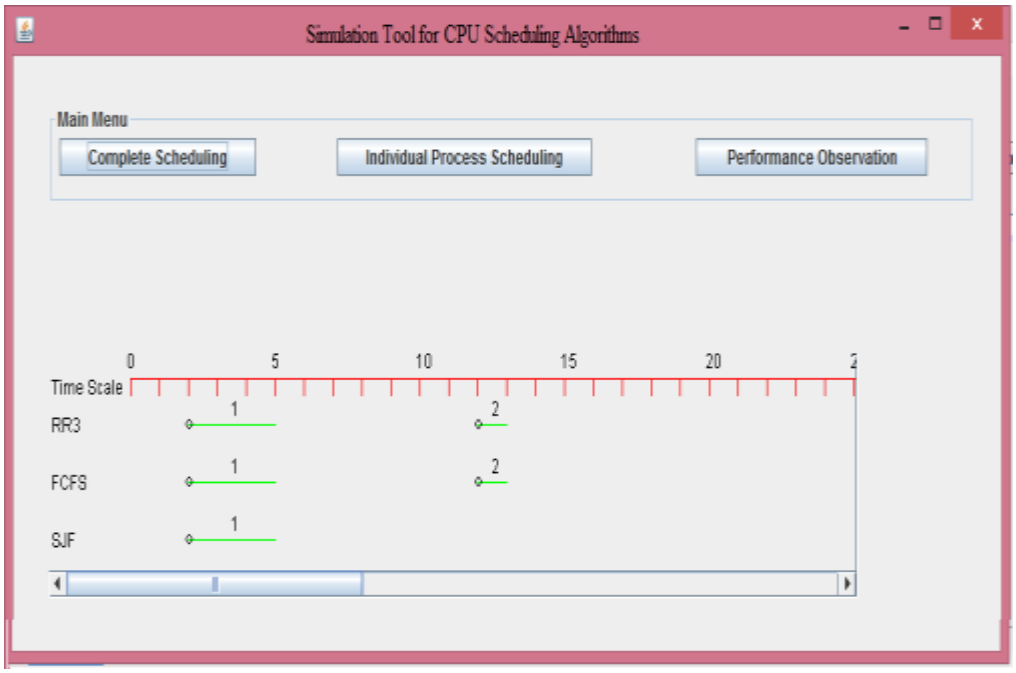

Fig. 6. Complete scheduling

In the individual scheduling, the user is asked to enter a process ID and the execution of that process will be shown for all three algorithms. Through animation the user visualizes which process is getting attention of the CPU at the current time and how does each scheduling algorithm work. In case if the user misses any execution sequence, he can replay the animation as many times as he wants to see the execution sequence again.

If the "Performance Observation" button is pressed, the performance calculation engine reads the execution trace files, calculates performance metrics, and display in the Performance Observation window in a Table (Figure 7). The following metrics are calculated: average wait time, turnaround time, throughput, response time, and CPU utilization for each algorithm. Performance metrics are calculated to compare the performance among the algorithms. The user can see which algorithm is better in what situation.

