MPRI-JSD | INTERNATIONAL JOURNAL OF AFRICAN SUSTAINABLE DEVELOPMENT
VOL. 10 NO.2 DEC-2019 ISSN: 2760-4106

CPU SCHEDULING SIMULATION ALGORITHM USING POISSON DISTRIBUTION RANDOM NUMBER GENERATOR

JOHN TEMITOPE OGBITI[1], SILOKO ISRAEL UZUAZOR[2], HENRY C. UKWUOMA[2] AND UDDIN O. OSEMENGBE[3]

[1]Department of Mathematics and Computer Science, Edo University Iyamho, Nigeria. [2]Department of Mathematics and Computer Science, Edo University Iyamho, Nigeria. [3]The National Institute for Policy and Strategic Studies, Kuru, Jos Plateau State, Nigeria.

KEYWORDS: *CPU Scheduling, Poisson distribution, Multiprogramming, Processors, Simulation*

Abstract

An operating system is a program that manages the computer hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the system hardware. CPU scheduling is the basis of multiprogramming systems. It refers to a set of policies and mechanisms to control the order of work to be performed by a computer system. The basic idea is to keep the CPU busy as much as possible by executing a user process until it must wait for an event, and then switch to another process. The problem of determining which processors should be assigned and to which processes is called CPU scheduling. How do we select a CPU Scheduling algorithm for a particular system? Since we have different scheduling algorithm with its own parameter selection can be difficult. To select an algorithm we must first define the relative importance of CPU Scheduling criteria. Next we use an evaluation method. This paper presents an algorithm and a life simulation of the CPU Scheduling algorithms using poison distribution to generate the random numbers for the burst times, arrival times and processes with Ms Visual Basic 2010 for the Scheduling algorithms and comparing their average waiting time to know which has the least average waiting time.

Introduction

CPU Scheduling is the task of selecting a waiting process from the ready queue and allocating the CPU to it. The CPU is allocated to the selected process by a dispatcher, giving the process autonomous use of the CPU and access to all the resources during the allotted time based on the algorithm used.

In this presentation, we simulate three types of CPU scheduling algorithm which are the First-come, first-serve (FCFS), Shortest-Job-First SJF), and the Round-Rubin to determine which of these scales most.

First-come, First-served (FCFS) scheduling is the simplest scheduling algorithm, but it can cause short processes to wait for very long processes. The FCFS algorithm is non-preemptive.

Shortest-job-first (SJF) scheduling is provably optimal, providing the shortest average waiting time. The shortest-job-first algorithm is a special case of the general priority scheduling algorithm, which simply allocates the CPU to the highest-priority process. Both priority and SJF scheduling may suffer from starvation.

Round-Robin (RR) scheduling is more appropriate for a time-shared system. RR scheduling allocates the CPU to the first process in the ready queue for a quantum time unit (q). After q time units, if the process has not relinquished the CPU, it is preempted, and the process is put at the tail of the ready queue. The major problem is the selection of the time quantum. If the quantum is too large, RR scheduling degenerates to FCFS scheduling; if the quantum is too small, scheduling overhead in the form of context-switched time becomes excessive.

Process State

As a process executes, it changes **state**. The state of a process is defined in part by the current activity of that process. A process may be in one of the following states:

- **New**. The process is being created.
- **Running**. Instructions are being executed.
- **Waiting**. The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
- **Ready**. The process is waiting to be assigned to a processor.
- **Terminated**. The process has finished execution

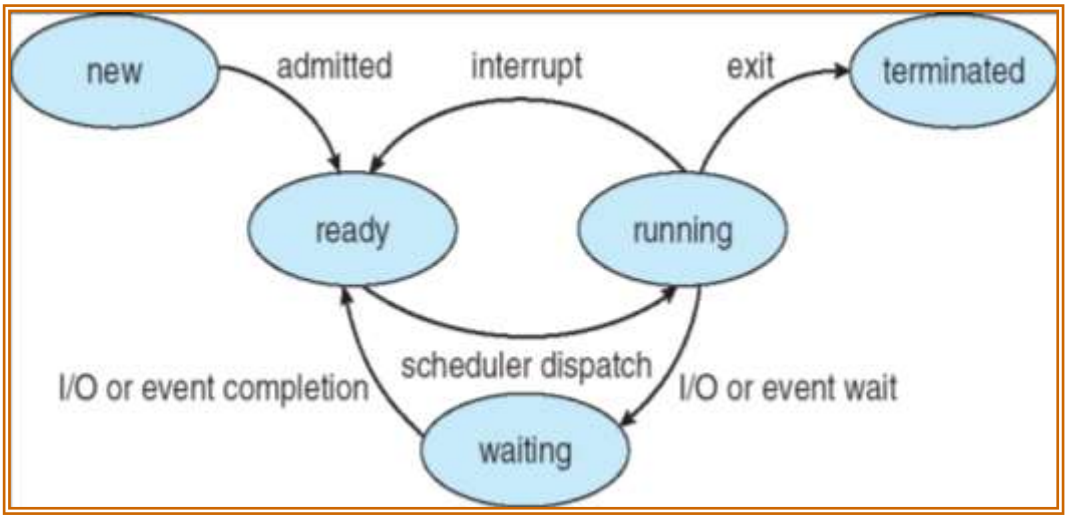
These names are arbitrary, and they vary across operating systems. The states that they represent are found on all systems, however. Certain operating systems also more finely delineate process states. It is important to realize that only one process can be *running* on any processor at any instant. Many processes may be *ready* and *waiting,* however. The state diagram corresponding to these states is presented in Figure below. (Silberschatz *et.al.* 2013)

