Program 3 {modified February 9, 2014} 42 Points
Due: February 13, 2014 at 11:59 p.m.
Event-Driven Simulation of a Processor Scheduling Queue in C

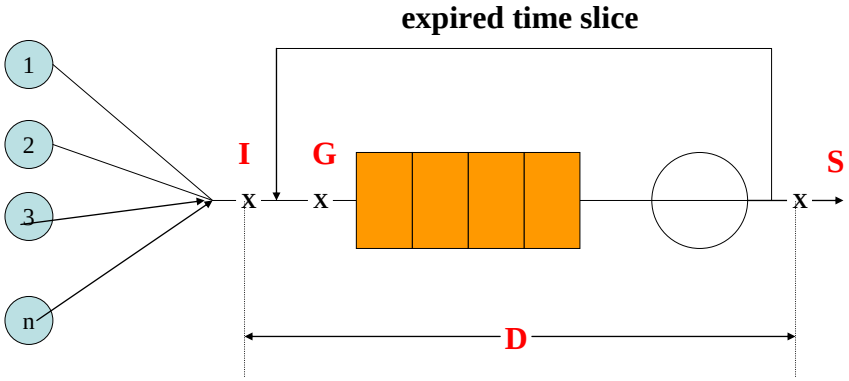
This assignment focuses on the use of data structures built in C with **structures.** One specific goal of this assignment is to give the students practice with C pointers.

Additionally, completing this assignment will require understanding the basics of event-driven simulation which is used prominently in the performance analysis of computer systems and computer networks. The primary objective of this assignment is to use event-driven simulation to compare the performance of a **FCFS (First Come First Served) scheduler** against a **RR (Round Robin) scheduler. If a student chooses to implement an additional, optional preemptive RR scheduler they can receive 6 extra credit points!** 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.

Developed for multiprogrammed, time-sharing operating systems, a **RR** scheduler allocates up to a time slice (e.g., 100 milliseconds) of CPU service per time slice 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 its 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



Preemptive RR is a variant of **RR** designed to give priority to very short jobs. Upon the arrival of a new process at the scheduler queue, preemptive RR preempts the currently running process in mid time slice. The newly arriving process then receives one initial time slice before being enqueued at the back of the scheduler queue. Once the preempting process completes, the preempted process is resumed to receive the remainder of its preempted time slice of service.

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** and **preemptive RR** simulations. **A time slice of 0 is an indicator that your program should simulate the FCFS queue for this run of the 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 granularity of the simulator.

Subsequently, the program reads `source` lines of input from an ASCII script file 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 millisecond units)

For example, the input line:

```
2166 30 204
```

indicates that process **2166** arrives at the scheduler at simulated time **3** seconds needing **20.4** seconds of CPU service (Note – I have converted milliseconds to seconds here!).

Since the input script will drive the simulation of **FCFS**, **RR** and optionally preemptive **RR**, your program should first store all the input script in memory such that the input script will be processed the appropriate number of times (2 or 3) .

Main Assignment

The assignment is to use an **event list** mechanism to simulate the performance of the customer script for the **FCFS, RR** and **preemptive RR processor** scheduling. Your final output consists of **two** or **three** files annotating the simulation of the queueing mechanisms. Namely, output to a file a time log that shows the simulated arrival time and completion time of all processes in simulated chronological order. **To simplify the I/O, you can execute your program once per each simulated processor**. The end of each output file should also record the mean and variance of customer response time for that queueing mechanism.

The Event List

Event-driven simulation is controlled by a linked list known as the 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 this assignment, the event list will hold two event types, process arrivals and the completion time of the process currently running on the CPU. The simulation runs in a continuous loop processing the next event at the front of the event list until the event list becomes empty. Processing an arrival event involves taking the entry off the event list and sending the process to the scheduler queue. Processing a CPU completion event involves first determining whether the process still requires more CPU time. If the process has completed, process statistics are calculated and recorded before the process is terminated. If the process still needs more CPU time, 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, 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.

The FCFS Scheduler

The **FCFS** scheduler is implemented as a queue data structure with arriving processes enqueued at the back of the queue.

The RR Scheduler

The **RR** scheduler is also implemented as a queue with arriving processes enqueued at the back of the queue.

For both the **FCFS** and the simple **RR** scheduler, when a process arrives at the scheduler, the scheduler needs to check if another process is currently running. If the CPU is idle, the arriving process is sent to the event list with the appropriate completion time. Note, it is the difference in completion time in the event list which is different for the **FCFS** and the **RR** scheduler.

The Preemptive RR Scheduler {Extra credit ONLY - 6 points}

The **preemptive RR** scheduler is also implemented as a queue. When a new process is sent to the **RR** queue, a check needs to be made to determine if another process is running. An indication of a currently running process implies taking its completion event off the event list and putting the interrupted process somewhere until the preemption is over. In either case, a new completion event is added to the event list for the newly arriving process with the appropriate completion time.

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, **your two output log files (one for FCFS and one for RR),** 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 **(e.g., you need to indicate how you entered ties in arrival time into your event list)**.