This assignment’s objective is to take the next step on the formulation, programming, and analysis of computer simulation of discrete-event systems. In this assignment, you will be assessing an application in the area of computer science: simulating CPU scheduling. Three of the most important CPU Scheduling algorithms are FCFS, SRTF, and RR. You will be simulating those three algorithms and comparing their performances for different types of environments.

CPU Scheduling

Processes arrive to a computer system under the conditions described in 1.b. below. Processes ready to be executed enter a queue called ready queue. A process from the ready queue is selected and the CPU is allocated to it. This selection is performed according to a scheduling algorithm. In this programming assignment, we will consider three algorithms: First-Come-First-Served, Shortest-Remaining-Time-First, and Round Robin.

Generally, processes alternate between a CPU burst and an I/O burst. When a process finishes its current CPU burst, it waits until I/O completion. At the time the process issues an I/O command, the CPU is taken from it, and allocated to a different process. When the I/O request has been satisfied (at the end of the I/O burst), the process gets back to the ready queue, waiting for execution.

CPU Scheduling Simulation

In this programming assignment you will be simulating the CPU Scheduling process according to the following description:

1. The CPU Scheduling simulation abstracts the execution of programs in real computers in many respects.
   a. The details of instruction execution are overlooked. It is irrelevant to scheduling how many instructions are executed in a CPU burst or what they do. Execution is seen as a CPU burst, where the important variable is its length.
   b. Context switching is considered as executed instantaneously.
   c. The host operating system’s processes are not modeled.

2. A process is characterized by a tuple <pid, at, cput, bt, iot>, where
   a. pid is the process identifier, a sequential number.
   b. at is the process arrival time. Arrival times will be randomly generated by interarrival times, which will be uniformly distributed in the interval [0, maxIAt], where maxIAt is the maximum interarrival time in the system —
a parameter of the simulation.

c. cput is the cpu time the process needs for completion. cput will be randomly generated, uniformly distributed in the interval [0, maxCput], where maxCput is the maximum cpu time in the system – a parameter of the simulation.

d. bt is the maximum cpu burst time. That is, burst times for that particular process will be randomly generated, following a uniform distribution in the interval [0, bt], until cput is exhausted. bt for the different processes, in turn, will be generated randomly, following a uniform distribution in the interval [0, maxBt], where maxBt is the overall maximum cpu burst time – a parameter of the simulation.

e. cpu bursts alternate with I/O bursts. I/O bursts for that particular process are also randomly generated using a uniform distribution in the interval [0, iot]. iot for the different processes, in turn, will be generated randomly, following a uniform distribution in the interval [0,maxIot], where maxIot is the overall maximum I/O burst time – a parameter of the simulation.

3. Computer systems or environments can be classified according to the number of processes they run at a given time, the proportion of time they spent in CPU bursts and I/O bursts. Given that, a scientific application that performs some kind of complicated computation is called CPU-intensive, while a data base application that spends a lot of time reading information from disk is called I/O-intensive.

4. Other considerations about the simulation are:
   a. As in all simulations, the time or clock variable can be considered to be an integer variable. Events occur only at integer values of time.
   b. We will use the Event Scheduling/Time Advance algorithm to produce the simulation of CPU Scheduling in a computer system.

5. Before starting coding your simulation answer the following questions:
   a. How will the stream of events be generated?
   b. What happens when a newly created process arrives to the ready queue?
   c. What is the “customer arrival” process responsible for?
   d. Answer the same last two questions for departure.
   e. What happens to a process when the CPU burst finishes?
   f. How do you encode the event when a process relinquishes the CPU and waits for I/O?
   g. In the Round Robin algorithm (a preemptive algorithm), how do you encode the event when the scheduler preempts a process because its time quantum is expired?
   h. What data structures do you need for the FEL in the different scheduling algorithms?

6. The experiments you are to execute through simulation have the following characteristics:
   a. All parameters of the simulation must be read from a simulation configuration file.
b. In the debugging phase, test your system with as little as 10 processes, with small execution times, so that the complete simulation can be traced and debugged. Insert plenty of prints, or use a versatile debugger that allows you to perform inspections, stops, etc.

c. Once your program is “bug-free”, run several simulations with different “types” of system (at least one CPU-intensive, one I/O intensive, and one in-between). Perform simulations for at least 1000 processes each.

d. For each type of system you define, simulate each of the CPU Scheduling algorithms.

e. Report the following metrics:
   i. CPU utilization
   ii. Throughput
   iii. Turnaround time
   iv. Waiting time
   v. Response time – assume a process responds as soon as it starts execution

f. Graduate students or extra-credit only.
   i. Perform simulations for multi-processor systems (i.e., multi-server systems).
   ii. Run every experiment at least 30 times and report the minimum, mean, maximum, and variance of each metric. This statistical processing must be done outside the simulation process.
   iii. Make a histogram of CPU burst times. Compare for the different types of systems. Compare their means and variances.

g. Write up a report including:
   i. Introduction, including a description of the problem to be solved.
   ii. Description of the main software components.
   iii. Description of the experiments.
   iv. Output from the simulator, including the metrics mentioned above.
   v. Your conclusions, derived from the results of the simulation.
   vi. Please, include your name in every file (report and sources).
   vii. Your report must include proof of execution of the programs. I may run some of the programs if I decide to do so, but I will not be digging in source code, looking for the classes to be executed and to see whether or not they compile and run.

h. Pack the report and the source code in a single file and email it to me (juan@cs.uoregon.edu) before midnight of the due date. See the course policies about late assignments.