Simulation Lab 1: Process Scheduling Simulation Exercises on Feb. 22.

Submission to the bb4.utc.edu is due on midnight of Feb. 24.

Name your submission file with firstName_lastName_Simulation1.

Turn In:

Include your experiences and answers to Part I (question 1 to 12) and Part II (question 1 to 9).

Include the following at the beginning of your report.

- Name: ____
- UTC ID: ____
- Course Number and Name: _________________________________
- Semester: ____
- Lab Name and Number: ___________________________________
- I spent __________hours and ________minutes to finish this hands-on lab.
- I have __________ (percent) finish this lab.
- I expect _________ (A, B, C, or F) of this lab.
- This lab helps me to understand process scheduling. Choose a number to indicate how much the lab is helpful.
  1 2 3 4 5
  (less helpful) (more helpful)

Part I
Instructions: Download the Process Scheduling simulator and extract the zip file. This produces a folder named **ps**. The user manual for this simulator, `ps_doc.html`, is included in the folder. It is strongly suggested that you carefully read through this documentation describing how the simulator operates prior to beginning this section.

Steps: Perform the following steps using the process scheduling simulator:

1. In the **ps** directory, execute `runps`(UNIX, Linux, Mac OS X) or `runps.bat` (Windows.) This will start the process scheduling simulator.

2. Click on the green button labeled **Run Experiment** in the lower left-hand corner of the Process Scheduling Simulator window. This will start the simulation which will complete within one second. This simulation involves creating 30 processes using the SJF (shortest-job-first) and FCFS (first-come-first-served) scheduling algorithms.

3. Terminate the simulator using the pink **Quit** button in the lower right-hand corner of the main window.

4. In the **ps** directory you will find the file **psconfig**. This is the configuration file for the process scheduling simulator. The fourth line in this configuration file begins with **user**. Change its default value of **Local User** to your name.

5. Execute the simulator again (as in Step 1).

6. Click the **Open Log** button in the top row of the center column of the main window to open a log file. After you have pushed this button, its name will change to **Close Log**.

7. Click the **Run Experiment** button again.

8. Click the **Show All Table Data** button at the right side of the main window. The Table Data window showing a table of statistics appears.

9. Click the **Log All Table Data** button in the middle column of the main window. The table of statistics is now sent to the log file.

10. Click the **Draw Gantt Chart** button at the right side of the main window. A pop-up window appears to allow you to draw the Gantt Chart for either the SJF or the FCFS scheduling algorithms. Choose the **Gantt Chart for FCFS button** and a Gantt chart window appears illustrating the states of the 30 processes. Click the **Log** button in the lower right-hand corner of the Gantt Chart for FCFS window to send the output of the Gantt Chart to your log file.

11. Return to the main window and click the **Draw Gantt Chart** button again, this time choose the Gantt Chart for SJF option. A Gantt chart illustrating the SJF algorithm of the states of the 30 processes is shown again.

12. Return to the main window and click the **Close Log** button and then the **Quit** button. You have created a file called **logfile.html** in the directory you started the simulator from.

13. Submit your log file -- **logfile.html** -- to your instructor per his or her instructions.

Questions:

Answer the following questions after completing Steps 1-13 above.

1. Examining the table generated in Step 8, what is the value of Key for myrun_1?
   a) FCFS
   b) SJF

2. Examining the table generated in Step 8, what is the CPU Utilization for myrun_2?
   a) .961584
   b) .961039
   c) .116711
   d) 10.07

This document was derived from simulation software created by Steve Robbins which was supported by NSF DUE-9752165

3. Examining the table generated in Step 8, which run has greater Throughput value?
   a) myrun_1
   b) myrun_2

4. Which scheduling algorithm has the shorter average turnaround time?
a) FCFS
b) SJF

5. Which scheduling algorithm has the longer maximum turnaround time?
a) FCFS
b) SJF

6. Examining the Gantt chart for FCFS generated in Step 10, at approximately what time did process 15 finish?
   a) 205
   b) 300
   c) 140
   d) 160

7. Examining the Gantt chart for SJF generated in Step 11, which process finished first?
   a) Process 1
   b) Process 15
   c) Process 16
   d) None of the above

8. Examining the Gantt chart for SJF generated in Step 11, which process finished last?
   a) Process 15
   b) Process 16
   c) Process 18
   d) None of the above

In the traditional UNIX scheduling algorithm the load average refers to the average number of processes in the ready queue, and can be calculated by using the table generated in Step 8. The load average is the total waiting time divided by the total time for the experiment. To determine the total waiting time, multiply the average waiting time by the number of processes. The total time for the experiment is the time the last processes finishes and is available in the table.