INTERNATIONAL JOURNAL – ASD
VOL. 10 NO.2 DEC-2019 ISSN: 2760-4106

A multiprogramming operating system allows more than one process to be loaded into the executable memory at a time and for the loaded process to share the CPU using time-multiplexing.

- Switches from running to ready state
- Switches from waiting to ready

A Diagram Illustrating Process State (Silberschatz *et.al.* 2013)



Goals for CPU Scheduling

To make sure that scheduling strategy is good enough with the following criteria:

- **Utilization/Efficiency:** keeps the CPU busy 100% of the time with useful work.
- Throughput: maximizes the number of jobs processed per hour.
- **Turnaround time**: from the time of submission to the time of completion and minimize the time batch users must wait for output.
- **Waiting time:** Sum of times spent in ready queue.
- **Response Time**: time from submission till the first response is produced and minimize response time for interactive users.
- **Fairness**: make sure each process gets a fair share of the CPU.

The Random Number Generator

Poisson distribution is used in this simulation to generate randomly the necessary data required for the algorithms.

INTERNATIONAL JOURNAL – ASD
VOL. 10 NO.2 DEC-2019 ISSN: 2760-4106

Recall that if events, say arrivals to a queue, have inter-event times which are independent exponentially distributed random variables with rate ƛ, then the number of events N(t) up to time t has a Poisson distribution with parameter ƛ t:

$$P(N(t) = k) = \frac{e^{-\lambda}(\lambda t)^k}{K!}, \quad k= 0,1,2,...$$

So, if we wish to generate random variates having a Poisson distribution, we can do the following:

1 Count = 0
2 Sum = 0
3 while 1
4 Count = Count + 1
5 Sum = Sum + expon(alpha)
6 if Sum > 1 return Count-1

The arrival of jobs to the CPU for processing can be described as a Poisson distribution. The Poisson distribution is a discrete distribution that uses integers as random variables in which the occurrence of events given an interval is independent.

In this simulation, we considered three algorithms

- First Come First Serve (FCFS)
- Shortest Job First
- Round Robin

The CPU burst time, arrival time are random variables from Poisson distribution. To generate these variables, Poisson distribution requires the mean approximation of the distribution called lambda, and the number of events, in this case processes.

Installation

The software can be installed on any system with basic resources necessary to perform basic user task. In addition it is recommended that you have visual studio 2010 installed on your system before installing CPU scheduling simulation software. If you don't have visual studio 2010 or your system does not support visual studio 2010, you will not be able to run the application because the functionalities and platform for the software is the visual studio2010.

INTERNATIONAL JOURNAL – ASD
VOL. 10 NO.2 DEC-2019 ISSN: 2760-4106

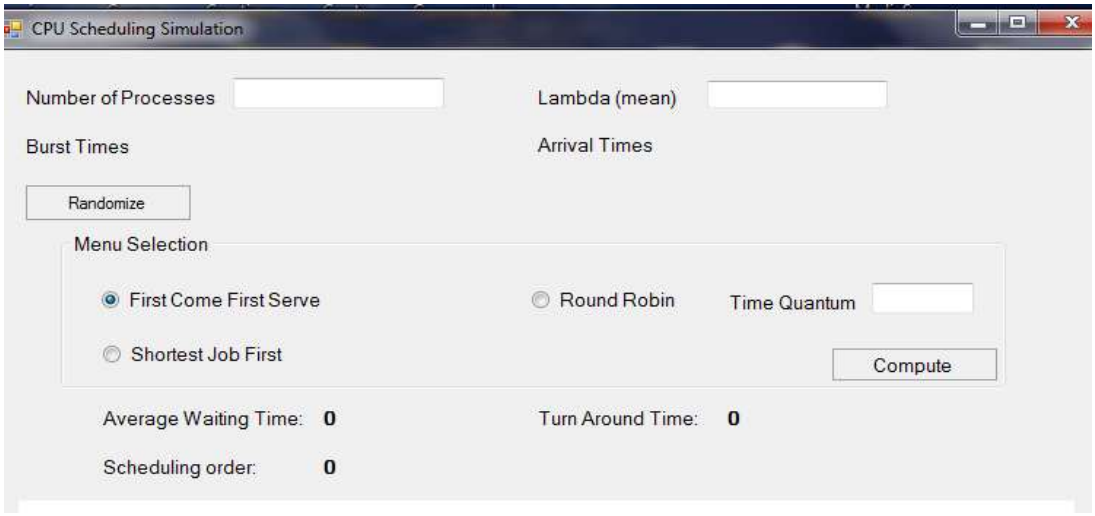
How to Install

To install the application, run the setup file on the disk and follow the wizard prompt.

