UNIVERSITY OF CALIFORNIA
College of Engineering
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

CS 162 Prof. Alan J. Smith

Assignment 1

In this assignment, you are to write a simulation of CPU scheduling. How to do this will be discussed in section.

You are to simulate a simple open queueing system that looks like this:

```
                             ______          _______
                                   |        |       |
----------->------>|||||||||--------> | CPU |------------>
                  ^          ______|        |_______|
                  |                                 |
                  |                                 |
                  |---------<---------<------|
```

1. Arrivals

You are to simulate the arrival of 1000 customers. Customers arrive with exponentially distributed interarrival times with mean interarrival time 1.0. Please generate the interarrival times *only once*; save them in an array and reuse the same interarrival times for each simulation. (Please discard any interarrival time that is less than 0.02 or greater than 40.0. If you have such a value, throw it away and generate a new one.) The *simulation ends* when the 1000'th customer arrives. Please compute the actual mean and standard deviation of the interarrival times, for all 1000, and print them out as part of your answer. Note that the system starts empty, and that the first customer arrives at time T1, (not 0); i.e. after the first interarrival time is added to zero.

Note: Use *at least* 32-bit arithmetic for all calculations; 64-bit would be even better. (That includes all random number generation, interarrival times and service times.) With 32-bit arithmetic, the chances that two numbers will be equal (e.g. that two events will happen at the same time) should be less than one in one million.

2. Service Times

The service time of customers will be hyperexponential, distributed as follows: With probability .8, the customer's service time will be exponential with mean 0.20. With probability .15, the customer's service time will be exponential with mean service time 1.0. With probability .05, the customer's service time will be exponential with mean 10.0. Please generate the service times only once, and save them for use in each simulation. (Please discard any service time that is less than 0.02 or greater than 40.0. If you have such a value, throw it away and generate a new one.) Please compute the actual mean and standard deviation of the service times, for all 1000, and print them out as part of your answer.

Note that for each simulation (below), the i'th customer will arrive at the same time, and will require the same amount of service time. Thus any difference in performance should be due to the different scheduling algorithms, not differences in the random numbers.

3. Scheduling Algorithms

You are to simulate the following scheduling algorithms. In each case, consider three cases: The time X to switch between processes is either 0.0, 0.01, or 0.10. (I.e. each simulation is run 3 times.) This overhead is incurred whenever: (a) there is a task switch from job A to job B or (b) a job arrives to an empty system. If (c) the clock interrupts at the end of a quantum and the same job is resumed, the overhead is X/2. (d) When a job arrives to a busy system, the overhead is X/2 if the currently running job continues (after the arrival of the new job), and is X if the running (currently active) job changes. Logically, you can consider the overhead in item (d) to consist of two parts: X/2 overhead to accept the new job, and put it in the job queue. X/2 overhead to start whichever job starts next, if it is a different job.

Overhead times are not atomic. If a job arrives in the middle of a task switch overhead, the task switch is suspended, the arrival is processed (with its own overhead), and then the prior task switch is resumed.

In each case, please compute the mean flow time for all customers that have completed service and left the system, the mean number of customers in the system (in the queue + in service), the standard deviation of the flow time, and the mean and standard deviation of f(i)/s(i), where s(i) is the service time for the i'th customer and f(i) is the flow time for the i'th customer. For the flow time computation, please ignore any customers left in the system after the 1000'th customer arrives. (I.e. your computation includes only customers who have completed service and have left the system.)

3.1. First Come, First Serve

Please simulate the system using FCFS scheduling, with each job run to completion.

3.2. Shortest Remaining Processing Time

Please simulate the system using SRPT scheduling. At any given time, you are to be running the job with the least remaining processing time.

Note: Consider the following case: You are running job A. Job B arrives, which is shorter than the remaining time of job A. You enter the task switch overhead to switch to job B. While in the task switch overhead, job C arrives, and C is shorter than the remaining time for B. You should process it the following way: Overhead X/2 to accept job B and put it in the job queue. Overhead of X/2 to accept job C and put it in the job queue. Overhead of X/2 to start job C.

3.3. Round Robin

Please simulate the system with round robin scheduling, using a time slice (quantum) of Q=0.0111, 0.11, and 1.0. (I.e. each of these simulations is run for each quantum size. Thus this item requires that you run 9 simulations.) If there is only one job in the system, the switching overhead (X/2) still occurs every time quantum, since the clock interrupts at that interval.

When a new job arrives, it always goes in the back of the queue.

3.4. Shortest Job First

Please simulate the system using SJF scheduling. Note that this is not preemptive; once a job starts, it runs to completion (except for overhead to accept new jobs and enter them in the queue).