9. What is the load average for FCFS?
   a) 0.069
   b) 20.59
   c) 10.29
   d) 16.20

10. What is the load average for SJF?

a) 26.84
b) 0.34
c) 10.07
d) 20.59

The simulator does not take into account the context switch time required by the scheduling algorithm. The context switch time can be taken into account as follows: The number of context switches can be determined from the table generated in Step 8 under the heading **Entries**, subheading **CPU**. Each time a context switch occurs, the context switch time must be added to the waiting time of each process in the ready queue. The previous two questions required calculating the load average for the FCFS and SJF scheduling algorithms. Assuming the context switch time would add 10% to the average waiting time, the context switch time can be determined as follows:

(a) Let the context switch time be ***S***.
(b) If there are ***N*** context switches and the load average is ***L***, the total waiting time due to context switches is ***N*** × ***S*** × ***L***.
(c) Set ***N*** × ***S*** × ***L*** equal to 10% of the total waiting time and solve for ***S***.

11. What is the context switch time for FCFS?
a) 0.286
b) 529.23
c) 1853.10
d) 25.70

12. What is the context switch time for SJF?
a) 258.69
b) 30.21
c) 0.5
d) 0.285

Part II
Instructions: Download the Process Scheduling simulator and extract the zip file. This produces a folder named **ps**. The user manual for this simulator, **`ps_doc.html`**, is included in the folder. It is strongly suggested that you carefully read through this documentation describing how the simulator operates prior to beginning this section. It is also recommended that you complete Part I as well.

In the **ps** directory, there are two files: (1) **`myrun.run`** (the *run* file); and (2) **`myexp.exp`** (the *experiment* file.) The run file specifies various parameters such as the number of processes, interarrival times, process duration (how long it requires the CPU), and the length of CPU and I/O bursts. The experiment file specifies the number of experiments that are to be run using the run file. Each experiment in the experiment file runs the scheduler according to the parameters specified in the run file. For example, if the experiment file `myexp.exp` contains the following:

```
name myexp
comment This experiment contains 2 runs
run myrun algorithm FCFS key "FCFS"
run myrun algorithm SJF key "SJF"
```

this means the experiment specifies two runs will be performed, one using FCFS scheduling, the other using SJF scheduling.

If the run file **`myrun.run`** contains:

```
name myrun
comment This run specifies one type of process
algorithm FCFS
seed 5000
numprocs 20
firstarrival 0.0
interarrival constant 0.0
duration constant 100
cpuburst constant 10
ioburst constant 10
basepriority 1.0
```

this means each experiment will run the scheduler with 20 processes, all arriving at time 0. The duration of each process is 100, and the CPU and I/O bursts are both constant at 10.

The user manual for this simulator, **`ps_doc.html`**, fully explains the parameters for the experiment and run files. It is strongly suggested that you read through this manual before proceeding, as well as keep it open while performing the following exercises.

Steps: Perform the following steps using the process scheduling simulator:

1. Using a text editor, replace the contents of **`myexp.exp`** and **`myrun.run`** with the experiment and run files shown above. (Both the `myexp.exp` and `myrun.run` files are located in the **ps** directory.)

2. In the **ps** directory, execute **`runps`**(UNIX, Linux, Mac OS X) or **`runps.bat`** (Windows.) This will start the process scheduling simulator.

3. Click the **Run Experiment** button and then click the **Show All Table Data** button. Next, click the **Draw Gantt Chart** button, once for the FCFS, a second time for SJF.

<< Unless otherwise specified, set the value of `myrun.run` and `myexp.exp` back to the default values that are shown above. >>

Questions:

Answer the following questions after completing Steps 1-3 above.

1. Examining the output from Step 3, you should be able to easily determine that the scheduling behavior was no different between the FCFS and SJF algorithms. Explain why.

2. Click the **Run Experiment** button, and then create a Gantt chart for the FCFS, and a second Gantt chart for the SJF. In both Gantt charts, the last process finishes at time 2000. Explain why.

3. Change the value of **cpuburst** in `myrun.run` from **`cpuburst constant 10`** to **`cpuburst uniform 10 25`**. This has the affect of assigning CPU bursts randomly with values ranging from 10 to 25. Run the simulator with this changed parameter, and examine the Gantt charts for the FCFS and SJF. Do the FCFS and SJF algorithms schedule processes the same or differently?

