CSCI-4210 Operating System Fundamentals

Project 3: Simple Operating System Simulator
Due Date: November 30

Introduction:
Operating systems are analyzed using three different techniques. First, we can build a queueing model and analyze the system mathematically (Queueing Theory). Secondly, we could build mechanisms into the operating system which would provide feedback regarding its performance. Once this feedback is rendered, we can then fine-tune the system. The final and most commonly used technique is simulation.

In this project, you will simulate a simple operating system using a technique known as **discrete event simulation**. You will gain insight into how jobs move through the system, changing states and accessing services. Most importantly, you will gather various statistics in order to analyze the efficiency of the system, find bottlenecks, and suggest improvements to the scheduling algorithm in order to decrease turnaround time.

System Description:
Consider a simple computer with one CPU, one I/O device, one page fault server, and a particular Multi Programming Level (MPL). The CPU has a Ready Queue associated with it, the I/O device has an I/O Queue associated with it, and the Page device has a Page Queue associated with it. A number of users sit at terminals, "sending" jobs into the system. Jobs arrive one at a time, requesting to enter the system. If the system is full (number of jobs in the system == MPL), then the job will wait for access to the system in an additional queue, the Arrive Queue.

Once a job is in the system, it attempts to gain use of the CPU. Scheduling is accomplished by use of the Round-Robin algorithm. If the CPU is being used by another job, then the requesting job is placed in the Ready Queue. Otherwise, the requesting job acquires the CPU, and runs until one of four events occurs:

Job completion The job has used the CPU for as long as it needs to, and can now be destroyed. Another job from the same terminal should then be created, and the Arrive Queue should be checked for jobs waiting to enter the system. (See arrow "A" in figure).

Time slice interrupt Execution of the job has been interrupted by having its time slice expire. In this case, the job is moved off of the CPU and to the tail of the Ready Queue. Its time slice is then reset. (See arrow "B" in figure).

I/O interrupt The job has requested use of the I/O device. In this case, the job is moved off of the CPU as it requests to use the I/O device. If the I/O device is busy, it waits in the I/O queue until it can get the device. Once done with its I/O operation, the job moves back to the Ready Queue. (See arrow "C" in figure).

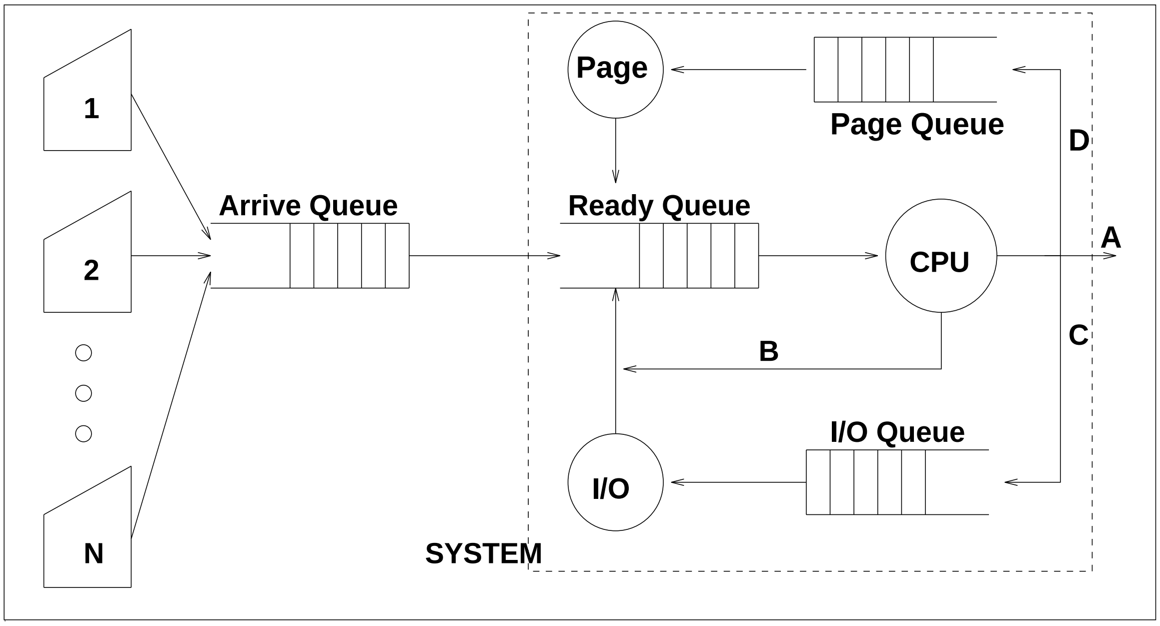

Figure 1: *Simple Operating System*

Page-fault interrupt The job requiers a page that is currently not in memory. In this case, the job is moved off of the CPU as it requests to use the Page-fault server. If the Page-fault sevrer is busy, it waits in the Page queue until it can get the server. Once the Page fault has been serviced, the job moves back to the ready Queue. (See arrow "D" in figure).

In all four of these cases, the CPU is freed of the current job, and the Ready Queue is examined for waiting jobs. The default scheduling algorithm you must implement is Round-Robin scheduling, where a time-slice can interrupt a job, and the head of the Ready Queue determines which job will run next.

Simulation Parameters:
When started up, your simulator should prompt the user for various system parameters. These parameters are outlined below.

t_{SIMUL}	Total simulation time (float, ms, constant)
n_{USERS}	Number of terminals/users (int, constant)
MPL	Multi-Programming Level (int, constant)
$ALGO$	CPU scheduling algorithm (0 = Round-Robin)
$\overline{t_{THINK}}$	Average thinking time (float, ms, exponential distribution)
$\overline{t_{CPU}}$	Average time a job needs to use the CPU (float, ms, exponential distribution)
t_{SLICE}	Time slice for Round-Robin scheduling (float, ms, constant)
$\overline{t_{IO}}$	Average I/O interarrival time (float, ms, exponential distribution)

t_{SIO}	I/O service time (float, ms, constant)
t_{Page}	Average Page fault interarrival time (float, ms, exponential distribution)
t_{SPage}	Page fault service time (float, ms, constant)

Simulation Output Results:
After running for the required amount of time, your simulation ends, and a report should be printed with the following statistics displayed.

- Number of jobs completed
- Number of I/O operations completed
- Number of Page faults serviced
- Average throughput
- Average turnaround time
- Amount and percent of CPU time spent on executing processes
- Amount and percent of CPU time spent idle
- Amount and percent of total simulation time spent on I/O operations
- Amount and percent of total simulation time spent on Paging
- For *each* of the four queues (Arrive, Ready, I/O, and Page):

Implementation Notes You are allowed to use any discrete event simulation package.

Project Submission:
Your documentation should discuss (in great detail) the results you obtained given various input parameters for the system.

Grading Policy:

Proper execution/statistics	60
Report with Design suggestions	20
Documentation	20

One week (7 days) early submission earns 5 bonus points.