Generating the Burst Time and Arrival Time

The module that handles the Poisson random number is an open source library called math.net. a reference was made to the class module. An install of the Poisson distribution class was made in the btnrandomize() method which generates the required burst time and arrival time.

Locate the executable file from start menu "CPU Scheduling", a simulation screen shown below will appear.



- Enter an integer value as the number of process. E.g 5
- Estimate the mean of distribution say 3
- Click on the randomize button.

The randomize button generate random values as the CPU burst time, ensure that there is no zero value in the CPU burst time. If zero appears, click on the button again until a better distribution is obtained.

Note that the arrival time for the processes is also generated. The initial arrival time for the first process is assumed to be zero. That is, no jobs were present when the first job arrived.

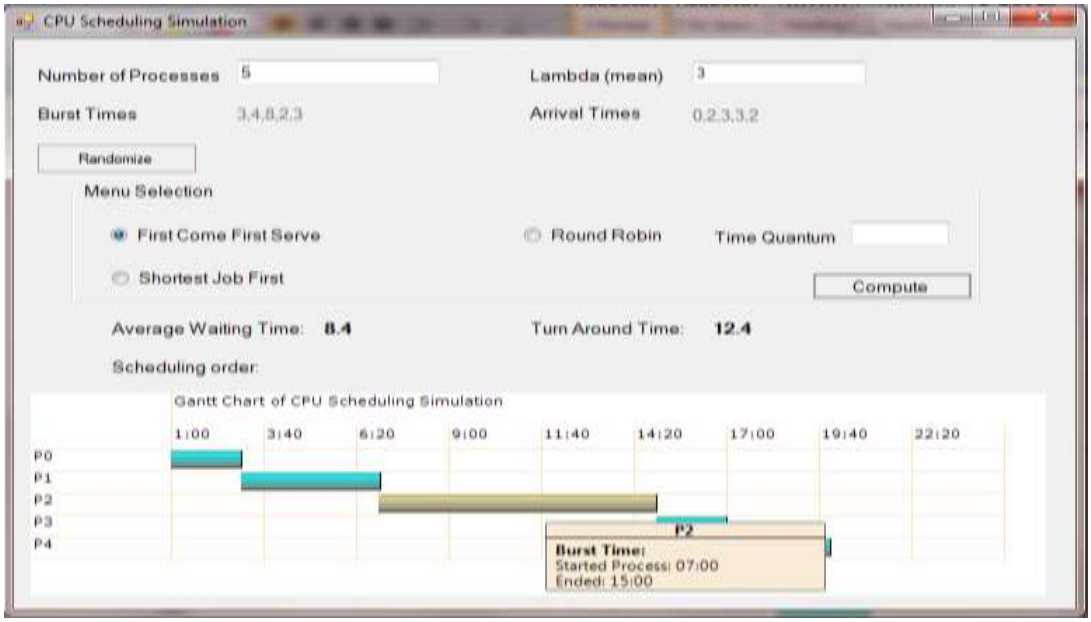
Simulating the Algorithm

To simulate any of the three algorithms provided, click on the algorithm and then click the compute button.

First-Come-First-Served:

The FCFS method handles the computation for the algorithm. A call to the Gantt chart method "gchart" was made to plot the Gantt chart for the algorithm. The gchart() method consist of a library component called test application that handles creation of bars for the chart. A class module bar info.cs creates the chart information that will be added to the Gantt chart.

Click on the first come first server option on the simulation screen which is by default selected. Click on the compute button. In this algorithm, the Gantt chart for the process is plotted as shown below.