4. In Section 5.3.1 of the text the **convoy effect** of FCFS scheduling is described. The following values of `myrun.run` specify two types of processes:

```
name myrun
comment This run specifies two types of processes
algorithm FCFS
seed 5000
numprocs 5
firstarrival 0.0
interarrival constant 0.0
duration constant 50
cpuburst uniform 1 5
ioburst constant 10
basepriority 1.0

numprocs 1
firstarrival 0.0
interarrival constant 0.0
duration constant 50
cpuburst constant 25
ioburst uniform 1 5
basepriority 1.0
```

The first type of process specifies 5 processes with a uniform CPU burst uniformly distributed between 1 and 5 and a constant I/O burst of 10. The second type of process is a single process with a constant CPU burst of 25, and a uniform I/O burst distributed between 1 and 5. Thus, there are 5 I/O-bound processes, and 1 CPU-bound process.

Change the value of `myrun.run` as shown above. Run the simulator with this changed parameter, and examine the Gantt chart for the FCFS. What process(es) are part of the convoy the second time process 6 gets the CPU?

a) Processes 1 through 5
b) Processes 1 through 6
c) Processes 2 through 5
d) There is no convoy effect in this situation.

Change the value of `myexp.exp` to the following:

```
name myexp
comment This experiment contains 5 runs
run myrun algorithm FCFS key "FCFS"
run myrun algorithm SJF key "SJF"
run myrun algorithm RR 1 key "RR 1"
run myrun algorithm RR 5 key "RR 5"
run myrun algorithm RR 10 key "RR 10"
```

This specifies 5 different runs of the scheduler, FCFS and SJF are the first two. The next three are round-robin (RR) with time quantums of 1, 5, and 10, respectively.

Change the value of `myrun.run` to the following:

```
name myrun
comment This contains one type of process
algorithm SJF
seed 5000
numprocs 20
firstarrival 0.0
interarrival constant 0.0
duration constant 50
cpuburst uniform 2 20
ioburst constant 10
basepriority 1.0
```

In this run file, there are 20 processes with a constant duration and I/O burst. The CPU burst will uniformly range between 2 and 20.

This document was derived from simulation software created by Steve Robbins which was supported by NSF DUE-9752165

Run the simulator with this changed parameter, and click the **Show All Table Data** button.

5. Which scheduling algorithm has most context switches?
   a) FCFS
   b) RR 1
   c) RR 5
   d) RR 10

6. Draw the Gantt chart for both the FCFS and SJF algorithms. Notice they look quite different. Yet in the table, both FCFS and SJF have the same number of context switches (109). Explain why.

Preemptive SJF (PSJF) scheduling is discussed in Section 5.3.2 of the text. PSJF is the situation whereby, if a newly arrived process has a shorter next CPU burst than the currently executing processes, the newly arriving process will preempt the process currently running. (SJF scheduling has no such preemption.) Add the following line to **`myexp.exp`**:

```
run myrun algorithm PSJF key "PSJF"
```

which adds an experimental run using PSJF scheduling. Run the simulator again, and examine the Table Data.

7. What can be said about the relationship between the number of context switches for the SJF and PSJF scheduling algorithms?
   a) They both have exactly the same number of context switches.
   b) They both have approximately the same number of context switches.
   c) SJF has more context switches.
   d) PSJF has more context switches.

8. For the 3 RR scheduling runs, what can be said about average waiting time with an increasing time quantum?
   a) Average waiting time increases with an increasing time quantum.
   b) Average waiting time decreases with an increasing time quantum.
   c) There is no correspondence between average waiting time and the length of the time quantum.

Section 5.3.4 of the text states that the performance of round-robin scheduling is heavily dependent upon the length of the time quantum. If the time quantum is large, the RR scheduling policy is the same as FCFS policy.

9. Assuming `myrun.run` stays the same as shown above, which of the following lines need to be added to the `myexp.exp` file for this to be true? (If necessary, modify `myrun.run`

with each answer choice to test for the correct answer.)

a) `run myrun algorithm RR 25 key "RR 25"`
b) `run myrun algorithm RR 15 key "RR 15"`
c) `run myrun algorithm RR 8 key "RR 8"`
d) None of the above lines will cause RR scheduling to behave the same as FCFS.