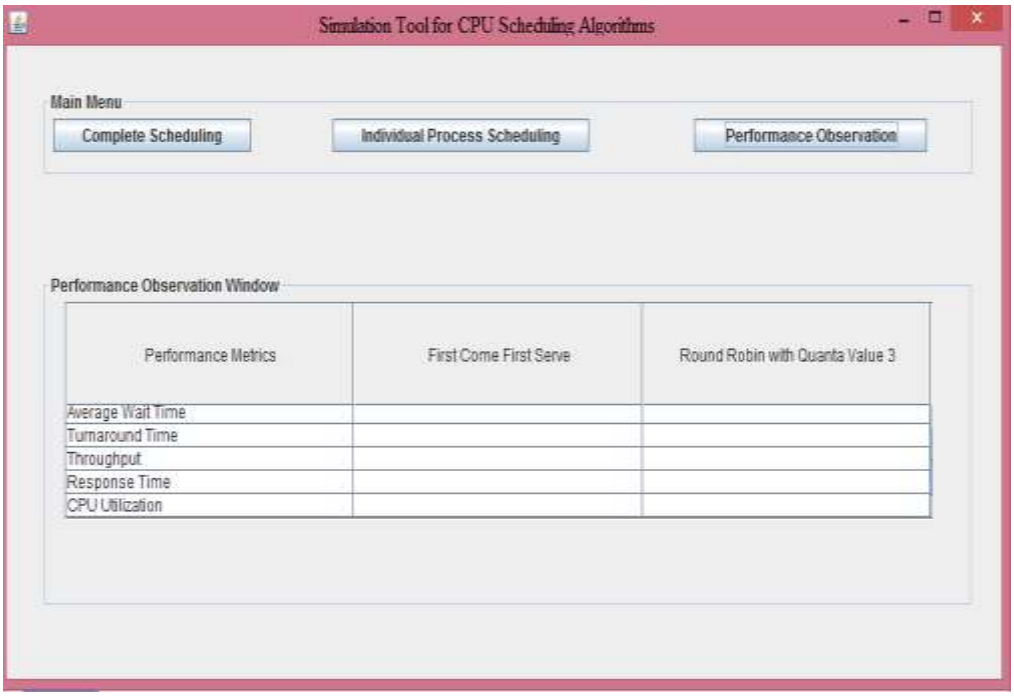

Fig. 7. Performance Observation Window

DISCUSSION AND CONCLUSION

The targeted user of this simulator is undergrad students who study an Operating System course. A teacher can use this simulator to teach students CPU scheduling algorithms. The strength and simplicity of this simulator is; (i) the user does not have to enter detail process information as input such as process ID, arrival time, and execution time for each process, (ii) with the increase of process number, the unit size of the time scale does not get shrunk which provides more clear vision of animation with more processes, and (iii) performance metrics presents comparison study of the algorithms. The user can use this simulator to implement other scheduling algorithms. More performance metrics could be added.

REFERENCES

Banks J (2000) Discrete-Event System Simulation. Prentice Hall International Series in Industrial and Systems Engineering. Prentice Hall, Upper Saddle, Third edition.

Breitenecker F, Troch I (Undated) Simulation Software – Development and Trends. *Control Systems, Robotics and Automation*. Vol. IV. pp. 233-280.

Casanova H (2001) Simgrid: a toolkit for the simulation of application scheduling. *First IEEE/ACM International Symposium on Cluster Computing and the Grid*. Brisbane, Australia.

Dadfar MB (2002) A Comparison of Common Processor Scheduling Algorithms. *Proceedings of the American Society for Engineering Education Annual Conference & Exposition.* 7(25), 1-7.

Dumitrescu CL, Foster I (2005) GangSim: a simulator for grid scheduling studies. *IEEE International Symposium on Cluster Computing and the Grid*. Cardiff, United Kingdom.

Gbarnett (2008) CPU Scheduling Simulator. Obtained from the webpage (https://cpuss.codeplex.com/).

Goyal P, Guo X, Vin HM (1996) A Hierarchical CPU Scheduler for Multimedia Operating Systems. *USENIX 2nd Symposium on OS Design and Implementation.* Washington, USA, pp. 107–122.

Mohtashim M (Undated) Learn Operating System. Obtained from the webpage (http://www.tutorialspoint.com/operating_system/index.htm).

Oren T (2011) The many facets of simulation through a collection of about 100 definitions. SCS, M&S Magazine. pp. 82-92.

Pidd M, Carvalho A (2006) Simulation software: not the same yesterday, today or forever. *Journal of Simulation.* 1(2), 7-20.

Saleem U, Javed MY (2000) Simulation of CPU scheduling algorithms. *IEEE Tencon Proceedings*. Kuala Lumpur, Malaysia.

Salih HM, Fadil YA (2009) CPU Scheduling Simulation. *Diyala Journalof Engineering Sciences.* 2(2), 39-52.

Silberschatz A, Peterson JL, Galvin PB (2006) Operating System Concepts. 7th Edition. Addison Wesley.

Singh A (Undated) Operating System. Obtained from the web page (http://www.studytonight.com/operating-system/).

Suranauwarat S (2007) A CPU scheduling algorithm simulator. *37th IEEE Annual Frontiers in Education Conference - Global Engineering: Knowledge without Borders, Opportunities without Passport.* Milwaukee, West Indies.

Suranauwarat S (2012) The Design and Development of a CPU Scheduling Algorithm Simulator. *Proc. of 12th WSEAS International Conference on Applied Computer Science.* Singapore. pp. 164-170.

Venkateswara GR (2013) Operating systems & Digital Logic Circuits. Obtained from the webpage (http://www.gitam.edu/eresource/comp/gvr%28os%29/gvr.htm).