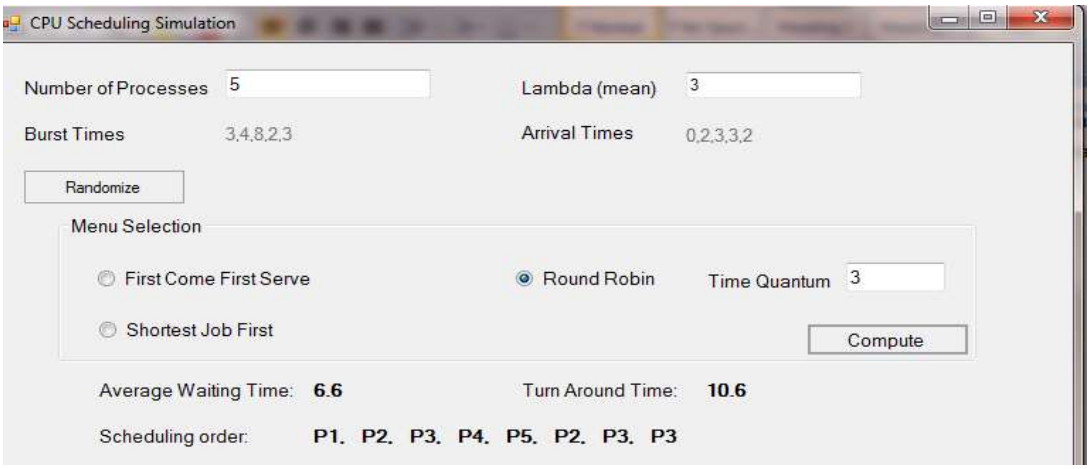
Notice that, placing your mouse over any of the processes gives a brief detail about process like when the process started and when it ended.

The average waiting time and turnaround time for the processes are also computed as shown on the simulation screen.

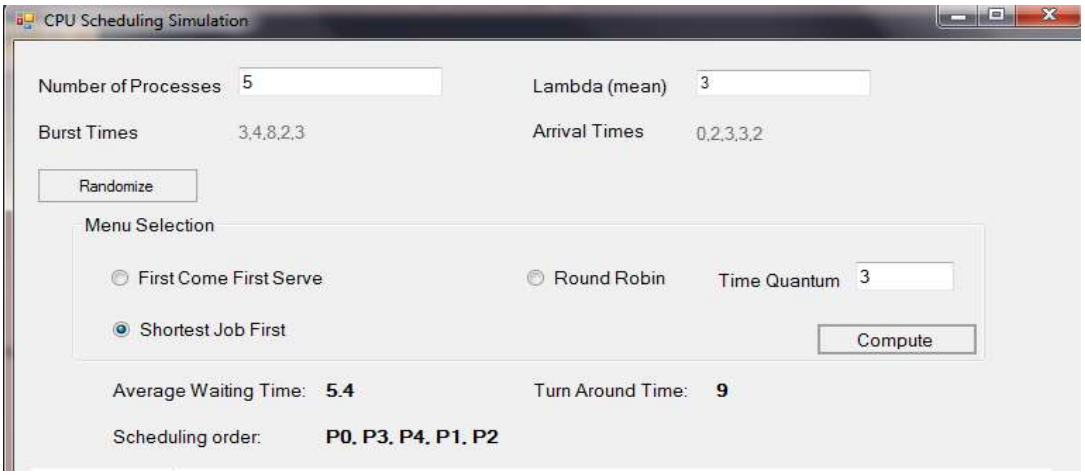
Round Robin

The round robin algorithm is handled by the RR() module on the code behind. The round robin algorithm is based on time slice called the quantum number. Enter an integer as the quantum number in the space provided, and using the random numbers generated click on the compute button. The average turnaround time and waiting time for the process will be computed. Note also that the scheduling order. See figure bellow.

INTERNATIONAL JOURNAL – ASD
VOL. 10 NO.2 DEC-2019 ISSN: 2760-4106



Shortest Job First: handled by the sjf() module. Select the algorithm type and click the compute button. See figure below



Conclusion

We used the random mean value of 3 to generate the burst time and the arrival time for five processes using the poisson distribution random number generator. From the simulation result shown above, FCFS scheduling algorithm perform worst with Turn Around Time of 12.4 and Average Waiting Time of 8.4 Round-Robbin algorithm perform better with a Turn Around Time of 10.6 and Average Waiting Time of 6.6 using the time quantum 3. Shortest Job First algorithm excels the most with a Turn Around Time of 9 and an Average Waiting Time of 5.4. We infer that Shortest Job First algorithm is the best of the CPU scheduling algorithm. The issue of starvation has always been of concern for considering SJF in the scheduling but that have been solved by introducing aging to jobs on the waiting queue. We therefore recommend that SJF CPU scheduling algorithm should be considered when designing an operating system.

INTERNATIONAL JOURNAL – ASD
VOL. 10 NO.2 DEC-2019 ISSN: 2760-4106

References:

Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne (2013) *Operating System Concepts*. John Wiley and Sons, inc. ninth Edition

Norm Matloff, (February 21, 2006) *Random Number Generation*

Lawrence Leemis and Steve Park (December, 1994) *Descrete-Event Simulation: A First Course*

Dr. S.E Abdullahi, (2012) *Advanced Operating System.* Unpublished Lecture note

www.matlab.net