3.5. Shortest Elapsed Time

Please simulate the system using SET scheduling. Note that when a job starts a period of execution, it will be allowed to run for some time Q (Q=0.0111, 0.11, and 1.0) before being interrupted, unless it completes first. (I.e. each of these simulations is run for each value of Q.) Note that an arriving job isn't eligible to run until the quantum for the currently running job ends.

3.6. Data

As noted above, you should generate your own arrival and service times. For debugging purposes, we will *also* provide you with a sample set of arrival and service times. They will be in the ~cs162/Homework directory. You should *also* show the results of all of your simulations using the provided (sample) arrival and service times.

4. Writeup

Please put together a table showing your simulation results in some clear and useful manner. Please do a writeup, *approximately one page single spaced*, commenting and explaining on what you've observed from the simulation runs. Please relate what you observed to what was presented in lecture. Part of your grade will depend on the quality and clarity of your writeup; it should be in grammatically correct standard English.

5. Frequently Asked Questions

Below are the answers to some questions that have occurred.

1. If the first job ends at the same time the second job enters the queue, is this considered entry into an empty system, assuming no other jobs in the queue?

This shouldn't happen, but if it does, the answer is "no."

Note: If two events appear to happen at identical times, then assume that the one that was entered in the event list first happens first. But in general, there should *never* be genuinely simultaneous events - if you are using 32-bit or 64-bit arithmetic, what are the chances that two numbers are exactly the same??

2. Is the "arrival time" of a job affected by the overhead associated with getting the job into the system? Do we say a job has not arrived until the work has been done to introduce it into the system?

No, the arrival time is simply (arrival time of j(i-1)) + (interarrival time of j(i)).

3. In the timesliced algorithms, if a job arrives in the middle of an executing job's quantum,

a. is the overhead of the arriving job during or after the quantum?

b. does the running job get to run until the end of its quantum, or does the scheduler run right away?

c. does the running job get to run for its entire alotted time, i.e. t(end_of_quantum) + overhead of all arriving jobs, if overhead is incurred during a quantum?

a. during; b. the arriving job is put into the job queue immediately, and then the job continues to run to the end of its quantum; c. yes, the job runs for its entire alloted time, so the time of the end of the quantum will be affected by the number of jobs that arrive during it.

6. General Instructions

1. The assignment must be programmed in Java.

2. The following files, and _only_ the following files, must be submitted: program source, writeup, README (optional - special information for readers), makefile (optional). 3. If the program is run with no arguments, it should generate random data for 1000 customers as specified in parts 1 and 2 of the

assignment and run the simulation on that data.

If the program is run with the name of a file in the current directory as an argument, it should read data (interarrival and service times) from that file and run a simulation on that data.

4. The program should print the following to the standard output when run (with the table filled in, of course):

```
% hw1
mean interarrival time: xx
std dev of interarrival times: xx
mean service time: xx
std dev of service times: xx

    A   B   C   D   E   F   G
--------------------------------------------------------------------

FCFS   0
FCFS   0.01
FCFS   0.1
SRPT   0
SRPT   0.01
SRPT   0.1
RR     0      0.0111
RR     0      0.11
RR     0      1.0
RR     0.01   0.0111
RR     0.01   0.11
RR     0.01   1.0
RR     0.1    0.0111
RR     0.1    0.11
RR     0.1    1.0
SJF    0
SJF    0.01
SJF    0.1
SET    0      0.0111
SET    0      0.11
SET    0      1.0
SET    0.01   0.0111
SET    0.01   0.11
SET    0.01   1.0
SET    0.1    0.0111
SET    0.1    0.11
SET    0.1    1.0
```

key --
A: Switching overhead
B: Quantum (if applicable)
C: Mean Flow Time
D: Std Dev of Flow Time
E: Mean flow_time(i) / service_time(i)
F: Std dev of flow_time(i) / service_time(i)

G: Mean number of customers

7. Hints and Information

1. You should write an *event driven simulation*, **not** a clock or interval type simulation. What this means is that your simulator basically cycles through the following loop: (get event) (reset clock to event time) (update system statistics) (update system state) (generate any new events and put them on event list) (continue loop).

The TAs will talk more about event driven simulation in discussion section.

2. You can generate an exponentially distributed random variable with mean 1.0 by generating a Uniform (0,1] random number U, and computing (-ln (U)). ("ln" is the natural log.) If you want an exponentially distributed random number with mean K, multiply the one you generate by K.

3. Any further information about this (and other) assignment(s) will be given out in discussion section or posted online.