CS2303 Systems Programming Concepts A08

Program 3 40 Points

Due: September 25, 2008 at 11:59 p.m.

Event-Driven Simulation of a Processor Scheduling Queue in C

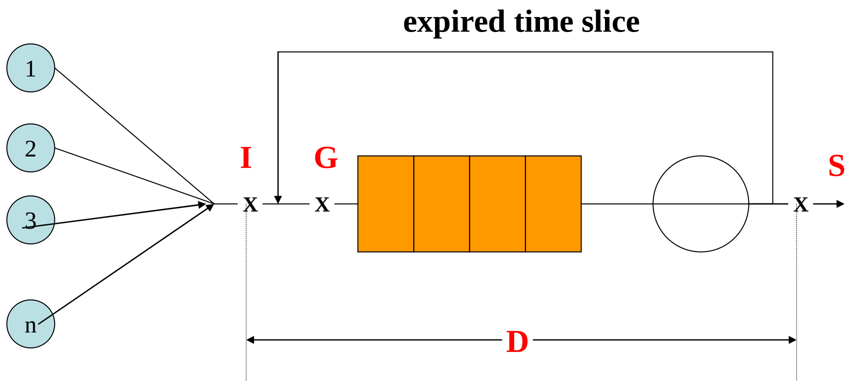
This assignment focuses on the use of data structures built in C with **structures.** One specific goal of this assignment is to give the students practice with C pointers.

Additionally, completing this assignment will require understanding the basics of event-driven simulation which is used prominently in performance analysis of computer systems and computer networks. The primary objective of this assignment is to use an event-list in an event-driven simulation to compare the performance of a FCFS (First Come First Served) scheduler against a RR (Round Robin) scheduler. The simulation topology consists of **n** processes arriving at the CPU scheduler for service (e.g. see the RR scheduler figure below). Assume queue sizes are unbounded in this program.

An FCFS scheduler serves customers in the order of their arrival to the scheduler queue. Each process runs on the CPU within interruption until completion.

Developed for multiprogrammed, time-sharing operating systems, a RR scheduler allocates up to a time slice (e.g., 100 milliseconds) of CPU service to the process at the front of the scheduler queue. In simple RR, arriving customers are enqueued at the back of the scheduler queue. If the process does not finish within a time slice, as indicated in the figure, the process returns to the back of the scheduler queue to await its turn for another time slice.

Round Robin Queue



Assignment Inputs

Your program begins by reading in two command line arguments source and time-slice to indicate respectively, the number of source processes to simulate and the time slice to be used in the RR simulation. For this simulator all time will be represented in 100 milliseconds units. Thus, a command line time-slice of 2 indicates a 200 millisecond time slice. Moreover, 100 milliseconds is the smallest time granularity of the simulator.

The program reads source lines of input from a script file (in ASCII) where each line of input contains:

```
process-id  arrival_time  cpu_time
```

where arrival_time and cpu_time are in simulation units (i.e. 100 milliseconds).

For example, the input line:

```
2166  30  205
```

indicates that process 2166 arrives at the scheduler at simulated time 3 seconds needing 20.5 seconds of CPU service.

Since the input script will drive both the simulation of FCFS and RR, your program should first store all the input script into memory.

Main Assignment

The assignment requires the use of an event list to simulate the performance of the CPU running processes for both FCFS and RR processor scheduling. In this simulation, to clearly understand the role of the event list, one needs an abstraction where the processor queue is separated from the current process being served.

The FCFS Scheduler

The FCFS scheduler is implemented as a queue data structure with arriving processes ***enqueued*** at the back of the queue. When the FCFS scheduler serves a process, the process node is taken from the front of the FCFS queue and its completion event is simulated by placing the process node in the event list.

The RR Scheduler

The RR scheduler is also implemented as a queue. When a new process is sent to the RR queue, it is ***enqueued*** at the back of the queue. When the RR scheduler serves a process, it executes the process for one time slice or for the remaining process execution if it is smaller than a time slice. To simulate time slice execution, the process node is updated and taken from the front of the RR queue and the completion event associated with the time slice is added to the event list. The process node needs to be updated to reflect the remaining processing time needed by the process ***after*** the time slice has expired.

In the case of multiple processes being placed in the scheduler queue at identical simulated times, use process-id as the time-breaker. For example, given process 3 and then process 6 scheduled arrive at the CPU scheduler at the simulated time 12, the process 6 node should be placed ***behind*** the process 3 node in the scheduler queue. This rule applies to both the FCFS and the RR scheduler.

The Event List

Pure event-driven simulation is controlled by a linked list known as the event list. While there are other ways to drive a simulation which can be used in Program 5, for this assignment you must use an event list. Future events are inserted into the event list in chronological order with the next event in simulated time at the front of the list. In the case of multiple events on the event list with identical event times, use the process-id as the time-breaker. For example, given two events, one for process 3 and one for process 6 that are both scheduled to occur at the simulated time 12, the process 3 node should be taken off the event list before the process 6 node.

In this assignment, the event list will hold two event types, process arrivals and the completion time of the process currently running on the CPU. Initially, the event list is populated with nodes indicating the arrival time of each process. The simulation runs in a continuous loop processing the next event at the front of the event list until the event list is empty. Processing an arrival event involves taking the entry off the event list and sending the process node to the CPU scheduler. Processing a CPU completion event involves first determining whether the process still requires more CPU time at the end of the time slice (for the RR scheduler). If the process has completed, process statistics are calculated and recorded before the process is terminated. If the process still needs more CPU time (for the RR scheduler), the process is enqueued at the back of the scheduler queue. In either case, taking the completion event off the event list ***triggers*** a call to the processor scheduler. If there is a process at the front of the scheduler queue, it is taken off the front of the scheduler queue and a new event with the appropriate completion time is inserted into the event list.

Assignment Output

Your final output consists of a single file annotating the simulation of the first FCFS and RR queuing mechanisms. Namely, the output logs the simulated arrival time and completion time of each process in simulated chronological order. Once the simulation ends, output the mean and variance of process response time for both queuing mechanisms. Note – no C file I/O is required for this assignment. The output file is redirected from standard output by Unix on the command line.

What to turn in for Program 3

An official test file will be made available a few days before the due date. Turn in your assignment using the ***turnin*** program on the CCC machines. You should turn in a tarred file that includes your source code, a make file and a README file. The README file should provide any information to help the TA or SA test your program for grading purpose.